ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова
праця на правах рукопису

Савенко Олег Станіславович

УДК 004.75:004.49

ДИСЕРТАЦІЯ

ТЕОРІЯ ТА ПРАКТИКА СТВОРЕННЯ РОЗПОДІЛЕНИХ СИСТЕМ ВИЯВЛЕННЯ ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ

05.13.05 – комп'ютерні системи та компоненти

05 «Технічні науки»
(галузь знань)

Подається на здобуття наукового ступеня доктора технічних наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ О. С. Савенко

Наукові консультанти:
Саченко Анатолій Олексійович, доктор технічних наук, професор
Марковський Георгій, доктор наук, професор, м. Ролла, штат Міссурі, США

Ідентичність всіх примірників дисертації
ЗАСВІДЧУЮ:
Учений секретар
спеціалізованої
вченої ради _____ / *Я. І. Луцик*

Львів – 2019

АНОТАЦІЯ

Савенко О.С. Теорія та практика створення розподілених систем виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах.

Дисертація на здобуття наукового ступеня доктора технічних наук за спеціальністю 05.13.05 – комп'ютерні системи та компоненти – Національний університет «Львівська політехніка», Львів, 2019.

У дисертації запропоновано нові та отримали подальшого розвитку теоретичні основи та практика створення розподілених багаторівневих систем для виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах на основі децентралізації та самоорганізації, розроблено нові методи виявлення зловмисного програмного забезпечення, які покращують ефективність його виявлення за рахунок їх застосування в розподілених системах.

Наукова новизна одержаних результатів полягає в наступному:

1) удосконалено модель архітектури розподіленої системи виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах, яка відрізняється від відомих комплексним врахуванням вимог розподіленості, децентралізованості, багаторівневості та самоорганізованості, що дозволяє створювати на її основі розподілені системи та їх компоненти, які функціонуватимуть автономно і самостійно прийматимуть рішення про наявність зловмисного програмного забезпечення та нарощення своїх функціональних можливостей;

2) вперше розроблено модель архітектури типових компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення на основі структур Кріпке з представленням компонентів через стани, в яких вони можуть перебувати під час функціонування, що дає змогу враховувати перебування їх в різних станах і є основою для визначення стану безпеки всієї розподіленої системи та її компонентів;

3) вперше розроблено метод взаємодії компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення на основі підтримки її цілісності та визначення порядку передачі знань між її компонентами і використання встановлених аналітичних залежностей між рівнями безпеки програмних модулів та рівнем безпеки всієї розподіленої багаторівневої системи, що дозволяє системі автономно змінювати свою архітектуру та функції без втручання користувача, а також визначати стратегію своєї подальшої роботи;

4) удосконалено моделі зловмисного програмного забезпечення шляхом їх подання алгебрами поведінки, що дозволило створити базис поведінкових сигнатур, і, на відміну від відомих представлень, врахувати особливості функціонування зловмисного програмного забезпечення в локальних комп'ютерних мережах та здійснити його класифікацію за типами поведінки;

5) вперше розроблено метод виявлення бот-мереж у локальних комп'ютерних мережах, який базується на здійсненні активного моніторингу системних подій та узгодженій взаємодії компонентів розподіленої системи при прийнятті рішення, і, на відміну від відомих методів, дає можливість створення на його основі засобів, здатних інтегруватись в розподілену систему та класифікувати бот-мережі за їх поведінковими сигнатурами, що формуються закладеними в їх компоненти функціями;

6) вперше розроблено метод виявлення файлового зловмисного програмного забезпечення в локальних комп'ютерних мережах, який полягає в поєднанні роботи програмних агентів, що здійснюють виявлення зловмисного програмного забезпечення в окремих комп'ютерних системах, відповідно до імплементованих в них методів: динамічного формування поведінкової сигнатури шляхом відстеження викликів прикладного програмного інтерфейсу, знаходження поліморфного та метаморфного програмного коду, сканування виконуваних програм шляхом створення для них автономних процесів та відповідних програмних агентів у розподіленій системі, що, на відміну від аналогів, дозволяє покращити аналіз і підвищити достовірність виявлення

зловмисного програмного забезпечення;

7) вперше розроблено метод виявлення файлового зловмисного програмного забезпечення на основі динамічного формування поведінкової сигнатури шляхом відстеження викликів прикладного програмного інтерфейсу, в якому, на відміну від відомих методів, поведінкова сигнатура формується на основі критичних викликів прикладного програмного інтерфейсу за групами зловмисної активності та відображає частоту їх входження і характер взаємодії критичних функцій, що дає змогу виявляти нові версії відомого зловмисного програмного забезпечення не тільки за наявністю критичних викликів, але й за їх взаємодією між собою;

8) вперше розроблено метод виявлення поліморфних та метаморфних вірусів з використанням функцій заплутування програмного коду, відмінністю якого є поетапний аналіз і порівняння функціональних блоків програмного об’єкта та його змінених версій, отриманих, в тому числі, від різних компонентів розподіленої системи шляхом їх взаємодії між собою.

Практичне значення одержаних результатів полягає в наступному: 1) розроблено архітектуру і компоненти розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах, здійснено їх програмну реалізацію, а також розроблено апаратно-програмні засоби захисту інформації в складі розподіленої багаторівневої системи, використання яких регламентується вимогами безпеки; 2) результати експериментальних досліджень підтверджують ефективність розроблених програмних засобів, а також правильність наукових положень теорії розподілених систем, оскільки впровадження розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення дозволяє підвищити достовірність виявлення на 5–12 % порівняно з відомими аналогами та знизити рівень помилок першого роду до 5 %; 3) теоретичні та практичні результати роботи впроваджено в Державному підприємстві «Новатор», ТОВ «ІТТ – telecommunication company», ТОВ «ЮКС++», компанії CYPRESS SEMICONDUCTOR та навчальному процесі Хмельницького національного

університету при викладанні дисциплін «Безпека та захист комп’ютерних систем», «Технічна діагностика і надійність комп'ютерних пристроїв та систем», «Паралельні та розподілені обчислення» та «Системне програмне забезпечення».

У першому розділі проведено аналіз технологій розвитку зловмисного програмного забезпечення, мережних систем його виявлення, існуючих антивірусних програмних засобів та відомих методів. В результаті проведеного аналізу зроблено висновок, що для покращення ефективності виявлення зловмисного програмного забезпечення є необхідним розвиток теорії і практики створення розподілених систем на основі децентралізації та самоорганізації у локальних комп’ютерних мережах.

Другий розділ присвячено методологічним засадам створення розподілених систем виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах на основі децентралізації та самоорганізації.

Методолічні засади створення розподілених систем виявлення зловмисного програмного забезпечення поєднують в них комплексне врахування розподіленості, децентралізованості, багаторівневості та самоорганізованості, що дає можливість автономного функціонування як системи так і її компонентів.

На основі розробленої методології пропонується:

удосконалена модель архітектури розподіленої системи виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах, яка відрізняється від відомих комплексним врахуванням вимог розподіленості, децентралізованості, багаторівневості та самоорганізованості, що дає можливість автономного функціонування як системи так і її компонентів;

нова модель архітектури типових компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення на основі структур Кріпке дає змогу враховувати перебування їх в різних станах, що є основою для визначення стану безпеки всієї системи та її компонент;

новий метод взаємодії компонентів розподіленої багаторівневої системи на основі підтримки цілісності системи та встановлення порядку передачі знань між компонентами системи і використання отриманих аналітичних виразів для

визначення рівнів безпеки програмних модулів та розподіленої багаторівневої системи, що дозволяє динамічно здійснювати гнучку зміну архітектури системи та впливати на подальші її дії без втручання користувача, а також визначати подальшу стратегію роботи системи.

В третьому розділі здійснено формалізацію характерних властивостей зловмисного програмного забезпечення з використанням теорії абстрактної алгебри, на основі чого було розроблено алгебри поведінки зловмисного програмного забезпечення та типів загроз, що дало змогу отримати:

удосконалені моделі зловмисного програмного забезпечення на основі їх подання алгебрами поведінки для створення бази поведінкових сигнатур, класифікації за типами на основі врахування особливостей їх функціонування в локальних мережах.

В четвертому розділі виділено типові компоненти бот-мереж (мережного зловмисного програмного забезпечення) на основі функцій, здійснено характеристику моделей бот-мереж та їх компонентів, на основі чого запропоновано:

новий метод виявлення бот-мереж у локальних комп'ютерних мережах, суть якого в здійсненні активного моніторингу системних подій, класифікації та узгодженій взаємодії компонентів розподіленої системи при прийнятті рішення, для створення на його основі засобів, що інтегруватимуться в розподілену систему для підвищення достовірності виявлення та здійснення класифікації бот-мереж за їх поведінковими сигнатурами, що формуються закладеними в їх компоненти функціями.

В п'ятому розділі здійснено формалізацію моделей файлового зловмисного програмного забезпечення, на основі якої було запропоновано:

новий метод виявлення файлового зловмисного програмного забезпечення в локальних комп'ютерних мережах, який полягає в поєднанні роботи програмних агентів, що здійснюють виявлення зловмисного програмного забезпечення в окремих комп'ютерних системах, відповідно до імплементованих в них методів: динамічного формування поведінкової сигнатури шляхом

відстеження викликів прикладного програмного інтерфейсу, знаходження поліморфного та метаморфного програмного коду, сканування виконуваних програм шляхом створення для них автономних процесів та відповідних програмних агентів у розподіленій системі, що, на відміну від аналогів, дозволяє покращити аналіз і підвищити достовірність виявлення зловмисного програмного забезпечення;

новий метод виявлення файлового зловмисного програмного забезпечення на основі динамічного формування поведінкової сигнатури шляхом відстеження викликів прикладного програмного інтерфейсу для отримання можливості виявляти інші його різновиди, а також нові версії відомого, та здійснення формування поведінкової сигнатури на критичних викликах прикладного програмного інтерфейсу за групами зловмисної активності з відображенням частоти їх входження і характеру взаємодії критичних функцій;

новий метод виявлення поліморфних та метаморфних вірусів на основі аналізу функцій обфускації програмного об’єкту та його модифікованих версій, отриманих від різних компонентів розподіленої системи, та здійснення подальшого аналізу на основі пошуку еквівалентних функціональних блоків для здійснення більш детального аналізу коду програмного об’єкту на наявність поліморфного та метаморфного коду.

В шостому розділі розроблено програмне забезпечення розподіленої багаторівневої системи та її апаратно-програмні компоненти захисту інформації для реалізації запропонованих теоретичних основ таких систем. Здійснено підтвердження можливості їх практичного створення та використання в експериментальних дослідженнях для порівняння з відомими системами виявлення і впровадження розробленої розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення в локальних мережах в організаціях та підприємствах.

У висновках сформульовано основні результати дисертаційних досліджень. **Додатки** містять опис програмного забезпечення розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення,

результати роботи, опис апаратно-програмного пристрою, таблицю функції переходів зі стану в стан, таблицю стратегій для визначення подальших кроків роботи системи, наукові праці здобувача та акти впровадження результатів роботи.

Ключові слова: розподілені системи, зловмисне програмне забезпечення, засоби захисту інформації, самоорганізовані системи, децентралізовані системи, методи виявлення.

ABSTRACT

Savenko O.S. The theory and practice of creating distributed detection systems for malware in local computer networks.

Thesis for a Doctor of Technical Sciences degree by specialty 05.13.05 – Computer Systems and Components – Lviv Polytechnic National University, Lviv, 2019.

In the dissertation proposes new and further developed theoretical foundations and practice of creating distributed multilevel systems for malware detection on local computer networks based on decentralization and self-organization. New malware detection methods have been developed that improve its detection efficiency through their use in distributed systems.

The scientific novelty of the obtained results is as follows:

1) an improved model of the architecture of distributed malware detection system on local computer networks, which differs from the known complex requirements of distribution, decentralization, multilevel and self-organization, which allows to create on its basis distributed systems and their components that will function independently. make decisions about malware and build capabilities;

2) first developed a model architecture for typical components of a distributed multilevel malware detection system based on Kripke structures with representation of the components through the states in which they may be in operation, which allows them to be considered in different states and is the basis for determining the state of security the entire distributed system and its components;

3) first developed a method of interaction between components of a distributed multilevel malware detection system based on maintaining its integrity and determining the order of knowledge transfer between its components and using established analytical relationships between the security levels of software modules and the security level of the entire distributed multilevel system, allowing the system to autonomously change its architecture and functions without user intervention, and to determine the strategy of its further work;

4) improved malware models by presenting them with behavior algebras, which allowed us to create a basis for behavioral signatures and, in contrast to the known concepts, to take into account the peculiarities of malware functioning in local computer networks and to classify it by types of behavior;

5) for the first time developed a method of detecting botnets in local computer networks, based on the active monitoring of system events and coordinated interaction of components of a distributed system in decision making, and, unlike known methods, allows the creation of tools based on it, capable of integrating into a distributed system and classifying botnets by their behavioral signatures, formed by the functions embedded in their components;

6) the first method of detecting malware in local computer networks, which is to combine the work of software agents that detect malware in individual computer systems, in accordance with the methods implemented in them: dynamic formation of behavioral signatures by tracking API-calls, finding polymorphic and metamorphic code, scanning executable programs by creating for us x Autonomous processes and related software agents in a distributed system, which, unlike its counterparts, improves the analysis and improves the detection of malware;

7) first developed a method for detecting malware on the basis of dynamic behavioral signature formation by tracking API-calls, in which, unlike known methods, a behavioral signature is formed based on critical API-calls by groups of malicious activity the occurrence and nature of the interaction of critical features, allowing detection of new versions of known malware lky the availability of critical challenges, but also their interaction with each other;

8) a method for the detection of polymorphic and metamorphic viruses was developed for the first time using software entanglement functions, the difference of which is the step-by-step analysis and comparison of functional blocks of a software object and its modified versions, obtained, including from various components of a distributed system by their interaction between yourself.

The practical significance of the results obtained is as follows: 1) the architecture and components of the distributed multilevel malware detection system on

local computer networks were developed, their software implementation was implemented, as well as the hardware and software protection of information within the distributed multilevel system, the use of which is regulated by security requirements; 2) the results of the experimental studies confirm the effectiveness of the developed software, as well as the correctness of the scientific provisions of the theory of distributed systems, since the introduction of a distributed multilevel malware detection system allows to increase the reliability of detection by 5–12 % compared to known analogues and reduce the level of error to 5 %; 3) the theoretical and practical results of the work were implemented at the State Enterprise "Novator", "ITT – telecommunication company", "UCS++", CYPRESS SEMICONDUCTOR and the educational process of Khmelnytsky National University in the teaching of subjects "Security and Protection of Computer Systems", "Technical Diagnostics and Reliability of Computer Devices and Systems", "Parallel and Distributed Computing" and "System Software".

The first section analyzes malware development technologies, network detection systems, existing antivirus software and known methods. As a result of the analysis, it is concluded that in order to increase the reliability of malware detection, it is necessary to develop the theory and practice of creating distributed systems on the basis of decentralization and self-organization in computer systems of local networks.

The second section is a methodological basis for the creation of distributed malware detection systems in local computer systems on the basis of decentralization and self-organization with hardware and software protection of information as its component.

The methodological principles for the creation of distributed malware detection systems combine them with the integrated consideration of distribution, decentralization, multilevel and self-organization, which enables the autonomous functioning of both the system and its components, as well as the inclusion of hardware and software information security in its structure.

On the basis of the developed methodology it is proposed:

an improved model of the architecture of a distributed malware detection system in local computer networks differs from the well-known complex taking into account the requirements of distribution, decentralization, multilevel and self-organization, which enables the autonomous functioning of both the system and its components, as well as the inclusion of its hardware -program of information security;

a new model of the architecture of the typical components of a distributed multilevel malware detection system based on the Kripke structures that allows them to be considered in different states, which is the basis for determining the security status of the entire system and its components;

a new method for interacting components of a distributed multilevel system based on maintaining the integrity of the system and establishing the order of transfer of knowledge between components of the system and the use of the obtained analytical expressions for the levels of security of program modules and distributed multilevel system, which allows dynamically to implement a flexible change of the architecture of the system and influence its further actions without user intervention, and determine the further strategy of the system.

In the third section, formalization of the characteristic properties of malware with the use of algebra theory was made, on the basis of which malicious software behavioral algebras and types of threats were developed, which allowed to obtain:

improved malware models based on their presentation of behaviors algebra to create a basis for behavioral signatures, classification by type based on the consideration of the peculiarities of their functioning in local networks.

In the fourth section, the typical components of botnet networks (network malware) are selected based on the functions, the characteristics of the models of botnets and their components are made, on the basis of which it is proposed:

a new method for detecting botnets in computer systems of local networks for active monitoring of system events, the classification and coordinated interaction of components of a distributed system when making a decision to increase the authenticity of detection based on its integration into a distributed system and the implementation

of the classification of botnets by their behavioral signatures formed by functions embedded in their components.

In the fifth section, the formalization of file malware models was made, based on which it was proposed:

a new method for detecting file malware on local networks by integrating the detection method in individual computer systems, the method of detecting polymorphic and metamorphic code, a method for scanning executable programs by creating separate processes for them, and various software agents in a distributed system to increase the authenticity of detection by account of involvement in the process of identifying other components of the distributed system;

a new method for detecting file malware based on the dynamic formation of a behavioral signature by tracking applications of an application software interface in order to be able to detect other varieties, as well as new versions of the known, and the implementation of the formation of a behavioral signature on the critical calls of the application software interface in the groups of malicious activity with reflection the frequency of their occurrence and the nature of the interaction of critical functions;

a new method for detecting polymorphic and metamorphic viruses based on the analysis of the object object's obfuscation functions and its modified versions derived from various components of the distributed system, and further analysis based on the search for equivalent functional blocks for a more detailed analysis of the code of the software object for the presence of a polymorphic and metamorphic code.

In the sixth section the software of the distributed multilevel system and its hardware-software components of information protection for the implementation of the proposed theoretical foundations of such systems are developed. The confirmation of the possibility of their practical creation and use in experimental research for comparison with known systems of detection and implementation of the distributed distributed multilevel malware detection system in local networks in organizations and enterprises has been confirmed.

The conclusions formulate the main results of dissertation research. The **appendices** contain a description of the distributed multilevel malware detection

system, the results of the work, a description of the hardware and software device, a table of functions of transitions from state to state, a table of strategies for determining the next steps of the system, the researcher's works and the implementation of the results of work.

Key words: distributed systems, malicious software, information security tools, self-organized systems, decentralized systems, detection methods.

СПИСОК ПУБЛІКАЦІЙ

Наукові праці, в яких опубліковані основні наукові результати дисертації

1. Савенко О. С. Метод антивірусного діагностування персональних комп’ютерів на основі матриць інцидентності / О. С. Савенко, В. М. Джулій, В. М. Стецюк // Вісник Технологічного університету Поділля. Технічні науки. – 2000. – № 3. – С. 136–139. – Розроблено представлення файлового ЗПЗ на основі матриць інцидентності та метод виявлення файлового ЗПЗ.

2. Савенко О. С. Методика генерації матриць інцидентності для використання в антивірусних програмних засобах / О. С. Савенко // Вісник Технологічного університету Поділля. Технічні науки. – 2003. – № 3, т. 2. – С. 48–56.

3. Савенко О. С. Генерація моделей комп’ютерних вірусних програм в системі оцінки достовірності результатів роботи антивірусних засобів / О. С. Савенко, С. В. Мостовий // Вісник Хмельницького національного університету. Технічні науки. – 2005. – № 4, т. 1. – С. 198–200. – Розроблено систему представлення файлового ЗПЗ на основі матриць та механізм їх отримання.

4. Савенко О. С. Дослідження методів антивірусного діагностування комп’ютерних мереж / О. С. Савенко, С. М. Лисенко // Вісник Хмельницького національного університету. Технічні науки. – 2007. – № 2, т. 2. – С. 120–126. – Розроблено типову структуру мережного ЗПЗ.

5. Савенко О. С. Дослідження та аналіз блокування процесів в

комп'ютерній системі / О. С. Савенко, Ю. П. Кльоц, С. В. Мостовий // Вісник Хмельницького національного університету. Технічні науки. – 2007. – № 3, т. 1. – С. 248–251. – Здійснено аналіз та формалізовано представлення станів процесів.

6. Савенко О. С. Життєвий цикл процесів комп'ютерної системи / О. С. Савенко, С. В. Мостовий // Радіоелектронні і комп'ютерні системи. – 2010. –№ 7 (48). – С. 35–38. – Здійснено формалізоване представлення станів процесів.

7. Локазюк В. М. Модель прогнозування стану взаємоблокування процесів комп’ютерної системи / В. М. Локазюк, О. С. Савенко, С. В. Мостовий // Вісник Вінницького політехнічного інституту. – 2011. – № 4. – С. 130–133. – Розроблено представлення граничних станів процесів.

8. Савенко О. С. Діагностування комп’ютерних систем на наявність шкідливого програмного забезпечення на основі антивірусної мультиагентної системи / О. С. Савенко, С. М. Лисенко, А. Ф. Крищук // Вісник Національного університету «Львівська політехніка». – 2011. – № 5. – С. 99–105. – Розроблено базові характеристики компонентів розподілених систем виявлення ЗПЗ.

9. Савенко О. С. Адаптивна інформаційна технологія виявлення троянських програм в комп’ютерних системах / О. С. Савенко, С. М. Лисенко // Комп’ютинг. – 2011. – Т. 10, № 3. – С. 85–92. – Розроблено концепцію поділу файлового ЗПЗ на класи за їх поведінкою.

10. Берников А. Р. Поиск вредоносных программ в распределенных тренажерах с использованием технологии нечеткой логики / А. Р. Берников, Р. П. Графов, С. Н. Лысенко, О. С. Савенко // Информационные технологии. *(РИНЦ)* – 2011. – № 10. – С. 42–47. – Розроблено методологію вирішення проблеми виявлення файлового ЗПЗ у розподілених системах на прикладі розробки методу виявлення троянських програм.

11. Савенко О. С. Прогнозування потрапляння процесів у стан взаємоблокування / О. С. Савенко, С. В. Мостовий // Вісник Вінницького політехнічного інституту. – 2013. – № 2. – С. 81–86. – Здійснено визначення граничного стану для групи процесів, які наближаються до стану

взаємоблокування.

12. Нічепорук А. О. Моделі життєвого циклу поліморфних вірусів / А. О. Нічепорук, О. С. Савенко // Комп'ютерно-інтегровані технології: освіта, наука, виробництво. – 2013. – № 11. – С. 64–71. – Розроблено поділ файлового ЗПЗ на класи за їх поведінкою.

13. Савенко О. С. Моделі рівнів поліморфних комп'ютерних вірусів / О. С. Савенко, С. М. Лисенко, А. О. Нічепорук // Вісник Вінницького політехнічного інституту *(Index Copernicus)*. – 2015. – № 2. – С. 75–83. – Розроблено деталізоване множиною дій представлення файлового ЗПЗ, яке здійснює заплутування програмного коду, на основі застосування його моделей.

14. Савенко О. С. Програмне забезпечення інформаційної технології моделювання поширення вірусних кодів в гетерогенних мережах / О. С. Савенко // Вісник Хмельницького національного університету. Технічні науки *(Index Copernicus)*. – 2017. – № 1. – С. 144–148.

15. Савенко О. С. Распределенная многоуровневая сетевая система обнаружения метаморфных вирусов в локальных компьютерных сетях / О. С. Савенко // Вестник Брестского государственного технического университета (физика, математика, информатика). – 2017. – № 5 (107). – С. 40–44.

16. Савенко О. С. Критерії класифікації методів виявлення шкідливого програмного забезпечення / О. С. Савенко // Вісник Хмельницького національного університету. Технічні науки. – 2018. – № 1. – С. 23–27.

17. Савенко О. С. Модель та архітектура розподіленої багаторівневої системи виявлення шкідливого програмного забезпечення в локальних комп'ютерних мережах / О. С. Савенко // Вісник Хмельницького національного університету. Технічні науки. – 2018. – № 2. – С. 153–163.

18. Савенко О. С. Формалізація шкідливого програмного забезпечення в локальних комп'ютерних мережах / О. С. Савенко // Вісник Хмельницького національного університету. Технічні науки. – 2018. – № 3. – С. 145–154.

19. Савенко О. С. Архітектура багаторівневої програмної системи

виявлення шкідливого програмного забезпечення в локальних комп'ютерних мережах / О. С. Савенко, В. І. Грибинчук, М. О. Кульчицький // Комп'ютерно-інтегровані технології: освіта, наука, виробництво. – 2018. – № 30–31. – С. 132–140. – Розроблено архітектуру РБС та її представлення UML-діаграмами.

20. Савенко О. С. Архітектура розподіленої багаторівневої системи виявлення шкідливого програмного забезпечення в локальних комп'ютерних мережах / О. С. Савенко // Вчені записки Таврійського національного університету. Технічні науки. – 2018. – Т. 29 (68), № 2. – С. 172–181.

21. Савенко О. С. Формування сигнатури поведінки програми на основі трасування API викликів / О. С. Савенко, А. О. Нічепорук, А. А. Нічепорук, Ю. О. Нічепорук // Електротехнічні та комп'ютерні системи. – 2018. – № 29 (105). – С. 67–77. – Розроблено метод отримання поведінкової сигнатури на основі трасування API-викликів.

Праці, які засвідчують апробацію матеріалів дисертації

22. Savenko O. Multi-agent based approach of botnet detection in computer systems / O. Savenko, S. Lysenko, A. Kryschuk // Communications in Computer and Information Science, ISSN: 1865-0929 *(Scopus, Web of Science*). – 2012. – Vol. 291. – Pp. 171–180.

23. Pomorova O. Multi-Agent Based Approach for Botnet Detection in a Corporate Area Network Using Fuzzy Logic / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk // Communications in Computer and Information Science, ISSN: 1865–0929 *(Scopus, Web of Science)*. – 2013. – Vol. 370. – Pp. 146–156.

24. Pomorova O. A Technique for detection of bots which are using polymorphic code / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, A. Nicheporuk // Communications in Computer and Information Science, ISSN: 1865–0929 *(Scopus, Web of Science)*. – 2014. – Vol. 431. – Pp. 265–276.

25. Savenko O. Metamorphic Viruses' Detection Technique Based on the Equivalent Functional Block Search / O. Savenko, S. Lysenko, A. Nicheporuk, B. Savenko // CEUR-WS, ISSN: 1613–0073 *(Scopus)*. – 2017. – Vol. 1844. –

Pp. 555–569.

26. Lysenko S. Information technology for botnets detection based on their behaviour in the corporate area network / S. Lysenko, O. Savenko, K. Bobrovnikova, A. Kryshchuk, B. Savenko // Communications in Computer and Information Science, ISSN: 1865–0929 *(Scopus, Web of Science)*. – 2017. – Vol. 718. – Pp. 166–181.

27. Markowsky G. The technique for metamorphic viruses' detection based on its obfuscation features analysis / G. Markowsky, O. Savenko, S. Lysenko, A. Nicheporuk // CEUR-WS, ISSN: 1613–0073 *(Scopus)*. – 2018. – Vol. 2104. – Pp. 680–687.

28. Markowsky G. Distributed Malware Detection System Based on Decentralized Architecture in Local Area Networks / G. Markowsky, O. Savenko, A. Sachenko // Advances in Intelligent Systems and Computing, ISSN: 2194–5357 *(Scopus)*. – 2019. – Vol. 871. – Pp. 582–598.

29. Savenko O. The Technique for Computer Systems Trojan Diagnosis in the Monitor Mode / O. Savenko, S. Lysenko // Proceedings of the 6-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Prague (Czech Republic), September 15–17, 2011 *(Scopus)*. – Prague, 2011. – Pp. 770–774.

30. Savenko O. Botnet detection technique for corporate area network / O. Savenko, S. Lysenko, A. Kryshchuk, Y. Klots / Proceedings of the 7-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Berlin (Germany), September 12–14, 2013 *(Scopus)*. – Berlin, 2013. – Pp. 363–368.

31. Lysenko S. DNS-based Anti-evasion Technique for Botnets Detection / S. Lysenko, O. Pomorova, O. Savenko, A. Kryshchuk and K. Bobrovnikova // Proceedings of the 8-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Warsaw (Poland), September 24–26, 2015 *(Scopus, Web of Science)*. – Warsaw, 2015. – Pp. 453–458.

32. Pomorova O. A Technique for the Botnet Detection Based on DNS-Traffic Analysis / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, K. Bobrovnikova //

Proceedings of the 22-nd International Conference Computer Networks, Brunów (Poland), June 16–19, 2015 *(Scopus, Web of Science)*. – Brunów, 2015. – Vol. 522. – Pp. 127–138.

33. Pomorova O. Anti-evasion Technique for the Botnets Detection Based on the Passive DNS Monitoring and Active DNS Probing / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, K. Bobrovnikova // Proceedings of the 23-nd International Conference Computer Networks, Brunów (Poland), June 14–17, 2016 *(Scopus, Web of Science)*. – Brunów, 2016 – Vol. 608. – Pp. 83–95.

34. Savenko O. Approach for the Unknown Metamorphic Virus Detection / O. Savenko, S. Lysenko, A. Nicheporuk, B. Savenko // Proceedings of the 8-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Bucharest (Romania), September 21–23, 2017 *(Scopus, Web of Science)*. – Bucharest, 2017. – Pp. 71–76.

35. Markowsky G. Distributed System for Detecting the Malware in LAN / G. Markowsky, O. Savenko, A. Sachenko // Proceedings of the 2018 IEEE 13th International Scientific and Technical Conference on Computer Science and Information Technologies, CSIT'2018, Lviv (Ukraine), September 11–14, 2018 *(Scopus, Web of Science)*. – Lviv, 2018. – Pp. 306–309.

36. Графов Р. П. Забезпечення надійності розподільних мереж на основі динамічної структурної реконфігурації / Р. П. Графов, О. С. Савенко // Інформаційні технології та комп'ютерна інженерія. – 2005. – №3. – С. 241–246.

37. Савенко О. С. Метод виявлення бот-мереж розподіленими системами на основі самоорганізації / О. С. Савенко // Штучний інтелект. – 2018. – № 4 (82). – С. 58–72.

38. Savenko O. Intelligent method of the search of the trojan program in computer systems / O. Savenko, S. Lysenko // Праці IV міжнародної конференції «Сучасні комп'ютерні системи та мережі: розробка та використання», (ACSN'2009), (Львів, 17–19 грудня 2009). – Львів, 2009. – С. 154–158.

39. Savenko O. Software for computer systems trojans diagnosing as a safety-case tool / O. Savenko, S. Lysenko // Proceedings of the 1-st International Workshop

on Critical infrastructure safety and security, CrlSS-DESSERT'11, Kirovograd, May 11–13, 2011. – Kirovograd, 2011. – Pp. 353–361.

40. Савенко О. С. Дослідження антивірусних технологій діагностування комп'ютерних систем на наявність шкідливого програмного забезпечення / О. С. Савенко, С. М. Лисенко, А. Ф. Крищук // Праці XII міжнародної науково-практичної конференції «Сучасні інформаційні та електронні технології», (СІЕТ-2011), (Одеса, 27–31 травня 2011). – Одеса, 2011. – С. 165–166.

41. Savenko O. Interoperability of distributed multiple system for malware detection based on components levels of safety / O. Savenko // Проблеми інформаційних технологій. – 2018. – № 24. – С. 78–92.

42. Савенко О. С. Функційна модель бота як складової ботнет-мережі / О. С. Савенко, С. М. Лисенко, А. Ф. Крищук // Тези доповідей І Міжнародної науково-технічної конференції «Захист інформації і безпека інформаційних систем», (Львів, 2012). – Львів, 2012. – С. 123–124.

43. Савенко О. С. Розподілена апаратно-програмна система та методи захисту інформації в комп'ютерних системах локальних мереж / О. С. Савенко // Наукові праці Чорноморського національного університету ім. П. Могили. Комп'ютерні технології. – 2018. – Т. 320. Вип. 308. – С. 72–75.

44. Савенко О. С. Метод взаємодії компонентів розподіленої системи виявлення зловмисного програмного забезпечення в локальних обчислювальних мережах / О. С. Савенко, А. О. Нічепорук // Збірник тез доповідей XIV Міжнародної конференції «Контроль і управління в складних системах», (КУСС-2018), (Вінниця, 15–17 жовтня, 2018). – Вінниця, 2018. – С. 37.

45. Савенко О. С. Формалізоване структурування шкідливого програмного забезпечення на основі алгебраїчних систем / О. С. Савенко // Вимірювальна та обчислювальна техніка в технологічних процесах. – 2018. – №1. – С. 67–72.

46. Савенко О. С. Розподілена система виявлення зловмисного програмного забезпечення та метод взаємодії її компонент в локальних мережах / О. С. Савенко // Матеріали доповідей V Міжнародної науково-практичної конференції «Інформаційні технології та взаємодії», (Київ,

КНУ ім. Т. Шевченка, 20–21 листопада 2018). – Київ, 2018. – С. 308–309.

Публікації, які додатково відображають наукові результати дисертації

47. Пат. на корисну модель 108238 Україна, МПК G06F 21/55 Мультиагентний спосіб локалізації бот-мереж у корпоративних комп'ютерних мережах / О. В. Поморова, О. С. Савенко, А. Ф. Крищук, С. М. Лисенко, К. Ю. Бобровнікова, А. О. Нічепорук; заявник і патентовласник Хмельницький національний університет. – № u201600127; заявл. 04.01.2016; опубл. 11.07.2016, Бюл. № 13/2016.

48. Пат. на корисну модель 118456 Україна, МПК G06F 21/55 Спосіб виявлення метаморфних вірусів на основі статистичних метрик для визначення еквівалентних функціональних програмних блоків / О. С. Савенко, С. М. Лисенко, К. Ю. Бобровнікова, А. О. Нічепорук, Б. О. Савенко; заявник і патентовласник Хмельницький національний університет. – № u201701743; заявл. 23.02.2017; опубл. 10.08.2017, Бюл. № 15/2017.

49. Пат. на корисну модель 118663 Україна, МПК G06F 21/55 Спосіб ідентифікації бот-мереж у корпоративних комп'ютерних мережах на основі аналізу DNS-трафіку / О. С. Савенко, С. М. Лисенко, К. Ю. Бобровнікова, А. О. Нічепорук, Б. О. Савенко; заявник і патентовласник Хмельницький національний університет. – № u201612041; заявл. 28.11.2016; опубл. 28.08.2017, Бюл. № 16/2017.

50. А. с. 80223 Україна. Комп'ютерна програма пошуку та визначення еквівалентних функціональних блоків у виконуваних файлах для ідентифікації ознак метаморфних вірусів в локальних комп'ютерних мережах / А. О. Нічепорук, О. С. Савенко, С. М. Лисенко. 2018.

51. А. с. 80445 Україна. Розподілена комп'ютерна програма для виявлення зловмисного програмного забезпечення в локальних обчислювальних мережах на основі аналізу поведінкових сигнатур / О. С. Савенко. 2018.

ЗМІСТ

Стор.

АНОТАЦІЯ 2

ABSTRACT 9

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ……. 26

ВСТУП 27

РОЗДІЛ 1. СИСТЕМОЛОГІЧНИЙ АНАЛІЗ ПРОБЛЕМ СТВОРЕННЯ РОЗПОДІЛЕНИХ СИСТЕМ ВИЯВЛЕННЯ ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ЛОКАЛЬНИХ МЕРЕЖАХ..... 39

1.1. Технології розвитку зловмисного програмного забезпечення …… 39

1.2. Аналіз мережних систем виявлення зловмисного програмного забезпечення та достовірності результатів їх роботи.…….. 47

1.2.1. Аналіз існуючих мережних антивірусних засобів …… 47

1.2.2. Характеристика відомих систем виявлення зловмисного програмного забезпечення ……..…….. 56

1.3. Методи виявлення зловмисного програмного забезпечення та критерії їх класифікації …..…….. 59

1.3.1. Методи виявлення зловмисного програмного забезпечення…… 59

1.3.2. Критерії класифікації методів виявлення зловмисного програмного забезпечення ……..…….…… 70

1.4. Розподілені системи як засоби протидії зловмисному програмному забезпеченню……..…… 78

1.5. Визначення напрямів досліджень…….……. 84

Висновки до першого розділу…..……..……..… 87

РОЗДІЛ 2. РОЗПОДІЛЕНА БАГАТОРІВНЕВА СИСТЕМА ВИЯВЛЕННЯ ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ У ЛОКАЛЬНИХ КОМП’ЮТЕРНИХ МЕРЕЖАХ …..……..... 89

2.1. Модель та архітектура розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення ……..... 89
2.2. Модель архітектури програмних модулів РБС виявлення зловмисного програмного забезпечення...…….. 99
2.3. Модель розподіленої багаторівневої системи та принципи взаємодії програмних модулів різних комп'ютерних систем на основі децентралізації та самоорганізації…… 112
2.4. Метод взаємодії компонентів розподіленої багаторівневої системи виявлення ЗПЗ у локальних мережах…….. 124
Висновки до другого розділу..... 150
РОЗДІЛ 3. ТЕОРЕТИЧНІ І МЕТОДОЛОГІЧНІ ОСНОВИ ПОБУДОВИ СИСТЕМ ВИЯВЛЕННЯ ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ….. 153
3.1. Формалізація характерних властивостей зловмисного програмного забезпечення на основі теорії алгебр ……..… 153
3.2. Алгебри поведінки зловмисного програмного забезпечення …..... 172
3.3. Типи загроз та їх представлення алгебрами поведінки ……..……. 180
Висновки до третього розділу..... 184
РОЗДІЛ 4. МЕТОД ВИЯВЛЕННЯ МЕРЕЖНОГО ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ ……...…….. 186
4.1. Типові компоненти еталонної моделі бот-мереж на основі функцій..……... 187
4.2. Характеристики компонентів моделей бот-мереж та їх представлення ……..…….. 213
4.3. Метод виявлення бот-мереж у локальних комп'ютерних мережах… 229
Висновки до четвертого розділу 255

РОЗДІЛ 5. МЕТОДИ ВИЯВЛЕННЯ ФАЙЛОВОГО ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ……... 257
5.1. Моделі файлового зловмисного програмного забезпечення…….. 257
5.2. Методи виявлення файлового зловмисного програмного забезпечення……... 268
5.2.1. Метод виявлення файлового зловмисного програмного забезпечення на основі динамічного формування поведінкової сигнатури шляхом відстеження викликів API…… 268
5.2.2. Метод виявлення файлового ЗПЗ на основі аналізу функцій обфускації……... 282
5.3. Метод виявлення файлового ЗПЗ у локальних комп'ютерних мережах…. 287
Висновки до п'ятого розділу……. 291
РОЗДІЛ 6. СТРУКТУРА, ОЦІНКА ДОСТОВІРНОСТІ ТА ЕФЕКТИВНОСТІ РОЗПОДІЛЕНОЇ БАГАТОРІВНЕВОЇ СИСТЕМИ ВИЯВЛЕННЯ ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ …… 294
6.1. Програмна реалізація розподіленої системи виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах……. 294
6.2. Постановка і проведення експериментальних досліджень застосування розробленої РБС……. 307
6.3. Оцінки ефективності та достовірності мережних засобів виявлення зловмисного програмного забезпечення……. 320
Висновки до шостого розділу…….. 325
ВИСНОВКИ 327
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 331
ДОДАТКИ 380
Додаток А. Таблиці…… 380

Додаток Б. Програмне забезпечення РБС Distributed Multilevel System.. 394
Додаток В. Апаратно-програмний пристрій РБС……... 395
Додаток Г. Фрагмент результатів роботи РБС Distributed Multilevel System……. 405
Додаток Д. Список публікацій…….. 409
Додаток Е. Акти впровадження результатів дисертаційної роботи…..... 417

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API	-	Application programming interface
NP	-	Non-deterministic polynomial
АЗ	-	Антивірусні засоби
АПЗ	-	Антивірусні програмні засоби
АПМ	-	Автономний програмний модуль
ЖЦ	-	Життєвий цикл
ЗПЗ	-	Зловмисне програмне забезпечення
ЕФБ	-	Еквівалентні функціональні блоки
КС	-	Комп‘ютерна система
ЛКМ	-	Локальна комп‘ютерна мережа
ОЗП	-	Оперативний запам’ятовуючий пристрій
ОС	-	Операційна система
ПЗ	-	Програмне забезпечення
ПМ	-	Програмний модуль
РС	-	Розподілена система
РБС	-	Розподілена багаторівнева система
СКБД	-	Система керування базами даних
ФБ	-	Функціональні блоки

ВСТУП

Актуальність роботи. Тенденції розвитку технологій створення і поширення зловмисного програмного забезпечення (ЗПЗ) демонструють активне розширення технічних можливостей таких засобів [28, 291–302, 308, 309, 319, 394–396]. Сучасне зловмисне програмне забезпечення представляє собою складні багатофункційні програмні системи та комплекси, які побудовані з використанням ефективних методів створення програмних засобів та методів поширення зловмисного коду.

Виявлення зловмисного програмного забезпечення здійснюється за допомогою різноманітних засобів. Ефективність та достовірність виявлення суттєво залежать від архітектури таких засобів, а також їх позиціонування та місця розміщення в комп'ютерних системах (КС), зокрема, і локальних мережах. Дослідження відомих антивірусних методів та засобів вказують, що реалізація нових принципів, моделей та методів виявлення конкретних типів зловмисного програмного забезпечення шляхом створення відповідних систем потребує подальшого розвитку. Перспективним напрямом досліджень створення ефективних систем виявлення ЗПЗ є мережні системи, які використовуються в локальних комп'ютерних мережах організацій та підприємств. Такі засоби виявлення мають розподілену архітектуру і, тому, покращення ефективності виявлення ЗПЗ можливе за рахунок залучення груп КС локальної мережі.

Розвитком теоретичних основ та науково-практичного застосування розподілених систем різного призначення активно займаються Бернс Б. [306], Луцький Г. М. [184, 286], Марковський Г. [172, 173], Мельник А. О. [180–183, 340], Мухін В. Є. [184, 286–288, 398, 399].

Фахівцями, які проводять наукові дослідження в сфері виявлення ЗПЗ є Андерсон Б. [14–17], Вінод П. [262–264], Головко В. А. [109, 138, 139], Дудикевич В. Б. [316–318], Дейгон Д. [69, 70], Дітріх Д. [80–82], Дрозд О. В. [177, 250, 279, 338–339], Ентонакакіс М. [59–62], Елаєн К. [7–9], Ішайз Х. [118], Ісламі М. [89–91], Карпінський М. П. [397], Касперський Є. В.

[128, 129, 330], Котенко І. В. [41, 142–146, 307, 315, 332–336], Крістодонеску Д. [59–62], Лі З. [154–155], Максимович В. М. [318], Малан Д. Дж. [168, 169], Мартінсен Є. А. [176], Махавер Д. К. [166], Подловченко Р. І. [357], Рувінська В. М. [360], Погребенник В. Д. [354–356], Рад Б. Б. [207–209], Сочор Т. [241–244], Сцор П. [255, 256], Саченко А. О.[138, 139, 172, 173], Сиротинський О. І. [391], Тодерічі А. Г. [260], Собейкіс В. Г. [392], Целік З. [49, 50], Хольц Т. [115, 116], Хюнсанг Чоі [57–58] та ін.

У сучасних системах виявлення ЗПЗ застосовуються методи та засоби, що використовують знання про його функціонування та поведінку. Дослідження функціонування ЗПЗ у різних КС локальної мережі розподіленими системами виявлення дає змогу здійснити більш детальний аналіз програмних об’єктів та процесів на наявність ЗПЗ за рахунок залучення інших КС. Важливим при цьому є вибір архітектури таких розподілених систем, яка дозволить ефективно організувати процес виявлення у локальній комп’ютерній мережі порівняно з виявленням в окремій КС.

Тому, в роботі пропонується використання розподілених багаторівневих систем на основі децентралізації та самоорганізації для виявлення ЗПЗ у локальних комп’ютерних мережах. У такому випадку, залучення інших компонентів розподіленої системи до виявлення ЗПЗ в певній КС підвищує ймовірність його прояву і, відповідно, покращує ефективність його виявлення.

Зв’язок роботи з науковими програмами, планами, темами. Дослідження, представлені у дисертації, проводились в рамках держбюджетних НДР Хмельницького національного університету № 1Б-2001 «Методологія тестового комбінованого діагностування мікропроцесорних пристроїв та систем (МПП та С) на базі компонентів штучного інтелекту» (номер державної реєстрації 0101U005058), № 4Б-2012 «Розвиток теоретичних основ та розробка методів статико-динамічного спектрального оцінювання сигналів в радіолокації» (номер державної реєстрації 0112U002247), № 1Б-2018 «Розроблення високоефективних методів відбору енергії від фотоелектричних модулів» (номер державної реєстрації 0116U001548), № 1Б-2019 «Агентно-

орієнтована система підвищення безпеки та якості програмного забезпечення комп’ютерних систем» (номер державної реєстрації 0119U100662).

Мета і завдання дослідження. Метою дисертаційної роботи є покращення ефективності виявлення зловмисного програмного забезпечення шляхом розвитку теорії і практики створення розподілених систем у локальних комп’ютерних мережах на основі принципів децентралізації та самоорганізації.

Задачі дослідження формулюються в роботі наступним чином:

1) дослідити особливості функціонування зловмисного програмного забезпечення в локальних комп’ютерних мережах та проаналізувати ефективність роботи сучасних розподілених систем виявлення, а також їх архітектуру і компоненти;

2) удосконалити модель архітектури розподіленої системи виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах, в якій синтезувати вимоги розподіленості, децентралізованості, багаторівневості та самоорганізованості, для створення на її основі розподілених систем та їх компонентів, що функціонуватимуть автономно і самостійно прийматимуть рішення про наявність зловмисного програмного забезпечення та нарощення своїх функціональних можливостей;

3) розробити модель архітектури типових компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення на основі структур Кріпке з представленням компонентів через стани, в яких вони можуть перебувати під час функціонування, для визначення стану безпеки всієї розподіленої системи та її компонентів;

4) розробити метод взаємодії компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення для підтримки її цілісності, визначення порядку передачі знань між її компонентами і використання встановлених аналітичних залежностей між рівнями безпеки програмних модулів (ПМ) та рівнем безпеки всієї розподіленої багаторівневої системи (РБС), на основі якого система змогла б автономно змінювати свою архітектуру та функції без втручання користувача, а також визначати стратегію

своєї подальшої роботи;

5) удосконалити моделі зловмисного програмного забезпечення шляхом їх подання алгебрами поведінки, що дозволило б створити базис поведінкових сигнатур, врахувати особливості функціонування зловмисного програмного забезпечення в локальних комп'ютерних мережах та здійснити його класифікацію за типами поведінки;

6) розробити метод виявлення бот-мереж в локальних комп'ютерних мережах, який би включав здійснення активного моніторингу системних подій та організацію узгодженої взаємодії компонентів розподіленої системи при прийнятті рішень, для створення на його основі засобів, здатних інтегруватись в розподілену систему та класифікувати бот-мережі за їх поведінковими сигнатурами, сформованими закладеними в їх компоненти функціями;

7) розробити метод виявлення файлового зловмисного програмного забезпечення в локальних комп'ютерних мережах, суть якого полягає в поєднанні роботи програмних агентів, що здійснюють виявлення зловмисного програмного забезпечення в окремих комп'ютерних системах, відповідно до імплементованих в них методів: динамічного формування поведінкової сигнатури шляхом відстеження викликів прикладного програмного інтерфейсу, знаходження поліморфного та метаморфного програмного коду, сканування виконуваних програм шляхом створення для них автономних процесів та відповідних програмних агентів у розподіленій системі;

8) розробити метод виявлення файлового зловмисного програмного забезпечення на основі динамічного формування поведінкової сигнатури шляхом відстеження викликів прикладного програмного інтерфейсу, в якому поведінкову сигнатуру формувати на основі критичних викликів прикладного програмного інтерфейсу за групами зловмисної активності з відображенням частоти їх входження та характеру взаємодії критичних функцій, для виявлення нових версій відомого зловмисного програмного забезпечення не тільки за наявністю критичних викликів, але й за їх взаємодією між собою;

9) розробити метод виявлення поліморфних та метаморфних вірусів з

використанням функцій заплутування програмного коду на основі поетапного аналізу і порівняння функціональних блоків програмного об’єкта та його змінених версій, отриманих, в тому числі, від різних компонентів розподіленої системи шляхом їх взаємодії між собою;

10) розробити програмне забезпечення розподіленої багаторівневої системи та її апаратно-програмні компоненти захисту інформації для реалізації запропонованих теоретичних основ таких систем, підтвердження можливості їх практичного створення та використання в експериментальних дослідженнях для порівняння з відомими системами виявлення і впровадити розроблену розподілену багаторівневу систему виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах організацій (підприємств).

Об’єкт дослідження – процес виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах.

Предмет дослідження – методи і засоби виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах.

Методи дослідження. Для розв’язання поставлених задач використовуються основні положення:

1) теорії розподілених систем для побудови моделей архітектури розподіленої системи та її компонентів, на основі яких можуть бути розроблені програмні та апаратно-програмні засоби;

2) теорії абстрактної алгебри для представлення зловмисного програмного забезпечення, як елементів підмножин із заданими характеристичними властивостями, у вигляді алгебр поведінки;

3) теорії множин і теорії графів для деталізованого подання зловмисного програмного забезпечення, як об’єктів дослідження;

4) методи класифікації для здійснення віднесення програмних об’єктів до класів зловмисного програмного забезпечення;

5) теорії комп’ютерних мереж для представлення і організації функціонування розподіленої системи, зокрема її програмних та апаратно-програмних компонентів.

Наукова новизна одержаних результатів полягає в наступному:

1) удосконалено модель архітектури розподіленої системи виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах, яка відрізняється від відомих комплексним врахуванням вимог розподіленості, децентралізованості, багаторівневості та самоорганізованості, що дозволяє створювати на її основі розподілені системи та їх компоненти, які функціонуватимуть автономно і самостійно прийматимуть рішення про наявність зловмисного програмного забезпечення та нарощення своїх функціональних можливостей;

2) вперше розроблено модель архітектури типових компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення на основі структур Кріпке з представленням компонентів через стани, в яких вони можуть перебувати під час функціонування, що дає змогу враховувати перебування їх в різних станах і є основою для визначення стану безпеки всієї розподіленої системи та її компонентів;

3) вперше розроблено метод взаємодії компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення на основі підтримки її цілісності та визначення порядку передачі знань між її компонентами і використання встановлених аналітичних залежностей між рівнями безпеки програмних модулів та рівнем безпеки всієї розподіленої багаторівневої системи, що дозволяє системі автономно змінювати свою архітектуру та функції без втручання користувача, а також визначати стратегію своєї подальшої роботи;

4) удосконалено моделі зловмисного програмного забезпечення шляхом їх подання алгебрами поведінки, що дозволило створити базис поведінкових сигнатур, і, на відміну від відомих представлень, врахувати особливості функціонування зловмисного програмного забезпечення в локальних комп'ютерних мережах та здійснити його класифікацію за типами поведінки;

5) вперше розроблено метод виявлення бот-мереж у локальних комп'ютерних мережах, який базується на здійсненні активного моніторингу

системних подій та узгодженій взаємодії компонентів розподіленої системи при прийнятті рішення, і, на відміну від відомих методів, дає можливість створення на його основі засобів, здатних інтегруватись в розподілену систему та класифікувати бот-мережі за їх поведінковими сигнатурами, що формуються закладеними в їх компоненти функціями;

6) вперше розроблено метод виявлення файлового зловмисного програмного забезпечення в локальних комп'ютерних мережах, який полягає в поєднанні роботи програмних агентів, що здійснюють виявлення зловмисного програмного забезпечення в окремих комп'ютерних системах, відповідно до імплементованих в них методів: динамічного формування поведінкової сигнатури шляхом відстеження викликів прикладного програмного інтерфейсу, знаходження поліморфного та метаморфного програмного коду, сканування виконуваних програм шляхом створення для них автономних процесів та відповідних програмних агентів у розподіленій системі, що, на відміну від аналогів, дозволяє покращити аналіз і підвищити достовірність виявлення зловмисного програмного забезпечення;

7) вперше розроблено метод виявлення файлового зловмисного програмного забезпечення на основі динамічного формування поведінкової сигнатури шляхом відстеження викликів прикладного програмного інтерфейсу, в якому, на відміну від відомих методів, поведінкова сигнатура формується на основі критичних викликів прикладного програмного інтерфейсу за групами зловмисної активності та відображає частоту їх входження і характер взаємодії критичних функцій, що дає змогу виявляти нові версії відомого зловмисного програмного забезпечення не тільки за наявністю критичних викликів, але й за їх взаємодією між собою;

8) вперше розроблено метод виявлення поліморфних та метаморфних вірусів з використанням функцій заплутування програмного коду, відмінністю якого є поетапний аналіз і порівняння функціональних блоків програмного об'єкта та його змінених версій, отриманих, в тому числі, від різних компонентів розподіленої системи шляхом їх взаємодії між собою.

Обґрунтованість і достовірність наукових положень, висновків і рекомендацій. Наукові положення, висновки і рекомендації дисертації обґрунтовані коректним та доцільним використанням математичного апарату, алгоритмами здійснення виявлення, успішною програмною реалізацією розробленої розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах, ефективним практичним впровадженням результатів дисертаційного дослідження на підприємствах, що експлуатують комп'ютерні системи, яке продемонструвало відповідність теоретичних досліджень з реальними результатами застосування.

Практичне значення одержаних результатів. У результаті виконаного дисертаційного дослідження розроблено програмне забезпечення розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення в комп'ютерних системах локальних мереж та апаратно-програмні засоби захисту інформації як її компоненти, використання яких регламентується вимогами безпеки. Результати експериментальних досліджень з використанням розробленого програмного забезпечення підтверджують правильність наукових положень теорії розподілених систем, оскільки впровадження розподіленої багаторівневої системи виявлення ЗПЗ дозволяє підвищити достовірність виявлення на 5–12 % порівняно з відомими мережними антивірусними програмними засобами та знизити рівень помилок першого роду до 5 %.

Теоретичні та практичні результати роботи впроваджено в Державному підприємстві «Новатор» (м. Хмельницький, акт про впровадження від 29.03.2019 р., відділ автоматизованих систем управління), ТОВ «ІТТ – telecommunication company» (м. Хмельницький, акт про впровадження від 21.02.2019 р.), ТОВ «ЮКС++» (м. Хмельницький, акт про впровадження від 26.02.2019 р.), компанії CYPRESS SEMICONDUCTOR (м. Львів, акт про впровадження від 28.02.2019 р.) та навчальному процесі Хмельницького національного університету при викладанні дисциплін «Безпека та захист комп'ютерних систем», «Технічна діагностика і надійність комп'ютерних пристроїв

та систем», «Паралельні та розподілені обчислення» та «Системне програмне забезпечення» (акт про впровадження від 05.04.2019 р.).

Особистий внесок здобувача. Всі основні результати дисертаційного дослідження, які представлені до захисту, одержані автором особисто. В роботах, опублікованих одноосібно автором, отримано наступні результати: [385] – розроблено модель розподіленої системи на основі синтезу багаторівневої, децентралізованої та самоорганізованої архітектури та реалізовано метод виявлення метаморфних вірусів; [377] – розроблено представлення файлового зловмисного програмного забезпечення матрицями, в яких поєднано середовища можливого перебування ЗПЗ та шляхи його поширення; [384] – розроблено систему оцінки безпеки початкового стану комп'ютерних систем у локальній мережі залежно від використовуваного системного програмного забезпечення; [372] – систематизовано критерії класифікації методів виявлення файлового ЗПЗ та здійснено їх класифікацію, виділено типові недоліки класів методів; [380, 381] – запропоновано модель розподіленої багаторівневої системи виявлення ЗПЗ та формалізовано її типову архітектуру; [387, 388] – розроблено алгебраїчні системи та алгебри поведінки для представлення об'єктів класів ЗПЗ; [364] – розроблено модель архітектури компонент РБС на основі структур Кріпке та метод їх взаємодії; [379] – розроблено метод виявлення бот-мереж самоорганізованими розподіленими системами; [365, 386] – розроблено метод взаємодії компонентів РБС; [304] – розроблено програмне забезпечення РБС.

У роботах, опублікованих у співавторстві, автору належать основні ідеї, теоретична та практична розробка положень, відображених у характеристиці наукової новизни отриманих результатів, а саме: [227, 202, 223, 368] – розроблено методологію використання агентних систем при здійсненні виявлення бот-мереж у комп'ютерних мережах, а також принципи комунікації між агентами; [200] – розроблено метод виявлення бот-мереж, які містять поліморфний програмний код; [226] – запропоновано методологію отримання різних зразків ЗПЗ із залученням різних КС мережі для отримання його

функціональних блоків; [164] – розроблено метод виявлення бот-мереж на основі врахування його поведінки в мережі та можливої присутності файлового ЗПЗ; [174] – розроблено метод виявлення файлового ЗПЗ, яке використовує техніки заплутування програмного коду, та здійснено вибір метрик для порівняння функціональних блоків; [172, 173] – розроблено модель та архітектуру РБС і її програмні модулі; [305] – розроблено методологію вирішення проблеми виявлення файлового ЗПЗ розподіленими системами; [373, 366] – розроблено систему представлення файлового ЗПЗ на основі матриць та механізм їх отримання; [313] – розроблено концепцію та механізми організації розподілених систем; [369, 390] – розроблено типову структуру мережного ЗПЗ; [370, 371] – здійснено аналіз та формалізовано представлення станів процесів; [337] – розроблено представлення граничних станів процесів та структуру сигнатури процесу; [367] – розроблено базові характеристики компонентів розподілених систем виявлення ЗПЗ; [362, 348, 225, 228] – розроблено концепцію поділу файлового ЗПЗ на класи за їх поведінкою; [383] – здійснено визначення граничного стану для групи процесів, які наближаються до стану взаємоблокування; [378] – розроблено деталізоване множиною дій представлення файлового ЗПЗ, яке здійснює заплутування програмного коду, на основі застосування його моделей; [363] – розроблено архітектуру РБС та її представлення UML-діаграмами; [389] – розроблено метод отримання поведінкової сигнатури на основі трасування API-викликів; [350-352, 163, 201, 203] – розроблено методологію виявлення ЗПЗ у корпоративних мережах; [303] – розроблено модуль пошуку функціональних блоків у виконуваних програмах; [229] – розроблено матричне представлення класів ЗПЗ; [224] – розроблено формальне представлення функціональних блоків виконуваних програм та деталізацію зловмисних дій критичними API-функціями; [374] – розроблено метод взаємодії компонентів РБС.

Апробація результатів дисертації. Основні положення та результати проведених у дисертаційній роботі досліджень доповідалися та обговорювалися на 45 міжнародних та всеукраїнських конференціях, а саме: 6-th, 7-th, 8-th, 9-th

IEEE Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications IDAACS'2011 (Prague, Chech Republic, 2011), IDAACS'2013 (Berlin, Germany, 2013), IDAACS'2015 (Warszawa, Poland, 2015), IDAACS'2017 (Bucharest, Romania, 2017); 19-th, 20-th, 21-th, 22-th, 23-th, 24-th, 25-th International Conference on Computer Networks: CN'2012 (Szczyrk, Poland, 2012), CN'2013 (Lwówek Slaski, Poland, 2013), CN'2014 (Brunów, Poland, 2014), CN'2015 (Brunów, Poland, 2015), CN'2016 (Brunów, Poland, 2016), CN'2017 (Brunów, Poland, 2017), CN'2018 (Gliwice, Poland, 2018); 4-th, 5-th, 7-th, 13-th International Scientific and Technical Conference on Computer Science and Information Technologies: CSIT (Lviv, Ukraine, 2007, 2008, 2012, 2018); International Conference Advanced Computer Systems and Networks: Design and Application ACSN (Lviv, Ukraine, 2009, 2011); 1-st International Academic Conference on Science and Education in Australia, America and Eurasia: Fundamental and Applied Science (Melbourne, Australia, 2014); Міжнародній науково-практичній конференції «Сучасні інформаційні і електронні технології (CIET)» (Одеса, 2009, 2010, 2011, 2012, 2013); Проблемно-науково-міжгалузевій конференції «Інформаційні проблеми комп'ютерних систем, юриспруденції, енергетики, економіки, моделювання та управління (ПНМК)» (Бучач – Яремча, 2011, 2014); Міжнародній конференції «Контроль і управління в складних системах (КУСС)» (Вінниця, 2010, 2012, 2014, 2018); Міжнародній науково-практичній конференції «Комп'ютерні системи в автоматизації виробничих процесів» (Хмельницький – Головченці, 2002, 2003, 2004, 2005, 2007); Міжнародній науково-технічній конференції «Захист інформації і безпека інформаційних систем» (Львів, 2012); Міжнародному науково-практичному семінарі молодих вчених та студентів «Програмовані логічні інтегральні схеми та мікропроцесорна техніка в освіті і виробництві» (Луцьк, 2018); V Міжнародній науково-практичній конференції «Інформаційні технології та взаємодії» (Київ, 2018); Міжвузівській науково-практичній конференції «Прогресивні інформаційні технології в науці та освіті» (Вінниця, 2007); 1-st International Workshop «Critical infrastructure safety and security (CrlSS-

DESSERT'11)» (Kirovograd, 2011); Міжнародній конференції «Інтелектуальний аналіз інформації (IAI)» (Київ, 2008, 2010, 2013); Міжнародній науково-технічній конференції «Системний аналіз та інформаційні технології (CAIT)» (Київ, 2008, 2009, 2012); щорічних наукових конференціях професорсько-викладацького складу Хмельницького національного університету.

Публікації. За результатами проведених досліджень основні наукові результати опубліковано у 21 науковій праці [305, 337, 348, 362–364, 366, 367, 369–373, 377, 378, 380, 383–385, 387, 389], з яких 2 статті – у періодичних зарубіжних виданнях [305, 385] і 3 статті індексовані у наукометричних базах [305, 378, 384], 19 статей у фахових наукових виданнях України [337, 348, 362–364, 366, 367, 369–373, 377, 378, 380, 383–384, 387, 389]. Апробація засвідчена публікаціями 7 статей у періодичних зарубіжних серійних виданнях [164, 172, 174, 200, 202, 226, 227] і 7 праць в матеріалах зарубіжних та українських конференцій [163, 173, 201, 203, 223, 224, 229], індексованих у наукометричній базі Scopus, з яких 9 індексовані у наукометричній базі Web of Science, 11 статей та тез доповідей у журналах та збірниках праць конференцій [225, 228, 313, 365, 368, 374, 379, 381, 386, 388, 390]. Опубліковано 3 патенти на корисну модель [350–352] та 2 свідоцтва про реєстрацію авторського права на твір (програму) [303, 304].

Структура дисертації. Дисертація складається з анотації, вступу, шести розділів, висновків, списку використаних джерел з 399 найменувань на 49 сторінках та шести додатків на 46 сторінках. Загальний обсяг дисертації становить 425 сторінок, з них 304 сторінки основного тексту, 54 рисунки, 46 таблиць.

РОЗДІЛ 1
СИСТЕМОЛОГІЧНИЙ АНАЛІЗ ПРОБЛЕМ СТВОРЕННЯ РОЗПОДІЛЕНИХ СИСТЕМ ВИЯВЛЕННЯ ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ЛОКАЛЬНИХ МЕРЕЖАХ

Використання зловмисного програмного забезпечення щоденно зростає, створюючи при цьому проблеми користувачам комп'ютерних систем. Новими сферами його застосування для отримання вигод і переваг [291–302, 327], враховуючи попередню фінансову, стали військова і політична. Військові доктрини багатьох країн світу включають розвиток військових кіберпідрозділів, які розвивають і використовують ЗПЗ для нанесення матеріальної шкоди інфраструктурі інших країн. Застосування руйнуючих технологій ЗПЗ [310] дозволяє здійснювати віддалені атаки, не потребуючи перебувати поряд з об'єктом атаки. Особливої уваги заслуговує захист від таких атак підприємств і організацій реального сектору економіки, атаки на які протягом останніх років суттєво зросли і завдають їм значних збитків. Проведення успішних атак в різних секторах країни може паралізувати на тривалий час її основну інфраструктуру та фінансовий сектор. Крім того, поява нових фінансових інструментів, зокрема альтернативних валют, мотивує зловмисників до подальшого технологічного розвитку ЗПЗ для отримання вигоди. При цьому вони залучають обчислювальні потужності не тільки власні, а й інших користувачів, комп'ютерні системи яких приєднані до глобальної мережі. Тому, своєчасне виявлення ЗПЗ для організацій та підприємств залишається актуальною проблемою.

1.1. Технології розвитку зловмисного програмного забезпечення

Кількість користувачів мережі Internet стрімко зростає, що розширює можливості зловмисникам для отримання вигоди за рахунок можливості застосування ЗПЗ до більшої кількості комп'ютерних систем. Для організації зловмисних дій застосовуються ті засоби, які першочергово створювались, як

засоби підтримки роботи мереж. Також, стрімко розвиваючі технології програмування розширюють можливості зловмисників вирішувати багато зловмисних задач віддалено, маючи ефективні програмні засоби. Важливим елементом при використанні ЗПЗ є можливість його поширення за рахунок закладеного в нього ефективного інструментарію. Розвиток сучасного ЗПЗ здійснюється в напрямку створення технологій його поширення в мережі.

Для вирішення проблем користувачів, пов’язаних з реальними загрозами поширення ЗПЗ, на сьогодні розроблено і використовується багато ефективних антивірусних засобів. Крім того, враховуючи тривалий час неможливості подолання цієї проблеми, в розвинутих країнах світу унормували законодавчо технічні аспекти. Це дозволило розвивати не тільки розробку антивірусних засобів приватними компаніями, але і створювати ефективні організаційні рішення, зокрема стандартизацію підходів до політик безпеки організацій та підприємств. Також, такі заходи стали основою розвитку наукових досліджень в цій сфері. Розглянемо їх основні термінологічні поняття. Основним об’єктом для зловмисників є інформація, для якої необхідно забезпечити при її зберіганні цілісність, доступність і конфіденційність, що визначається стандартами безпеки. В [119–121, 311, 312, 321–326, 328, 329, 359] визначено термінологію із захисту інформації в комп’ютерних системах і мережах та порядок роботи з інформацією в автоматизованих системах. Основні терміни і поняття, які використовуються при дослідженні ЗПЗ та засобів його виявлення [84, 119–121, 341–346, 349]: атака, атака типу «replay» (атака перехоплення і повтору), вірус (комп'ютерний вірус), мережний вірус, мережний черв'як, детектор (вірусів), брандмауер (firewall), монітор, ревізор, сканер, сигнатура, електронний підпис, контрольний код цілісності (CRC), поведінкова сигнатура, файлова сигнатура, пермутації (від англ. permutate), поліморфний вірус, олігоморфний вірус, метаморфний вірус, приховане адміністрування системи, люк або «чорний хід» (backdoor), DoS-атака (Denial of Service), DDoS-атака (Distributed Denial of Service), скрипт (від англ. Script), скрипт-віруси, стелс-вірус (від англ. Stelth), троянська програма, експлойт (від англ. Exploit – використовувати), вразливість

(vulnerability), емуляція (процесора), симбіотичні програми (від грец. Symbiosis - взаємовигідне співіснування), приманки, приховані канали (covert channels), приховані канали за часом, приховані канали по пам'яті, словникова атака, система виявлення вторгнень, помилка 1-го роду (для антивіруса), помилка 2-го роду (для антивіруса), помилка 3-го роду (для антивіруса) [310], статистичні методи, евристичний аналіз.

Введені поняття і терміни є основою для представлення зловмисного програмного забезпечення, аналізу можливостей існуючих антивірусних засобів та відомих методів виявлення, а також побудови розподіленої системи виявлення ЗПЗ у локальних комп'ютерних мережах.

Основними інструментами поширення ЗПЗ в мережах є стандартні утиліти для роботи в мережах, які надаються з операційними системами. Зловмисники створюють на їх основі, використовуючи зростаючі можливості сучасних систем програмування, ефективні компоненти, модулі і автономні програмні одиниці, які володіють засобами поширення в мережах.

Основними засобами, які використовуються для поширення ЗПЗ [5, 40, 42, 60, 73, 77, 115, 124, 167, 230, 233, 248, 310], є утиліти консольних налагоджувальників в операційних системах, конфігурування мережних інтерфейсів, отримання різної мережної інформації про функціонування мережі, перерахування всіх відкритих файлів, виведення часу виконання програми, профілювання, пошук функцій в файлах, відслідковування всіх викликів заданих програм, відслідковування всіх викликів програм до динамічних бібліотек, дослідження пам'яті, для вірної збірки змінених файлів, виведення всіх динамічних бібліотек для заданих програм, дослідження структури файлів, виведення розмірів секцій програми, видалення таблиці символів з програми, класифікації файлів, виведення сумісно використовуваних сегментів деяких процесів, створення статичних бібліотек, здійснення архівації, отримання доступу до системного кешу, мережний аналізатор пакетів.

Розглянемо створене на їх основі відоме мережне ЗПЗ, зокрема бот-мережі [3, 7, 27, 31, 52, 67, 71, 228]. Для зловмисників одним з основних і стабільних

джерел доходів залишаються розробка і використання бот-мереж. Інфікування КС бот-мережами щорічно зростає [258, 265]. Відсутність необхідності у високому рівні знань для створення та керування бот-мережами, а також незначні фінансові витрати для їх створення та адміністрування, призводять до зростання кількості зловмисних дій, здійснених саме із їх застосуванням [294-296, 397, 398]. Зростання кількості КС підключених до мережі Internet та засобів віддаленого доступу в комп'ютерних мережах в 1990-х роках сформували основу для створення можливостей здійснення розподілених атак на відмову в обслуговуванні [165, 333, 391]. Вперше цілісне мережне ЗПЗ з множиною функцій віддаленого керування інфікованою КС було розроблено в 1999 р. [66, 71–73]. Для нього введено термін «бот-мережа», що означає розподілену програмну систему для контролю над комп'ютерними системами, які були інфікованими ЗПЗ «backdoor». Зловмисник завдяки бот-мережам отримував можливості віддалено керувати інфікованими КС, встановлювати на них необхідні задачі для виконання та використовувати їх ресурси на власні потреби. Бот-мережі можуть включати засоби, які реалізуються в файловому ЗПЗ [75, 200], троянських програмах [362], руткітах [310, 392] і worm-вірусах [72, 204, 205, 310]. Ці засоби необхідні для пришвидшення отримання доступу над атакованою КС. Отриманий доступ над двома і більше КС дозволяє створити бот-мережу. Між вузлами бот-мережі може бути встановлений зв'язок і може бути, що такого зв'язку немає. Це пояснюється різною архітектурою таких бот-мереж, яка визначається зловмисником. Для уникнення виявлення бот-мереж та їх вузлів зловмисники використовують різні програмні технології, особливості побудови комп'ютерних мереж, різні архітектури розподілених систем та додаткові зловмисні засоби і технології, зокрема файлового ЗПЗ як компонентів для вирішення локальних задач. Беручи до уваги дані розробника антивірусних засобів McAfee, бот-мережі сімейств Agobot, Sdbot і DSNXbot, а також evilbot і Spybot разом складають 98,2% відомих різновидів бот-мереж [72, 73]. Протокол IRC є технологією для управління бот-мережею і дозволяє реалізувати її централізовану модель взаємодії. Основні технології розподілених систем бот-

мереж виникли з принципів функціонування Internet Relay Chat (IRC) [108, 198, 230].

Різна архітектура бот-мереж вплинула на поділ їх на такі класи: ago-bot, DSNX-bot, evil-bot, G-SYS-bot, SD-bot, SPY-bot [97]. Інша класифікація, сформована дослідниками з проекту Honeynet, поділяє їх на такі чотири типи: ago-bot, SD-bot, DSNX-bot, бот-мережі MIrc [71, 72, 115]. Інші класифікації базуються на функціональності бот-мереж та їх вузлів або технологій приховування своєї присутності і свого коду [110, 129]. Проаналізуємо особливості кожного типу бот-мереж. Ці особливості є необхідними для чіткого відокремлення класів між собою з метою подальшого представлення типових класів, як об’єктів виявлення.

Типова бот-мережа ago-bot [97] має кросплатформену основу, реалізована засобами C/C++, використовує IRC-канали для керування, модульність, стандартні і спеціальні команди IRC для приховування конфіденційної інформації [108], доступ до файлів на віддаленому комп'ютері через відповідні директиви, систему захисту за допомогою закриття NetBIOS ресурсів, змогу запускати різні DoS-атаки, засоби атакувати велику кількість цілей, функції кодування оболонки, засоби збирання інформації через фільтрування трафіку [140, 147, 163], засоби здійснення клавіатурного шпіонажу або пошуку записів в реєстрі, засоби ухилення від виявлення антивірусним ПЗ, засоби закриття бекдорів, засоби відключення доступу до антивірусних сайтів, змогу перевіряти середовище виконання, зокрема відлагоджувальників чи віртуальних машин для уникнення дизассемблювання. При здійсненні пошуку нової КС типовим для класу є сканування певного діапазону адрес в мережі.

Типовий клас SD-bot бот-мережі має такі характеристики: реалізований в середовищі програмування C, невеликий програмний код (до 2500 рядків), набір команд аналогічний типовій бот-мережі ago-bot [97], свої функції IRC для здійснення шпигунства і клонування. SD-bot надає функції управління інфікованим «хостам», але не здійснює поширення. SD-bot вважають компактною реалізацією IRC. Пошук нових КС для інфікування здійснює за

допомогою багатьох інструментів сканування. Елементи цього класу завжди використовуються для простих «flooding» атак та DDoS-атак.

Типова бот-мережа SPY-bot характеризується такими особливостями: реалізована в середовищі програмування C, має невеликий програмний код (до 3000 рядків) та багато модифікацій [97], є розширенням типового класу SD-bot, має видозмінену командну мову, містить сканування, функції керування «хостом», модулі DDoS-атак і «flooding»-атак. Керування вузлом бот-мережі реалізовано аналогічно до типовго класу ago-bot, але на відміну від бот-мережі ago-bot, Spybot не має модульності і відповідно переваг, які нею досягаються.

Бот-мережі MIrc є багатоваріантними [28, 97, 171] та характеризуються такими особливостями: керування IRC «хостом», здійснення DoS-атак, сканування портів, NetBIOS/RPC виконання, надання обмеженого набору бінарних файлів і скриптів, використання бінарних файлів програм для забезпечення невидимості від користувачів.

Бот-мережі можуть бути побудовані на основі різних архітектур та в декілька рівнів. Модель побудови бот-мережі може бути ієрархічною, клієнт-серверною, частково централізованою, децентралізованою та інші [128, 223, 367]. Кожна з цих моделей має переваги та недоліки. Для досягнення швидкодії використвують централізовано-ієрархічну модель, але вона є уразливою у випадку виявлення центру. Якщо використовується децентралізована модель, тоді виникають проблеми з швидкодією при керуванні її вузлами, бо частина КС може бути вимкнена. Можливе також використання гібридної архітектури. Але для зловмисника найкращою за часом обслуговування є централізована модель. Всі інші архітектури бот-мереж потребують більших витрат часу для обслуговування та документування подій.

Основні способи розбудови бот-мереж та поширення їх компонентів такі [71–73]: засоби електронної пошти, методи соціальної інженерії, миттєві повідомлення, використання вразливостей систем, засобами worm-вірусів, використання віддалених вразливостей, відвідування веб-сторінок, за запитами документів при їх пошуку, приховування під корисне ПЗ.

Засоби захисту бот-мереж та їх вузлів від виявлення такі [55, 79, 92, 194, 196]: аналіз середовища виконання, застосування технік заплутування програмного коду, використання руткіт-технологій, зараження головного завантажувального запису, зберігання модулів бот-мереж не в таблиці розділів, організація шифрованого віртуального диску, маскування під системний процес, виконання поряд із запущеними процесами, використання декількох самоперезапускаючих процесів, заміна системних файлів, перезавантаження КС при спробах доступу до ЗПЗ, використання нестандартних методів запуску, блокування фаєрволів, завершення процесів АПЗ тощо.

Бот-мережі для зв'язку з своїми компонентами вузлами в КС використовують такі мережні протоколи [189, 239, 369]: IRC (Internet Relay Chat), IM (канали служб Instant Messaging, зокрема, AOL, MSN, ICQ), веб (www), власний протокол (на основі протоколів TCP/IP).

Технічні засоби комп'ютерних мереж надають можливість зловмисникам покращити анонімність в роботі та надійність зв'язку між вузлами бот-мережі за рахунок використання сервісів динамічної системи Domain Name System (DNS) або власних розподілених DNS-сервісів [32, 93, 95, 106, 261, 352]. DNS, яка є всесвітньою ієрархічною розподіленою комп'ютерною системою, використовується для пошуку та ідентифікації доменів в мережі Інтернет. Про можливості DNS [179, 187, 203] подано в RFC-1034 та RFC-1035.

При створенні та адмініструванні бот-мереж використовуються складні технічні методи: швидкий потік обслуговування мереж (FFSN), прихована передача всієї комунікації клієнта до сервера за допомогою мережі проксі-серверів, динамічна зміна C&C-інфраструктури, методи генерування домену (DGA) [73, 248], використання множини серверів, створення прихованого каналу в межах інших спільних доступних інтернет-технологій та послуг, використання піддомену третього рівня від DDNS-провайдера (Dynamic Domain Name System) [230], методи зворотніх DNS-запитів для отримання додаткових доменних імен C&C-сервера, методи DNS-тунелювання (DNS-tunneling), метод «швидкозмінні» мережі, технології domain flux та періодичну зміну IP-

відображення для зловмисного домена [49, 214], методи базовані на періодичній зміні IP-відображення для доменного імені C&C-сервера та встановлення для нього коротких TTL-періодів [186], методи регулярної зміни локації C&C-сервера, методи перенесення адреси зловмисного домену на певний час на приватну адресу RFC 1918 [284, 289], методи «потік доменів» [276] поєднують короткі TTL-періоди та часті зміни доменного імені C&C-сервера [70], методи ухилення від занесення до «чорних списків» DNS (DNSBL [230], DNS blacklist) поєднанням коротких TTL-періодів та циклічного методу розподілення навантаження round-robin DNS, метод DNS-тунелювання [93].

Крім мережного ЗПЗ, розглянемо, також, файлове ЗПЗ, оскільки частина з нього може бути основою для мережного, а інша частина, яка є самостійним ЗПЗ при ураженні багатьох КС локальної мережі може мати однакову поведінку в різних КС і, також, створювати проблеми користувачам. Файлове ЗПЗ на відміну від бот-мереж здатне переважно до самокопіювання і поширення в файлах.

Комп’ютерні віруси переважно пов’язані з типом операційної системи, бо для свого поширення і зберігання втілюються у виконувані програми. На їх поширення впливає те, на скільки правильно в операційній системі вибудована система розмежування прав доступу. Зокрема, в ОС типу Linux для отримання таких прав потрібно мати права адміністратора. Але при наявності вразливості в системі прав доступу не потрібно.

Для приховування своєї присутності в КС та файлах для створення комп’ютерних вірусів зловмисники можуть використовувати технології приховування та заплутування їх коду.

Worm-віруси є самопоширюваними в мережах комп’ютерними програмами. Їх відмінність від комп’ютерних вірусів в тому, що вони поширюються в мережі і є самостійними програмами.

Мережне і файлове ЗПЗ поєднує те, що для свого поширення і функціонування вони застосовують механізми встановлення контролю над КС, здійснюють пошук нового місця свого поширення та виконують певні типові

алгоритми при кожному початковому запуску.

Проведений аналіз технологій розвитку і поширення ЗПЗ [6, 28, 33, 34, 66, 76, 85, 104, 211, 212, 220, 227, 234, 245] відображає подальше використання зловмисниками новітніх технічних можливостей апаратних і програмних засобів їх створення. Особливо складними за будовою є бот-мережі, які є розподіленими програмними засобами, що здійснюють контроль зловмисником над комп'ютерними системами в мережі. Файлове ЗПЗ та worm-віруси частіше почали використовуватись для підготовки проникнення в КС та з подальшим їх використанням для розбудови бот-мереж.

Отже, розвиток технічних можливостей комп'ютерних систем та мереж і технологій програмування використовується зловмисниками для створення якісного ЗПЗ для використання в КС. Шкода, яку створює ЗПЗ користувачам КС, постійно зростає. Тому, необхідність виявлення відомого та нового ЗПЗ залишається актуальною проблемою сьогодення [172]. Із зростанням таких загроз та збільшенням їх масштабів, проблема виявлення ЗПЗ особливо актуалізується для підприємств та організацій.

1.2. Аналіз мережних систем виявлення зловмисного програмного забезпечення та достовірності результатів їх роботи

1.2.1. Аналіз існуючих мережних антивірусних засобів

Мережне антивірусне програмне забезпечення (АПЗ) є спеціальним типом антивірусного програмного забезпечення, що призначене для використання у мережах. На відміну від звичайного антивірусного програмного забезпечення призначеного для використання в персональному комп'ютері [375], мережний антивірус переважно має модуль централізованого управління [68, 87, 252, 353, 363], що дозволяє системному адміністратору керувати оновленнями і налаштуваннями всіх антивірусних клієнтів в мережі з єдиної консолі. Мережний антивірус може входити до складу засобів захисту таких класів, як

Web Security, E-Mail Security, UTM, Firewall, або бути окремим рішенням. Зазвичай мережне антивірусне програмне забезпечення використовується разом із засобами антивірусного захисту вузлів мережі (робочих станцій і серверів) як другий рівень захисту [363]. При цьому рекомендується використовувати Host Antivirus і Network Antivirus від різних виробників антивірусних систем, щоб підвищити ймовірність виявлення і здійснити вчасно блокування вірусних атак. Якщо на одному рівні захисту атака не буде виявлена, то зберігається ймовірність її блокування на іншому рівні захисту при комбінації мережних і хостових рішень від різних виробників АПЗ.

Як технологія захисту у мережних антивірусах використовується Endpoint Security - система захисту кінцевих точок у корпоративних мережах [87, 233, 363, 386]. Як правило, вона містить антивірусне програмне забезпечення, контроль над встановленими застосунками, контроль за веб-трафіком і контроль пристроїв, що підключаються. Управління всіма функціями системи Endpoint Security здійснюється з єдиної консолі, що дозволяє спростити розгортання і адміністрування цілої системи.

Мережним АПЗ від компанії «Dr.Web» [66, 85, 253] є спеціалізоване рішення «Dr.Web CureNet!» - централізований керований антивірус для страховки і посилення безпеки мереж, в яких використовується антивірус інших виробників [170]. «Dr.Web CureNet!» можна класифікувати як мережний антивірус-сканер, який централізовано поширюється локальною мережею, запускається на робочих станціях, виконує свою роботу, після чого самовидаляється. «Dr.Web CureNet!» не вимагає наявності сервера або установки будь-якого додаткового ПЗ. Його запуск можливий з будь-якого зовнішнього носія, наприклад, USB-носія. Розподіл одиниць системи «Dr.Web CureNet!» в комп'ютерних системах і серверах локальної мережі створює мінімальний мережний трафік, завдяки використанню спеціально розробленого протоколу обміну інформацією в мережах, побудованого на основі протоколів TCP / IP з можливістю застосування особливих алгоритмів стиснення трафіку.

Для захисту від дії вірусних програм, спрямованих на виведення

антивірусів з ладу, використовується технологія «Dr.Web SelfPROtect», що захищає модулі і файли «Dr.Web CureNet!» у віддаленій комп’ютерній системі перед запуском сканування. «Dr.Web CureNet!» перевіряє файли, архіви, файлові контейнери і поштові повідомлення на наявність різних загроз від ЗПЗ. Основними можливостями програми «Dr.Web CureNet!» є:

1) централізована перевірка віддалених комп'ютерів без установки антивірусу в кожну ПК, що перевіряється;

2) виявлення зловмисного програмного забезпечення та лікування інфікованих об'єктів;

3) контроль процесу сканування з комп'ютера адміністратора та збір статистики антивірусної перевірки і її відображення в комп'ютері адміністратора.

Symantec Endpoint Protection є мережним антивірусним рішенням від компанії «Symantec», що забезпечує адміністратора мережі необхідними інструментами з розгортання антивірусної мережі, її моніторингу, а також з управління параметрами роботи антивірусних клієнтів на об'єктах, що захищаються [251-254]. В рамках Symantec Endpoint Protection реалізована класична клієнт-серверна архітектура, прийнята для використання у багатьох мережних антивірусних продуктах, але існують, також, і відмінності.

У «Symantec Endpoint Protection» було досягнуто збільшення швидкодії сервера і клієнтів за рахунок наступних удосконалень:

1) сервер управління автоматично виконує обслуговування бази даних сервера;

2) компонент LiveUpdate, що відповідає за оновлення програмних модулів і вірусних баз, може запускатися під час бездіяльності клієнтів, тільки якщо є наявними оновлені модулі на сервері оновлень.

Основними перевагами даного АПЗ над іншими продуктами у цій сфері програмного забезпечення є такі:

1) висока масштабованість, що дозволяє використовувати «Symantec Endpoint Protection» у локальних комп’ютерних мережах підприємств різної

величини, наявність бази даних власної реалізації для невеликих мереж, можливість інтеграції з іншими популярними СКБД для великих мереж;

2) високий рівень можливостей з управління антивірусними клієнтами, яке організовано через множину політик безпеки, пов'язані з кожним компонентом захисту, які можна застосовувати як відразу до всіх комп'ютерів, так і до окремих комп'ютерів і груп;

3) гнучка, багатовимірна і наочна система звітності, що дозволяє в будь-який момент часу виявити як існуючі проблеми в роботі усієї мережі, так і виявити проблеми, причини яких виникли деякий час тому.

Серед недоліків «Symantec Endpoint Protection» [85, 253] є нерівноправність у підтримці різних операційних систем. Зокрема, відсутній антивірусний сервер для Linux-систем.

Апаратно-програмний антивірусний комплекс для виявлення ЗПЗ типу «0-day», для яких ще не сформовано сигнатури, запропоновано компанією Palo Alto Networks [28, 63, 85, 190]. Модель WF-500 використовує технологію Palo Alto Networks WildFire, яка полягає в запуску підозрілого файлу на множині віртуальних операційних систем і моніторингу наступних дій цього файлу. Якщо підозрілий файл виконує небезпечні або свідомо зловмисні дії, то до нього можуть бути застосовані політики безпеки, такі як видалення, карантин або сповіщення адміністратора для прийняття остаточного рішення. Описана технологія WF-500 аналогічна до хмарного сервісу Palo Alto Networks WildFire, але при цьому відсутня необхідність відсилання файлів для аналізу в хмару Palo Alto Networks.

«Malwarebytes Endpoint Security» - мережне антивірусне програмне забезпечення від компанії «Malwarebytes» [170], що пропонує розширений набір локальних засобів для виявлення і видалення загроз в комп'ютерах у мережі [258]. «Malwarebytes Endpoint Security» реалізує централізований локальний захист комп'ютерів в мережі, що ґрунтується на багаторівневій технології, здатній розірвати ланцюг атаки. Даний антивірус об'єднує провідні технології виявлення і нейтралізації зловмисного ПЗ в одному застосунку. Запатентована

«Malwarebytes» технологія, на основі якої працює «Linking Engine», забезпечує комплексний всеосяжний захист і здатна повністю відновити функціональність комп'ютера в мережі за мінімальної участі кінцевого користувача.

Рішення мережного захисту від компанії «Cisco» включає технологію «Cisco® Network Admission Control (NAC)», яка була спеціально розроблена для того, щоб допомогти забезпечити адекватний захист усіх кінцевих точок мережі (персональні комп'ютери, ноутбуки, сервери), що звертаються до мережних ресурсів, від загроз безпеці [66, 85]. Технологія NAC дозволяє адміністраторам мережі аналізувати і контролювати всі пристрої, що підключаються до мережі. Маючи в своєму розпорядженні гарантію того, що кожний кінцевий пристрій відповідає корпоративній політиці безпеки і використовує актуальні версії антивірусного програмного забезпечення, ця технологія допомагає істотно знизити або взагалі виключити кінцеві пристрої з числа джерел зараження. «Cisco» пропонує два варіанти підходу до реалізації NAC: серверний і архітектурний. Наявність альтернативного вибору дозволяє задовольнити функціональні і експлуатаційні вимоги будь-якої організації, незалежно від того, яку політику безпеки в ній проводять, що включає в себе рішення різних виробників і корпоративні засоби контролю в комп’ютерних системах.

Антивірусна програма Comodo Internet Security, що розроблена компанією Comodo Group Inc [68], від інших продуктів для захисту від ЗПЗ відрізняється тим, що виробник позиціонує його не як антивірус, а як продукт, який призначений для забезпечення комплексної безпеки. Крім антивіруса і фаєрвола, Comodo Internet Security включає в себе HIPS (входить до складу компоненти «Defense+»), автоматичну «пісочницю», віртуальний робочий стіл і великий набір спеціалізованих утиліт. Основною перевагою даного продукту є проактивний захист. Comodo Internet Security включає технологію HIPS для здійснення поведінкового аналізу дій процесів і застосунків, що дозволяє забезпечити захист системи від нового зловмисного програмного забезпечення.

«Лабораторія Касперського» пропонує спеціальний продукт для антивірусного захисту мережі - Kaspersky Administration Kit, який реалізує

наступний принцип: мережний антивірус повинен працювати автономно, приймати рішення самостійно, не видаючи запитів користувачеві і не покладаючись на доступність адміністратора [128]. Адміністратор мережі не знає, коли хтось із користувачів має намір скопіювати інформацію з зовнішнього носія або запустити програму, завантажену з підозрілого веб-сайту. Він не може виконати перевірку на вимогу неблагонадійних об'єктів, а тому виконує роботу опираючись на постійний захист і на періодичні перевірки кожного комп'ютера у мережі. Даний антивірус забезпечує високий ступінь захисту в найкоротші терміни, що дає можливість не покладатись на здатність користувача або адміністратора ініціювати ретельну перевірку окремих файлів. Крім того, адміністратор може обмежити доступ користувачів до підозрілих джерел (зовнішніх носіїв, веб-сайтів певних категорій). Більшість інцидентів безпеки обробляється автоматично. Адміністратор має можливість оцінити дії антивірусу, коректуючи їх або змінюючи налаштування, щоб в подальшому антивірус аналогічні інциденти обробляв інакше.

Крім мережних антивірусних засобів, підприємства та організації можуть використовувати і хостові антивірусні засоби. Деякі з розробників мережних антивірусних засобів використовують технології, які дозволяють комбінувати застосування мережних та хостових компонентів систем виявлення ЗПЗ, причому різних виробників. Антивірусні засоби для хостових комп’ютерних систем, які можуть здійснювати виявлення мережного і файлового ЗПЗ: AntiVir (Avira) [26], Avast! [24], AVG Antivirus [25], BitDefender [35], Clam AntiVirus [63], Dr. Web [83], Kaspersky [128], McAfee [178], Microsoft, Nod32 (Eset) [87], Norton AntiVirus (Symantec) [251, 252], Panda Cloud [190]. Для встановлення переваг мережних чи хостових засобів здійснимо аналіз результатів їх роботи.

Для проведення експерименту з перевірки достовірності виявлення ЗПЗ існуючими АПЗ було згенеровано 250 зразків файлового ЗПЗ та здійснено їх виявлення різними АПЗ. Крім, проведеного експерименту було проаналізовано дані протестування такого типу ЗПЗ з відкритих джерел Virus Bulletin [265] та AV-TEST [258]. Результати аналізу та тестування АПЗ для хостових

комп'ютерних систем представлено в табл. 1.1, а для мережних систем виявлення в табл. 1.2. Відповідні діаграми на рис. 1.1 і рис. 1.2 для цих таблиць показують ймовірності виявлення ЗПЗ на основі їх усереднених значень.

Таблиця 1.1

Порівняльний аналіз тестування АПЗ (хостові засоби)

Антивірусне ПЗ	Виявлено ЗПЗ,	Виявлено ЗПЗ,	Виявлено ЗПЗ,
Avira	77	80	74
Microsoft	78	83	75
AVG	75	74	73
BitDefender	78	78	77
Panda Cloud	82	76	66
Kaspersky	84	88	76
McAfee	85	85	78
Nod32 (Eset)	86	84	81
Norton (Symantec)	81	79	72
Avast	85	88	82
ClamAV	80	77	71
Dr.Web	84	89	81

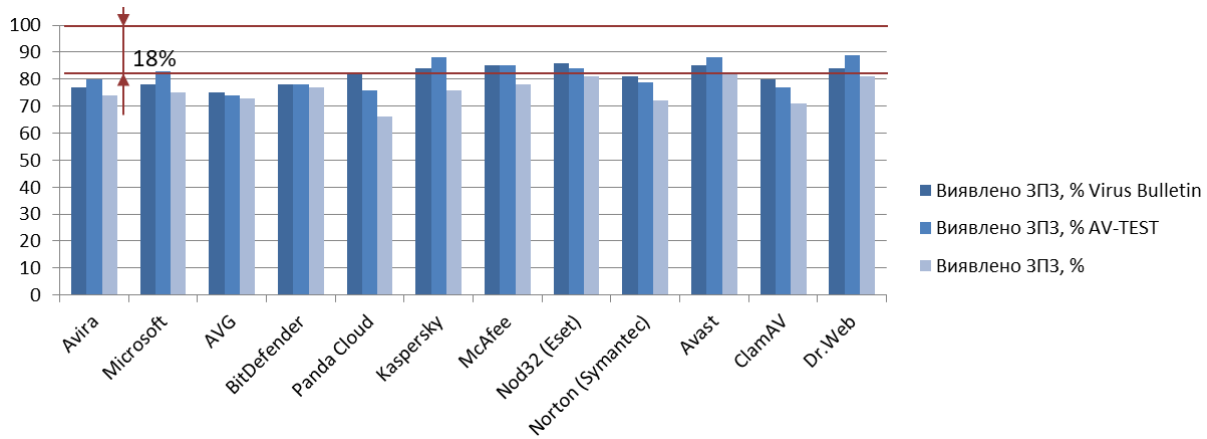


Рис. 1.1. Порівняльний аналіз тестування АПЗ (хостові засоби)

Таблиця 1.2

Порівняльний аналіз тестування АПЗ (мережні засоби)

Антивірусне ПЗ	Виявлено ЗПЗ,	Виявлено ЗПЗ,	Виявлено ЗПЗ,
Avast Endpoint Protection Suite	89	87	82
AVG Internet Security Business Edition	81	81	83
Avira Small Business Security Suite	84	79	80
BitDefender Corporate Security	78	79	79
Dr.Web CureNet	82	86	80
ESET Endpoint Security	85	83	81
Kaspersky Endpoint Security	79	82	79
McAfee Endpoint Protection Suite	91	88	85
Microsoft System Center Endpoint Protection	82	84	74
Panda Endpoint Protection	81	79	82
Symantec Endpoint Protection	87	85	84

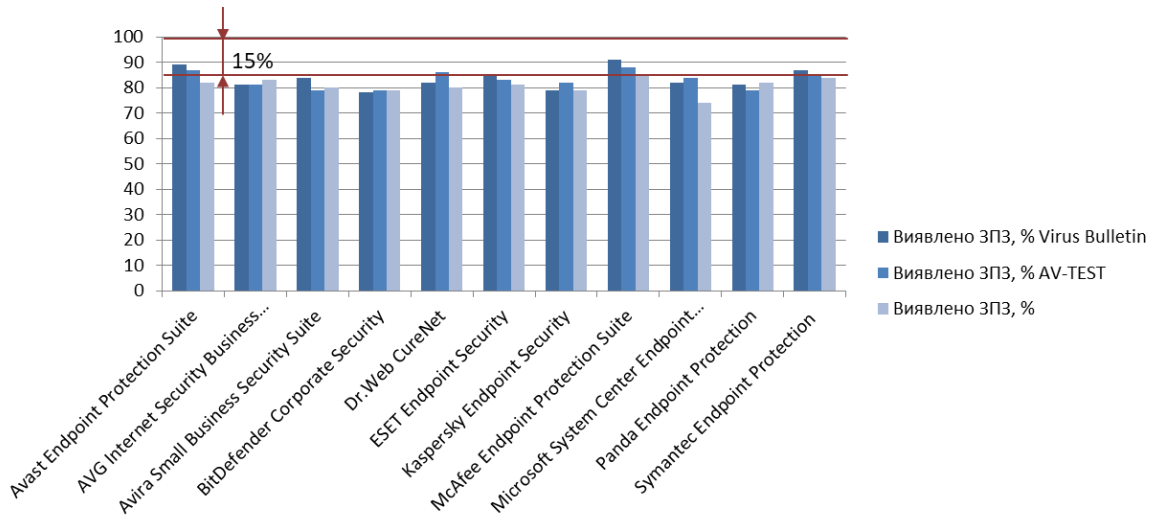


Рис. 1.2. Порівняльний аналіз тестування АПЗ (мережні засоби)

Результати аналізу та проведеного експерименту підтверджують неможливість повного виявлення файлового ЗПЗ існуючими АПЗ. Достовірність виявлення хостовими та мережними АПЗ майже не відрізняється. При великій кількості ЗПЗ (AV-TEST [258]) відсоток невиявленого ЗПЗ містить достатньо велику його кількість, що вимагає здійснення постійного подальшого дослідження з метою застосування інших методів виявлення.

На основі проведених досліджень мережних АПЗ на наявність ЗПЗ можна зробити наступні висновки:

– за умов інтенсивного зростання кількості нового ЗПЗ та високого рівня кваліфікації розробників ЗПЗ, існуючі антивірусні засоби, які здійснюють виявлення, не задовольняють вимогам ефективного виявлення;

– розробники АПЗ демонструють високу достовірність виявлення ЗПЗ, приховуючи результати реальної достовірності виявлення нового ЗПЗ;

– згідно проведеного тестування АПЗ, використання методів і технологій на основі евристичних аналізаторів показує низьку достовірність виявлення нового ЗПЗ;

– для використання в локальних комп’ютерних мережах організацій та підприємств розроблено відповідні мережні АПЗ, достовірність виявлення якими не суттєво відрізняється від достовірності виявлення хостовими АПЗ;

– при необхідності здійснення виявлення ЗПЗ в локальних комп’ютерних мережах перспективними є мережні системи виявлення, особливо якщо КС піддаються ураженню мережним ЗПЗ;

– для покращення виявлення ЗПЗ в КС актуальною є розробка нової мережної системи виявлення ЗПЗ в КС, яка б мала здатність виявляти відоме та нове ЗПЗ в ЛКМ.

Результати тестування АПЗ показують, що мережні АПЗ не забезпечують повного виявлення ЗПЗ [66, 258, 265]. Тому, необхідним є подальша розробка мережних АПЗ з використанням нових методів і програмних технологій, які б для покращення ефективності виявлення враховували архітектурні особливості локальних комп’ютерних мереж.

1.2.2. Характеристика відомих систем виявлення зловмисного програмного забезпечення

Для виявлення та протидії ЗПЗ розроблено системи та методи, зокрема представлені в [32, 41, 62, 79, 88, 103, 107, 111, 112, 122, 139, 192, 210, 222, 235, 235, 240, 257, 270, 317, 318]. Розглянемо типові з них.

В роботах [317, 318] представлено концепцію створення багаторівневої комплексної системи безпеки кіберфізичних систем (КФС), яка орієнтована на розвиток концептуальних засад захищеної взаємодії рівнів та компонентів у просторі “конфіденційність – цілісність – автентичність” відповідно до етапів створення та функціональної реалізації КФС у предметних сферах. Також, обгрунтовано доцільність розподілу задач системи між рівнями.

В роботах [206, 332–333] запропоновано систему для дослідження поширення бот-мереж. При цьому розроблено візуазізацію подій та представлення атак графами. Це дало можливість динамічно аналізувати і перебудовувати роботу адміністратору мережі.

В роботі [41] запропоновано систему виявлення та класифікації підключень в мережному трафіку використовуючи різні схеми гібридизації з метою ефективного виявлення мережних атак. З цією метою було використано комбінацію різних методів обчислювального інтелекту, а саме нейронні мережі, імунні системи, нейронечіткі класифікатори та метод опорних векторів. Відмітною особливістю запропонованого підходу є багаторівневий аналіз мережного трафіку, що забезпечує можливість виявлення атак за допомогою технології на основі підпису та об'єднання набору адаптивних детекторів на основі використання обчислювального інтелекту.

В роботі [139] розроблено систему захисту інформації для виявлення кібератак на основі залучення нейромережних імунних детекторів. Розроблена система захисту інформації складається з двох частин. Перша реалізована апаратно й працює постійно в режимі реального часу (реалізація здійснена на основі використання програмованих матриць логіки). Друга частина

представлена програмним забезпеченням на виділеному комп'ютері, який використовується для аналізу поточних атак та створення відповідних засобів захисту. Прийняття рішення про можливий вплив ЗПЗ здійснюється із залученням системи нейромережних детекторів.

Авторами роботи [137] запропоновано систему класифікації зловмисного коду, де в якості ознак було використано послідовності n-грам програмного коду. Після вибору найбільш інформативних n-грам для класифікації, автори оцінили різні методи машинного навчання, включаючи наївний байєсів класифікатор, дерева рішень та метод опорних векторів. В ході дослідження було встановлено, що найкраща ефективність класифікації досягається при використанні дерев прийняття рішень.

У роботі [103] запропоновано гібридну систему виявлення рідковживаних кібератак на основі використання штучних імунних систем та нечіткої кластеризації (FC-ANN). Спочатку FC-ANN розподіляє навчальні дані на декілька підгруп з використанням методів нечіткої кластеризації. Для отримання остаточних результатів, визначається оцінка для кожного елементу з сформованих підгруп та виконується їх поєднання з використанням штучної імунної системи.

У роботі [107] запропоновано метод, що заснований на подібності бінарних поведінок. Для відстеження та запису поведінки бінарних файлів в процесі виконання залучається контрольоване віртуальне середовище. Підхід передбачає, що поведінка кожного двійкового файлу може бути представлена значеннями вмісту пам'яті в момент виконання. Тобто, значення, що зберігаються в різних регістрах під час запуску зловмисного програмного забезпечення в контрольованому середовищі, може відрізнятися від тих, що притаманні корисним програмам. Потім значення регістрів для кожного виклику інтерфейсу прикладних програм (API) видобуваються до і після викликів API. Після цього здійснюється розподіл та визначається зміна значень регістрів по виконуваному файлу і створення вектора для кожного з регістрів EAX, EBX, EDX, EDI, ESI і EBP. Порівнюючи показники подібності між існуючими та

невідомими векторами зловмисних програм, здійснюється формування висновку про інфікування системи.

Проведений аналіз показав, що для виявлення ЗПЗ відомі системи здійснюють аналіз мережного трафіку, файлів аудиту, пакетів, що передаються по мережі, перевіряють конфігурацію відкритих мережних сервісів та структурних особливостей виконуваних файлів. Для встановлення факту порушення роботи КС, відомі системи використовують різні методи машинного навчання, а саме нейронні мережі, штучні імунні системи, метод опорних векторів, Байєсові мережі, нечітку кластеризацію. Проте, основним недоліком представлених систем є їх хост-орієнтований підхід до виявлення ЗПЗ.

Таким чином, хостові антивірусні засоби та мережні системи виявлення ЗПЗ не забезпечують його повного виявлення. В КС залишається невеликий відсоток невиявленого ЗПЗ, про що інформують і самі розробники АПЗ [63, 254, 258, 265], публікуючи свої тести, і сторонні тестувальники АПЗ. Для підвищення достовірності виявлення ЗПЗ в мережах підприємств та організацій, як другий рівень, використовують мережні системи виявлення ЗПЗ, які теж не забезпечують повного виявлення. В результаті в КС залишається певний невеликий відсоток ЗПЗ за даними самих же виробників АПЗ. Якщо ж врахувати, що до мережі Internet під’єднується все більша кількість КС, швидкість передачі інформації в мережах стрімко зростає та суттєво покращуються технічні показники КС, а також кількість ЗПЗ зростає в декілька разів в порівнянні з попередніми періодами, то невеликий відсоток невиявленого ЗПЗ, який декларують розробники АПЗ, може містити кількісно багато ЗПЗ, причому різного типу. Саме це ЗПЗ і може завдавати тривалої шкоди власникам КС, перебуваючи там тривалий час. Наприклад, таким ЗПЗ можуть бути вузли бот-мереж, експлоїти, троянські програми та ін. Для їх виявлення потрібне тривале спостереження. Враховуючи актуальність цього виявлення для підприємств і організацій, розробка нових методів та засобів виявлення ЗПЗ в локальних комп’ютерних мережах, особливо орієнтованих на тривале спостереження за процесами, які виконуються в КС та мережі, є актуальною проблемою.

1.3. Методи виявлення зловмисного програмного забезпечення та критерії їх класифікації

1.3.1. Методи виявлення зловмисного програмного забезпечення

Розділимо відомі методи виявлення мережного ЗПЗ на дві групи, в яких згідно їх стратегій передбачено тривале спостереження або оперативна реакція на нові події: активні і пасивні [354, 355].

Розглянемо пасивні методи виявлення, до яких віднесемо методи, суть яких у використанні мереж-приманок, здісненні віддаленої аутентифікації коду, здійсненні пасивного моніторингу трафіку, виконанні моніторингу груп проявів в DNS-трафіку [351]. Розроблені методи на основі мереж-приманок (Honeypot і Honeywall), суть яких така: в [27] в якості приманки використовується комп'ютерна система, яка є вразливою до атак зловмисників і успішно атакованою в дуже короткий проміжок часу, і Honeywall є програмним забезпеченням для моніторингу, збору, контролю та зміни трафіку через пастки, такі як Snort; в роботі [213] запропоновано приманки низької взаємодії, для цього використовуються PHP та емуляції декількох вразливостей в Мамбо (Mambo) і Австатс (Awstats), снорт (Snort) виявляє первинний обмін сигналами, зокрема, вихід 'ID' команди; в роботах [152-154] запропоновано дослідження сканування трафіку бот-мереж, що базується на Honeynet, на основі досліджень було зроблено припущення, що більшість бот-мереж дійсно використовують випадкові стратегії сканування, але така стратегія є неефективною для виявлення P2P бот-мереж; у [52] запропоновано використовувати приманки і P2P бот-мережі, при цьому приманки використовуються в синхронізації з P2P бот-мережею, але є обмеження на кількість таких приманок (в середньому від 50 до 100).

Завдяки віддаленій аутентифікації коду, яка представлена в [289, 290], здійснюється допомога в зборі інформації про різні бот-мережі з приманки бота. Дослідження показують, що worm-віруси і бот-мережі можна

контролювати, але зупинити атаки, навіть після виявлення загрози, складно.

Метод пасивного моніторингу трафіку, який представлено в [281] передбачає здійснювати виявлення бот-мереж на базі з семи основних компонент: фільтрації, застосування класифікатору, моніторингу трафіку, детектору зловмисної активності, аналізатору, моніторингу та кластеризації і поточного аналізатору виявлення бот-мереж. Від інших методів він відрізняється тим, що відсутня необхідність мати попередні знання про бот-мережі, але для бот-мереж, які використовують компоненти поліморфного коду, даний метод має низьку достовірність результату виявлення.

Методи на основі моніторингу групи проявів в DNS трафіку: в [57, 58] запропоновано виявлення бот-мереж за допомогою різних функцій бот-мереж і легітимних DNS, де вимагається дослідження трьох складових частин, зокрема вставка-DNS-запит, видалення-DNS-запит, виявлення-BotDNS-запит, при цьому використовується база даних для зберігання даних DNS-запитів, які включають IP-адресу джерела в запиті, доменне ім'я запиту і мітку часу отриманого запиту, всі згруповані дані DNS-запитів за іменем домену і тимчасовою міткою, проблемою є час обробки даних.

Розглянемо такі активні методи: Data-mining, метод аналізу графів «користувач-користувач», сигнатурний метод. Суть цих методів в наступному: в [31] представлено створене середовище Vmware на сервері з процесором Intel та встановленою операційною системою Windows XP з усіма новими оновленнями, призначеною статичною адресою, в ході експерименту було інфікована КС одним ботом протягом 30 хвилин, далі їх код зчитували бінарними кодами бота з файлу один за одним і виконували його у відкритому середовищі, отримані результати були надіслані на сервер у вигляді файлу, включаючи повне корисне навантаження, таке середовище виконання мереж-приманок дозволило впровадити зловмисний зразок ботів в систему і підключитись назад в свій початковий стан, недоліком цього методу є використання даних антивірусу Symantec для класифікації ботів; в [284] було запропоновано моделі випадкового графу для аналізу графів «користувач-

користувач», які показують, що групи бот-користувачів відділяють себе від груп користувачів, що можна представити компонентами у графі, недоліком методу є використання інформації тільки з поштової служби (Hotmail); сигнатурний метод базується на знаннях автентичних частин коду, зокрема система Snort є системою виявлення вторгнень (IDS) з відкритим вихідним кодом, яка контролює мережний трафік для знаходження ознак вторгнення. Вона налаштовується за допомогою набору правил або сигнатур для реєстрації трафіку, який вважається підозрілим. Сигнатурний метод виявлення може бути використаний для виявлення відомих бот-мереж, але недолік його у неможливості виявлення невідомих ботів [96].

Методи виявлення бот-мереж спрямовані на збір інформації про стан мережі або на виявлення існуючих бот-мереж. Відслідковування надсилання спам-повідомлень не є ефективним для виявлення бот-мереж, а спрямоване на зниження кількості надісланих спам-повідомлень. Ці недоліки вимагають розробки нових методів виявлення бот-мереж, які б дозволили, крім збору інформації про стан мережі, визначати присутність бот-мереж та блокувати їх, що дозволить б запобігти поширенню спам повідомлень і блокуванню електронних ресурсів, з яких можуть надходити зовнішні команди.

Розроблено методи виявлення бот-мереж, які базуються на основі DNS, і поділяються за таким особливостями: за динамікою активності бот-мереж в залежності від часових зон; на основі динамічної оцінки репутації доменних імен; побудовою евристики для виявлення DNSBL-розвідувальної діяльності зловмисника та складання списків ймовірних ботів; проведення аналізу поведінки ботів і не ботів, пов’язаної з реакцією на DNS-відповіді; на основі властивості групової активності бот-мереж в DNS-трафіку; моніторингу та захоплення DNS-трафіка в різних часових інтервалах та вимірювання відношення подібності між будь-якими двома групами КС, що запитують одне й те саме доменне ім’я; на основі кластеризації методом х-середніх (*x-means*) ознак, вилучених з DNS-трафіку; виявлення групових неуспішних DNS-запитів ботів в мережі, що здійснюють неуспішні спроби мережних з’єднань та мають

подібний розмір корисного навантаження пакетів при спробі мережного з'єднання; виявлення C&C-серверів бот-мереж за аномально високими або сконцентрованими в часі рівнями DDNS-запитів.

Розглянемо методи виявлення бот-мереж на основі DNS: в [69, 70] побудовано моделі, які описують динаміку активності бот-мереж в залежності від часових зон та допомагають передбачати зростання чисельності популяції бот-мережі, тому в графіку активності бот-мереж за країнами походження інфікованих комп'ютерів проявляється добовий шаблон, утворений значним підвищенням обсягів рекурсивного DNS-трафіку; в [18–20] запропоновано динамічну систему оцінки репутації доменних імен, яка використовує три групи ознак для побудови моделей легітимних та зловмисних доменів – мережні, зональні та доказові ознаки, на основі яких здійснюється оцінка репутації домена, тому доказові ознаки базуються на даних «чорних списків» та систем-«приманок» і дозволяють визначити, в якій мірі домен пов'язаний з відомими зловмисними доменними іменами або IP-адресами; в [214] для виявлення DNSBL-розвідувальної діяльності зловмисника та складання списків ймовірних ботів запропоновано евристику, при цьому з метою визначення, чи знаходяться спам-боти в «чорному списку», зловмисник виконує DNSBL-запити, і згідно методу будується граф запитів, на основі якого визначено очікувані просторова та часова характеристики легітимних пошуків в порівнянні з розвідувальними пошуками, що допомагають відрізнити запити зловмисника від запитів легітимних поштових серверів, недоліком методу є те, що він передбачає використання одного хоста для вхідного та вихідного поштового серверів, а у великих мережах вони можуть бути розділені, і тоді запити від вхідного поштового сервера можуть бути розцінені, як спроба розвідки; в [179] розроблено метод, заснований на аналізі поведінки ботів і не ботів в КС, пов'язаної з реакцією на DNS-відповіді, при цьому визначено чотири підозрілі процеси, що можуть мати місце в поведінці: успішне пряме перетворення імені, підключення не відбулось – аномальний процес, успішне зворотне перетворення імені, підключення не відбулось – притаманний для ЗПЗ, в тому числі ботів;

неуспішне зворотне перетворення імені, підключення відбулось – притаманний для ЗПЗ, в тому числі ботів; неуспішне зворотне перетворення імені, підключення не відбулось – домінуючий для ботів; в [3] визначено особливості характерної поведінки для бот-мереж, на основі якої зв'язок між зловмисником та вузлами бот-мережі здійснюється за рахунок контролю множини вузлів через видачу однієї команди для всіх, тому вузли бот-мережі виявляють групову поведінку, що дозволяє виявити їх активність на основі моніторингу мережного трафіку: вузли бот-мережі отримують одну команду від зловмисника, підтримують зв'язки між собою та атакують одночасно, що дозволяє виявити групу вузлів бот-мережі за допомогою високого показника трафіку, відповіді вузлів бот-мережі на команди зловмисника, що дозволяє врахувати час відгуку, який стає постійним і може бути використаний для виявлення.

Методи, які базуються на властивості групової активності бот-мереж в DNS-трафіку: в [171] розроблений метод передбачає здійснення моніторингу, захоплення DNS-трафіка в різних часових інтервалах та вимірювання відношення подібності між будь-якими двома групами КС, що запитують одне й те саме доменне ім'я, основним недоліком є його зосередженість на групові запити лише однакових доменних імен, не враховуючи міграцій C&C-серверів та інших DNS-запитів, пов'язаних з діяльністю бот-мережі; в [57] метод орієнтований на виявлення ботів в межах класу, для обчислення подібності між двома групами КС з використанням коефіцієнту Кульчинського, суть якого в порівнянні списків IP-адрес КС, що запитували різні доменні імена, але які подібні за розмірами, основним його недоліком є значне зростання часу обробки та потреба у великих обсягах обчислювальних ресурсів при застосуванні до великих мереж; визначення множини доменних імен бот-мережі, і на їх основі виявлення міграцій C&C-серверів з використанням кластеризації методом x-середніх (*x-means*) ознак, вилучених з DNS-трафіка; в [58] запропоновано метод, в якому з метою обчислення значення подібності між двома групами КС використовується косинусний коефіцієнт, недоліками якого є його орієнтація на виявлення ботів у великих розподілених мережах, тому не придатний для

невеликих локальних мереж. Недоліками розлянутих методів з [57, 58, 171] є поділ періоду моніторингу на довільні інтервали, в межах яких здійснюється пошук груп інфікованих КС, що призводить до зменшення рівня виявлення та використання для порівняння двох груп КС симетричних мір подібності.

Метод виявлення групових неуспішних DNS-запитів ботів в мережі представлений в [216] передбачає пошук неуспішних спроб мережних з'єднань, недоліком якого є необхідність у додатковому зборі та аналізі TCP-трафіку.

Метод виявлення C&C-серверів бот-мереж за аномально високими або сконцентрованими в часі рівнями DDNS-запитів представлено в [70], який пов'язує рівні надходження DNS-запитів щодо доменів другого рівня з рівнями DNS-запитів щодо піддоменів третього рівня. Тоді, можна здійснити оцінку рейтингу доменів на основі обсягу трафіку. В [261] розроблено експериментальну оцінку двох відомих підходів до виявлення C&C-серверів бот-мереж на основі аномально високих або сконцентрованих в часі обсягах запитів доменних імен та аномально високому рівні відповідей.

Методи виявлення на основі використання бот-мережами технік ухилення на основі DNS: в [272] використано сервіс пасивної реплікації DNS, що надає можливість виявлення застосування періодичної зміни IP-відображення для доменних імен C&C-серверів бот-мереж, суть якого полягає в спостереженні DNS-трафіку, зберіганні всіх завершених DNS-резолюцій та побудові частин файлу зони, недоліком є те, що такі сенсори корисні лише на ділянках мережі з передбачувано високим трафіком; в [186] виявлення доменних імен бот-мереж, що використовують технологію «швидкозмінних» мереж, здійснюється за множиною ознак, отриманих на основі аналізу A-, NS-, SOA- та BGP-запитів щодо доменного імені, недоліком якого є необхідність активного DNS-зондування, тому його неможливо реалізувати виключно на основі пасивного аналізу DNS-трафіку; в [32, 33] виявлення зловмисних доменів здійснюється на основі пасивного DNS-аналізу, що здійснює класифікацію доменних імен за чотирма групами ознак, які можуть бути вилучені з DNS-трафіку; в [118, 277] виявлення бот-мереж здійснюється на основі аналізу історії NS-записів; в [284]

здійснення ідентифікації легітимних доменів CDN (Content Delivery Network) та доменних імен C&C-серверів бот-мереж, які використовують технологію ухилення «швидкозмінних» мереж (Fast Flux Service Networks, FFSN) базується на виявленні ознак, отриманих на основі активного DNS-зондування, а саме співставлення домену з IP-адресами, які знаходяться в різних сегментах мережі, а також врахування коливань часу обробки запитів, що пов'язані з відмінностями у рівні продуктивності різних проксі-серверів в «швидкозмінних» мережах, для множини послідовних в часі DNS-запитів щодо певного доменного імені; в [276] виявлення бот-мереж, що використовують технологію «потік доменів», базується на аналізі успішних та невдалих DNS-запитів, представлених в інтервалі часу, що містить успішні запити, перевагою якого є те, що він не покладається на високі рівні DNS-запитів, а недоліком є концентрування на невдалі доменні імена, а також чутливість до вибору часового інтервалу, збільшення якого підвищує ймовірність хибних спрацювань, а зменшення знижує відсоток виявлення.

Методи виявлення з використанням DNS-тунелювання можна поділити на дві групи: засновані на аналізі корисного навантаження DNS-повідомлень; засновані на аналізі DNS-трафіку.

Методи виявлення бот-мереж з використанням DNS-тунелювання такі: в [93] надано оцінку статистичних методів виявлення аномалій у вмісті DNS-пакетів шляхом порівняння ймовірнісних розподілів нормального та тунельованого DNS-трафіку; в [123, 124] проведено аналіз реальних DNS-запитів та обчислено порогові значення довжини запитаного імені хоста та кількості унікальних символів в ньому, що дозволяють відрізнити легітимний DNS-трафік від тунельованого; в [78] представлено бот-мережу Feederbot, в якій використовується технологія DNS-тунелювання. Основним недоліком методів є зосередження на виявленні типового ЗПЗ, що використовує окремі технології DNS-тунелювання.

Крім методів, які систематизовано за певними критеріями, розглянемо інші методи виявлення бот-мереж, суть яких, зокрема: в [29, 45] пропонується

використовувати динамічне виявлення, в [56] досліджені керування бот-мережею на основі фільтрації трафіку, в [105] закладено в основу підходу врахування поведінки за результатами дослідження DNS, в [113] на основі швидкозмінних мереж, в [125] врахування співвіснування з операційними системами, в [126, 127] на основі аналізу аномалій, в [136] на оснвоі системи ведення атак проти атак бот-мереж, в [141] враховано особливості для систем IoT, в [150] досліджено поведінку КС мережі, в якій наявна бот-мережа, в [160] враховано здійснення атаки за наявності пристрою в системі, в [161] на основі прихованих марківських моделей для децентралізованих бот-мереж і врахуванням евристик, в [175] на основі динамічного використання і дослідження DNS, в [195] на основі пасивного аналізу DNS, в [199] на основі розробленого генератору із самонавчанням, в [217] в якості індикатору вибрано трафік, в [231] на основі поведінки вузла бот-мережі із використанням DNS, в [236] здійснення аналізу для виявлення на основі побудови відповідного графу, в [245, 246] на основі використання класифікатору, в [249] на основі моделей поведінки з врахуванням DNS-трафіку, в [268] на основі здійснення локації в реальному часі, в [269] на основі використання комбінованого децентралізованого зв’язку, в [274] з використанням машинного навчання при виявленні, в [280] на основі класифікації для здійснення виявлення, в [282] на основі зібраної статистики про трафік, в [285] на основі протидії розподіленій атаці в реальному часі використанням коректуючого аналізу, в [320] на основі використання підготовлених стратегій шляхом підтвердження наявності користувача, в [350] на основі поділу подій на декілька груп в КС з оцінкою їх віднесення до проявів бот-мереж, в [271] на основі алгоритму виявлення бот-мереж за шаблонами.

Аналіз наявних технологій приховування бот-мережами своєї присутності відображає складність процесу їх виявлення. Відомі методи виявлення є різноманітними за етапами застосування в процесі функціонування бот-мереж, але не охоплюють комплексно наявні технології розвитку бот-мереж. Це унеможливлює застосування відомих методів до бот-мереж з різними технологіями. Використання всіх наявних методів виявлення є громіздким і

тривалим в часі, що не відповідатиме вимогам актуального часу для виявлення. Тому, потрібна уніфікація певних особливостей та характеристик бот-мереж для розробки більш узагальнених методів виявлення на різних етапах функціонування бот-мереж, які використовують різні техніки приховування своєї присутності.

Крім аналізу методів виявлення мережного ЗПЗ, важливим є аналіз відомих методів виявлення файлового ЗПЗ, яке може самостійно функціонувати, так і бути частиною мережного. Розвиток методів виявлення файлового ЗПЗ пов'язаний з розвитком безпосередньо самого ЗПЗ. Крім традиційних технологій створення ЗПЗ, активно розвиваються технології з використанням поліморфізму та метаморфізму, що дозволяють створювати ефективне від виявлення ЗПЗ. Методи виявлення файлового ЗПЗ на основі класичних технологій (сигнатурний аналіз, метод контрольних сум та ін.) відомі зловмисникам і тому, вони використовують технології їх обходу. Всі методи виявлення поліморфних та метаморфних вірусів можна розділити на групи: сканери першого покоління, сканери другого покоління, спеціалізовані методи виявлення та методи емуляції програмного коду. Розглянемо детальніше кожну групу методів. Сканери першого покоління застосовували для виявлення існуючих вірусних програм, використовуючи пошук частин коду, що були характерними для визначеного типу вірусу [43, 51, 158, 208, 232] і зберігаються в базах сигнатур. В подальшому сигнатурне сканування здійснювалось додаванням деякої умови під час формування сигнатури, тобто виконанням оптимізації формування сигнатури вірусу. Також, подальші дослідження і розвиток стратегій виявлення полягали в оптимізації показників ЗПЗ та їх взаємозв'язку, зокрема: метод виявлення за маскою дозволяє не порівнювати постійні значення байтів або діапазони байтів підозрілого коду з сигнатурою; метод загального ступеня схожості застосовується для виявлення вірусів, що належать до одного сімейства і характеризується формуванням загальної сигнатури, що відповідає множині вірусів; методи підвищеної швидкодії пошуку сигнатури, які характеризуються створеням закладок, хешування, сканування початку та кінця і відстеження точки

входу у програму [74, 86]; метод виявлення метаморфних вірусів скануванням файлу та створенням його сигнатури, починаючи з точки входу в програму [330]. Використання сканерів другого покоління дозволяє підвищити достовірність виявлення поліморфних та метаморфних вірусів. В цю групу входять наступні методи: інтелектуальне сканування [176, 207], метод визначення каркасу вірусу, метод наближеної ідентифікації [208], метод точної ідентифікації [209] та евристичні методи [185] виявлення. Евристичні методи поділяють на дві групи: статичні та динамічні [185]. Статичні евристичні методи базуються на аналізі структури файлу і організації коду вірусу. Динамічний евристичний сканер виконує емуляцію вірусного коду з метою збору інформації. Основною перевагою евристичних методів виявлення є здатність до ідентифікації нового ЗПЗ. Але, основним недоліком цих методів є високий відсоток помилок першого та другого роду. Спеціалізовані методи виявлення орієнтовані на виявлення вірусів одного типу або одного сімейства, зокрема статичний метод виявлення розшифровувача, фільтрування та X-Ray сканування [151, 135, 197, 260]. Методи виявлення поліморфних та метаморфних вірусів такі: на основі емуляції виконання підозрілого коду [39, 98, 99, 137], динамічне виявлення розшифровувача на основі комбінації методів емуляції виконання та статичного виявлення розшифровувача; в [283] розроблено статичний метод виявлення метаморфних вірусів, що ґрунтується на аналізі потоків даних та управління у трасах викликів виконуваного файлу; на основі оцінки частоти появи інструкцій [166, 209]; порівняння копій метаморфних вірусів шляхом розбиття їх на блоки з метою формування гістограм частоти появи інструкцій у відповідних блоках [209]; у [166] для побудови гістограми частот інструкцій використано інструкцію SUB та 14 найпоширеніших інструкцій, таких як add, and, call, cmp, jmp, jnz, jz, lea, mov, push, pop, retn, test, xor, що відібрані на основі дослідження частоти появи операційних кодів у вірусних програмах. З метою ухилення від емуляції деякі види поліморфних та метаморфних вірусів здатні розпізнавати захищене середовище та призупиняти власне виконання, що унеможливлює виявлення такого ЗПЗ [53, 92, 140, 160].

Методи, які базуються на основі статистичної оцінки інструкцій процесора, для метаморфних вірусів, що використовують техніку переміщування блоків коду, є неефективними, оскільки частота появи інструкцій в зміненій версії метаморфного вірусу не зміниться. Методи виявлення метаморфних вірусів такі: в [283] статистичний метод, який також в якості ознак для ідентифікації використовує операційні коди; в [260] розроблено метод виявлення з використанням критерію ксі-квадрату на основі очікуваних частот появи інструкцій; методи з використанням прихованих марківських моделей [11, 21, 22, 259]; метод, що передбачає використання мнемонік n-грам (4-грам), інструкцій та структурних особливостей PE EXE заголовку файлу для виявлення представлено у роботі [264]; в [4, 12, 13, 23, 47, 48, 134, 156, 157, 159, 193, 219, 257] виявлення метаморфних та поліморфних вірусів здійснюється на основі аналізу графу API викликів; в [215, 218, 221] на основі виділення операційних кодів програмних об’єктів для виявлення метаморфних кодів та невідомого ЗПЗ, в [192] застосовано використання представлення вузлів, що позначають системний виклик, у вигляді кортежу з чотирьох ознак; в [41]метод передбачає представлення підпрограм у вигляді вершин графу, а їх викликів у вигляді орієнтованих ребер; в [148,149] використовується граф залежностей між рядком програми, що представляється у вигляді вершини графу, та ребрами, які визначають залежність між двома рядками коду програми; в [54] здійснюється автоматичне отримання сигнатури поліморфного коду з використанням абстрактного інтерпретатору; в [94] отримання шаблону поведінки експлоїту з використанням апаратного пристрою для віртуальної машини; в [132] дослідження веб-аномалій з використанням штучних імунних систем, в [273] основою виявлення подано структуру метаморфного механізму, в [278] розроблено систему виявлення обфускації в ЗПЗ. Використання розглянутих методів є утрудненими через NP-повноту завдання і обчислювальну складність, що властива таким алгоритмам.

В [357] представлено підхід на основі теоретичного програмування для виявлення метаморфного зловмисного коду, суть якого полягає в здійсненні

перевірки еквівалентності в алгебраїчних моделях послідовних програм, оцінці стійкості деяких обфускуючих перетворень. Авторами доведено, що для частини випадків є ефективні методи виявлення, а для певних випадків задача виявлення зводиться до NP-повних задач, рішення яких є неефективним.

Таким чином, розробники файлового ЗПЗ використовують сучасні технології програмування і методи для поширення і приховування його присутності в КС [238].

Проведений аналіз показав, що сучасне ЗПЗ, зокрема бот-мережі [269], worm-віруси, поліморфні [275] і метаморфні [266] віруси, активно застосовують технології приховування та заплутування програмного коду, технології ухилення від емуляції, технології приховування свої присутності, мережні технології поширення, що призводить до недостатньо високого рівня достовірності їх виявлення сучасними методами. Тому, актуальним є розроблення нових методів виявлення ЗПЗ, які б дозволили покращити ефективність виявлення, отримання якої досягалось би за рахунок залучення розподілених обчислювальних ресурсів в локальних комп'ютерних мережах для прийняття рішення про наявність ЗПЗ.

1.3.2. Критерії класифікації методів виявлення ЗПЗ

Для здійснення класифікації методів виявлення ЗПЗ використовують різні підходи. Всі методи виявлення ЗПЗ можна розділити на два класи: методи виявлення відомого та нового ЗПЗ [101, 102]. Класифікація була представлена в [358] з урахуванням наступних параметрів: дані, що отримані про досліджуване ПЗ; способи отримання цих даних; математичні методи, що застосовуються для аналізу даних та ознаки підозрілості, що характеризують ПЗ. Однак, запропонована класифікація не враховує очікуваний результат, що отримується в процесі виконання аналізу ЗПЗ.

Інший однокритеріальний підхід до класифікації методів виявлення ЗПЗ і систем виявлення вторгнень заснований на виборі методу машинного навчання:

нейронні мережі, метод опорних векторів, байєсівські мережі, нечітка логіка, дерева прийняття рішень, тощо [240]. Проте, використання однокритеріального аналізу для проведення класифікації ЗПЗ є неприйнятним, оскільки не враховується, які саме дані будуть використані в якості навчальної та тестової вибірки та яким чином було отримано ці дані.

З метою усунення недоліків відомих класифікацій розроблено класифікацію методів виявлення ЗПЗ, в основу якої закладено шість критерії: характер отриманих даних; ознаки, що виступають об’єктом пошуку та дослідження; методи аналізу; алгоритм прийняття рішень; очікуваний результат та оцінка класифікації (табл. 1.3). Розглянемо більш детально кожен критерій класифікації.

Таблиця 1.3

Критерії класифікації

№ п/п	Назва критерію	Ознаки критерію
1	Алгоритм прийняття рішення	Експертна система, дерева рішень (алгоритм C4.5), приховані марківські моделі, найбільша спільна послідовність, метод опорних векторів, міра подібності косинус
2	Очікуваний результат	Точні та наближені методи
3	Ознаки, що виступають об’єктом пошуку та дослідження	Структурна інформація про файл, мережний трафік, множина операційних кодів, послідовності API викликів, граф потоку управління
4	Характер отриманих даних	Сигнатурні та евристичні методи
5	Методи аналізу	Динамічні та статичні методи
6	Оцінка класифікації	Правильність класифікації, точність, повнота, F-міра, крива помилок, AUC

За характером отримуваних даних всі методи виявлення ЗПЗ можна розподілити на сигнатурні та евристичні. Сигнатурні методи використовують пошук частин коду, що були характерними для визначеного типу вірусу чи іншого ЗПЗ. Ці ділянки коду називаються сигнатурами вірусної програми (string pattern) і зберігаються в антивірусній базі сигнатур. Така послідовність байтів повинна вибиратися з якомога менш схожими з рядками коду в довірених застосунках чи інших типах вірусів.

Методи, що використовують набори правил (евристик), активізація яких, є припущенням про інфікування КС вірусною програмою називаються евристичними методами виявлення. Головною перевагою використання евристичних методів є здатність виявляти нові види вірусних програм. Крім того, евристичні методи можуть не використовувати бази даних про вірусні програми, що представлені сигнатурами вірусів.

Наприклад, для ЗПЗ в якості складових евристичних правил використовують елементи, які виражають деякі структурні особливості, що не притаманні корисним застосункам, а саме: наявність розриву між розділами, розташування виконуваного коду в останній секції коду, підозрілі характеристики розділу, підозріле ім’я секції (стандартними вважаються імена секцій .data, .code, .reloc, .idata, .text та ін.), невірне значення віртуального розміру в заголовку PE, використання декількох заголовків PE, підозрілі значення таблиці імпорту з бібліотеки KERNEL32.DLL, підозріле направлення коду [226]. Для бот-мереж, наприклад, евристичними складовими можуть бути такі: повторні DNS-запити до певного вузла мережі, високий вихідний SMTP-трафік, декілька КС в мережі, що виконують однакові DNS-запити, підозрілі вихідні повідомлення тощо [201].

Відомі методи виявлення для ідентифікації різного виду ЗПЗ використовують набори структурних особливостей виконуваних файлів, мережного трафіку, вміст оперативної пам’яті, значення ключів системного реєстру, тощо. Фактично даний критерій визначає особливості (або ознаки), які вибираються для формування сигнатури або евристичних правил.

Важливою характеристикою методів виявлення ЗПЗ є спосіб отримання даних про досліджуваний об’єкт. Всі методи аналізу можна розділити на дві групи: статичні та динамічні. Статичні методи аналізу досліджують вихідний зловмисний код без його виконання в реальній чи віртуальній системі та включають в себе пошук і виокремлення поведінкових властивостей про виконуваний файл. Статичний аналіз передбачає використання наступних підходів: дослідження структури виконуваних файлів (наприклад інформацію про компіляцію виконуваного файлу, експортовані та імпортовані бібліотеки), вилучення рядків та повідомлень, що можуть ідентифікувати ЗПЗ, дослідження відбитку ЗПЗ (fingerprinting), що включає формування хешу, визначення ознак навколишнього виконання, рядків реєстру, тощо. Також, одним із найпоширеніших підходів статичного аналізу є дизасемблювання виконуваного файлу. Процес дизасемблювання передбачає виконання процедури реверсінженерінгу та перетворення виконуваного файлу у низькорівневий набір інструкцій для пошуку закономірностей та зв’язків. Перевагами використання даного підходу є висока швидкість та низьке ресурсоспоживання (у порівнянні з динамічними методами аналізу). Проте, використання статичного аналізу не дозволяє в повній мірі здійснювати виявлення ЗПЗ, що змінює власну структуру в процесі виконання.

На противагу статичним динамічні методи аналізу передбачають виконання досліджуваного виконуваного файлу з метою отримання знань про поведінку ЗПЗ. Переважно, при використанні динамічних методів аналізу залучають віртуальні машини, “пісочниці” або ресурси, що представляють собою приманку для ЗПЗ (honeypot). В процесі дослідження виконуваного файлу можуть бути виявлені атрибути, що проявляються при виконанні, зокрема, створення файлів, ключів реєстру, відкриття/закриття системних портів, створення мютексів та ін.

Після отримання даних або ознак про ЗПЗ наступним етапом є обробка цих даних з використанням алгоритмів прийняття рішень. Взагалі алгоритми прийняття рішень можна розділити на дві категорії: методи засновані на

експертних знаннях та методи машинного навчання.

Методи, що засновані на експертних знаннях використовуються для формалізації знань про ЗПЗ та знань спеціалістів в області кібербезпеки. Система, що побудована на основі методів експертної оцінки, крім виконання обчислювальних операцій, формує висновки, що засновані на базі знань та продукційних правилах. Знання можуть бути пов’язані наприклад, з тим, які дії є зловмисними (поведінковий аналіз), або які особливості структури свідчать про зловмисність виконуваного файлу (структурний аналіз). Прикладами методів на основі експертної оцінки є такі: метод продукційних правил, в основі яких закладено причинно-наслідкові зв’язки у вигляді конструкцій “якщо-то”; нейромережні методи, тобто методи в основу яких закладено продукційні правила та нечіткий логічний висновок, що дозволяє визначати комплексні ознаки, в той час як елементи штучних нейронних мереж дозволяють адаптувати правила під відоме ЗПЗ. Методи на основі експертної оцінки характеризуються високою достовірністю виявлення відомих зразків ЗПЗ, проте вони є не досить ефективними для ЗПЗ, що застосовує нові техніки [267] протидії антивірусному програмному забезпеченню.

Основне завдання методів машинного навчання полягає у встановленні залежності між ознаками зловмисності та набором досліджуваних даних. Основна ідея будь-якої задачі машинного навчання полягає в тому, щоб навчити модель на основі алгоритму машинного навчання для виконання завдань класифікації, кластеризації, регресії тощо. Навчання, що проводиться на основі вхідного набору даних і моделі, що будується, згодом використовується для прогнозування. Вихід такої моделі залежить від початкового завдання та реалізації. Одними із найпоширеніших алгоритмів машинного навчання для виявлення ЗПЗ є: метод опорних векторів, C3.5, наївний баєсів класифікатор, дерева прийняття рішень, метод найближчого сусіда, тощо.

Метою здійснення задачі виявлення або ідентифікації ЗПЗ є встановлення факту зловмисності програмного забезпечення, тобто очікуваного результату. За ознакою результату, що формується після виконання процесу дослідження ЗПЗ

на предмет зловмисності, всі методи виявлення можна розділити на точні та наближені. Методи, що передбачають формування точного висновку, оперують бінарною множиною – зловмисне або не зловмисне програмне забезпечення. Як правило, точні методи дозволяють здійснювати виявлення визначеного класу ЗПЗ або програмного забезпечення, яке використовує тільки деякі із технологій, що застосовуються для маскування або зміни зловмисного коду.

Наближені методи виявлення ЗПЗ при визначені факту зловмисності використовують терміни нечіткої логіки, тобто при формуванні висновку можуть бути задіяні такі категорії: висока, середня, низька підозрілість (зловмисність).

Для здійснення дослідження методів виявлення ЗПЗ, що представлене зловмисними програмами, та визначення подальших перспективних стратегій розробки нових методів використаємо розроблену класифікацію, зображену на рис. 1.3. Розглянемо детальніше кожен з цих методів.

У роботі [130, 131] запропоновано систему виявлення атак нульового дня, що працює в режимі реального часу та передбачає аналіз мережного трафіку. Запропонована система поєднує в собі використання трьох складових для виявлення ЗПЗ: виявлення аномалій, сигнатурний та евристичний аналіз. Архітектура системи побудована з використанням трьох шарів, де кожен шар відповідає за одну складову та працює паралельно з усіма рештою. В якості алгоритму прийняття рішення застосовується метод опорних векторів.

Статичний підхід, що заснований на відстеженні API викликів представлено в роботі [219]. В якості наборів даних використовуються комбінація множин корисних застосунків та ЗПЗ. Кожен набір даних це API виклики, що отримуються з виконуваних файлів формату PE EXE з міткою класу, до якого вони належать. Для виконання класифікації застосовуються дерева прийняття рішень.

Інший підхід для виявлення ЗПЗ, зокрема поліморфних вірусів, полягає у представлені сигнатури вірусу у вигляді множини графів потоку керування (control flow graph) та пошуку подібності між ними [51]. Пошук схожості

здійснюється на основі метрики відстані між векторами ознак, що представляються k-підграфами фіксованого розміру.

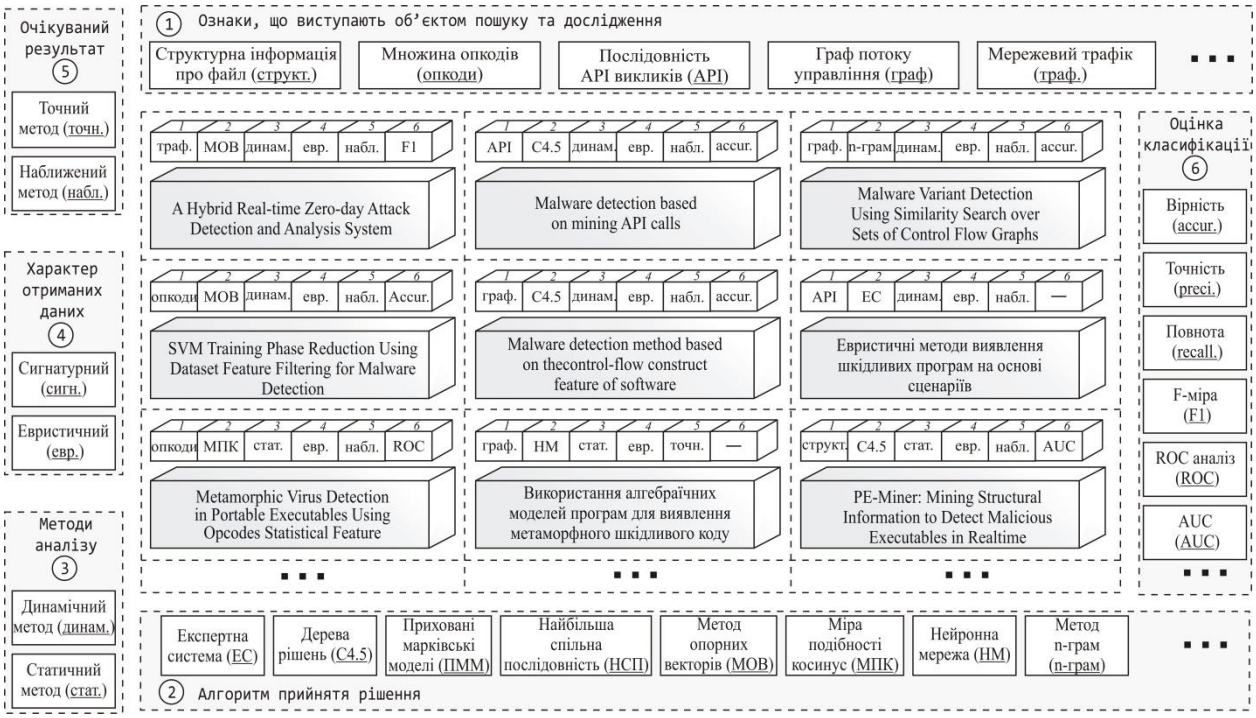


Рис. 1.3. Класифікація методів виявлення файлового ЗПЗ

У роботі [188] автори здійснили аналіз щільності опкодів з використання методу опорних векторів. Для формування вектору ознак було використано віртуальне захищене середовище, в якому кожен тестовий зразок ЗПЗ виконувався протягом трьох хвилин. З метою зменшення розмірності вектору ознак залучено метод головних компонент.

В [235] представлено метод, що базується на аналізі структури виконуваного файлу, зокрема секції заголовку, для виявлення нового ЗПЗ в режимі реального часу. В результаті дослідження було отримано 189 ознак для ідентифікації ЗПЗ. Для здійснення класифікації було використано методи машинного навчання: С3.5, наївний баєсів класифікатор та дерева прийняття рішень.

В роботі [357] представлено підхід для виявлення ЗПЗ з метаморфним навантаженням на основі перевірки еквівалентності в алгебраїчних моделях

послідовних програм та здійснюється оцінка стійкості деяких обфускуючих перетворень. Авторами на основі апарату абстрактних алгебр доводиться, що пошук еквівалентності для програм, які використовують техніки переключенням режиму і використанням зворотніх операторів та їх комбінацій є мало ефективним та відносить задачу пошуку еквівалентності до класу NP повних задач.

Метод, що базується на побудові і використанні сценаріїв для виявлення ЗПЗ, представлено у роботі [360]. Запропоновані сценарії на основі API викликів, що здійснює програма, дозволяють представити поведінку зловмисних програм в ієрархічному вигляді Розроблено архітектуру евристика, що містить експертну систему на основі сценаріїв.

Представлені методи володіють рядом недоліків, які проявляються у високому рівні хибних спрацювань, спрощенням та наближенням тестових даних, орієнтацію на конкретні класи та технології, що використовує ЗПЗ, що не дозволяє повністю вирішувати проблему виявлення нового ЗПЗ, тощо. Тому, при розробці нових методів та підходів до виявлення ЗПЗ доцільно враховувати недоліки відомих методів та використовувати комбіновані ознаки.

При розробці нових методів та алгоритмів виявлення ЗПЗ, в результаті яких здійснюється віднесення досліджуваного об’єкта до одного із класів зловмисних чи корисних програм потрібно виконувати оцінку проведеної класифікації. Процес оцінки класифікації заснований на використанні тестової вибірки, що складається з множини пар “набір характеристик – клас, що відповідає цим характеристикам”. При наявності тестової вибірки перевірка роботи розробленого методу зводиться до встановлення зв'язку його рішення з відомим правильним рішенням. Використання методики ROC-аналізу, яка є загальноприйнятою, дозволяє здійснити оцінку класифікації та достовірності виявлення ЗПЗ за результатами проведених експериментів.

В результаті виконання дослідження здійснено аналіз існуючих критеріїв та класифікацій методів виявлення зловмисного програмного забезпечення. З урахуванням недоліків відомих класифікації виокремлено критерії класифікації,

зокрема характер отриманих даних, ознаки, що виступають об’єктом пошуку та дослідження, методи аналізу, алгоритми прийняття рішення, очікуваний результат та оцінка класифікації. На основі запропонованих критеріїв оцінки методів здійснено класифікацію методів виявлення зловмисного програмного забезпечення. Запропонована класифікація в [372] може бути використана при розробці нових методів та підходів для виявлення ЗПЗ і є основою для визначення напрямку подальших досліджень.

1.4. Розподілені системи як засіб протидії зловмисному програмному забезпеченню

До найбільш складнішого ЗПЗ за будовою відносять бот-мережі, які реалізовано у вигляді розподілених систем. Тобто, частина сучасного ЗПЗ побудована на основі теорії розподілених систем [10, 46, 100, 117, 123, 177, 182, 184, 306, 313, 331]. Крім такої особливості, в процесі вивлення ЗПЗ для організацій і підприємств [385] в їх локальних мережах необхідним є охоплення групи КС для здійснення протидії ЗПЗ, що вимагає розробки засобів виявлення, які б узгоджували взаємодію і використовували ресурси різних КС локальної мережі. Тому, перспективним напрямком вирішення проблеми виявлення ЗПЗ для організацій та підприємств є створення розподілених систем виявлення.

Розглянемо основні поняття теорії розподілених систем. Процес опрацювання інформації є невід’ємною складовою будь-якої системи, тобто певної єдиної форми організації множини елементів, що знаходяться у відносинах та зв'язках один з одним, а також певний порядок в розташуванні цих елементів та зв'язків частин таких елементів. Наприклад, в автоматизованих системах з CAN-архітектурою [361] та системах IoT основою їх функціонування є технології розподілених систем і взаємозв’язок між ними встановлюється переважно оптимізовано.

Будь-яка система може перебувати у двох формах – у формі концентрації або у формі розподілу. Зокрема, це стосується і комп’ютерних систем. Форма

концентрації в системах передбачає централізацію. При цьому здійснюється процес, при якому елементи системи спрямовані від "навколишнього простору" до "єдиного центру". У свою чергу при розсіюванні (розподілі або децентралізації) відбувається зворотний процес – "від центру" до "оточуючого простору". Схематичне зображення централізованих та децентралізованих систем наведено на рис. 1.4.

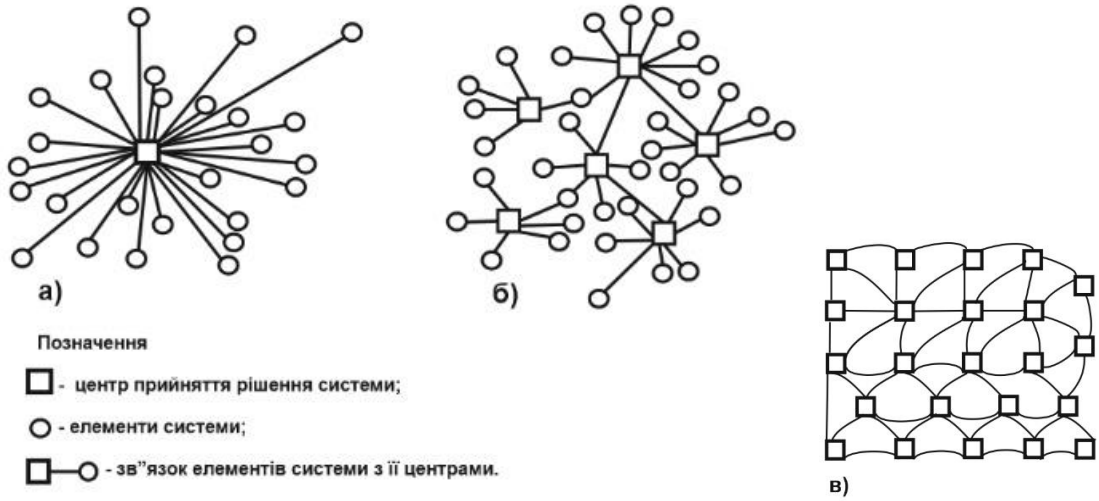


Рис. 1.4. Схематичне зображення систем: а) централізованої;
б) децентралізованої; в) розподіленої

На рис. 1.4а наведено схематичне зображення централізованої системи з одним центром управління (контролю, обробки), в той час як на рис. 1.4б) та рис. 1.4в) – децентралізовані системи. Різниця між схемою на рис. 1.4б) і рис. 1.4в) лише в типі розподілу процесів. В таких системах у випадку атаки на будь-яку точку (самостійний центр) усталеність функціонування системи загалом не зміниться. Такий стан зберігається, поки під контролем знаходиться більше 50% центрів. У випадку атаки на центральний елемент системи (рис. 1.4а) зловмисник отримує контроль над всією мережею елементів.

Централізація в системі – це форма організації системи, при якій є як мінімум два рівні елементів, між якими існує нерівноправна взаємодія. У такій системі завжди є ієрархія, коли "головний" елемент володіє більшим рівнем впливу, ніж інші елементи. Такі системи мають велику швидкість реакції на дії

"центру".

Децентралізація [314] – це процес перерозподілу, розсіювання функцій, та ресурсів або інших об'єктів від монопольного центру, при якому існує безліч локальних, віддалених, самостійних і рівноправних центрів. У свою чергу ці центри є одиницями, які не зважаючи на власну самостійність і неспроможність поодинці здійснити прийняття рішення в результаті всі разом створюють стійку систему.

Розподілена система – система, в якій обробка інформації зосереджена не в одній обчислювальній машині, а розподілена між декількома комп'ютерами.

Характеристиками розподілених систем є такі [314]:

1. Спільне використання ресурсів. Розподілені системи допускають спільне використання як апаратних (жорстких дисків, принтерів), так і програмних (файлів, компіляторів) ресурсів.

2. Відкритість. Це можливість розширення системи шляхом додавання нових ресурсів.

3. Паралельність. У розподілених системах кілька процесів можуть одночасно виконуватися в різних комп'ютерах в локальній мережі. Ці процеси можуть взаємодіяти під час їх виконання.

4. Масштабованість. Під масштабованість розуміють можливість додавання нових засобів і методів, а також додавання нових елементів системи.

5. Відмовостійкість. Наявність декількох комп'ютерів в системі дозволяє здійснювати дублювання інформації і підтримує стійкість до деяких апаратних і програмних помилок. Розподілені системи в разі помилки можуть підтримувати часткову функціональність. Повний збій в роботі системи відбувається тільки при збоях в мережі.

6. Прозорість. Користувачам надається повний доступ до ресурсів в системі, в той же час від них прихована інформація про розподіл ресурсів в системі.

Концептуально розподілені системи будуються пошарово [314]: презентаційний шар, шар прикладної логіки і шар керування ресурсами. Ці шари

можуть бути тільки абстракціями, що існують лише на етапі проектування, але можуть бути чітко видимі в програмному забезпеченні, коли їх реалізують у вигляді окремих підсистем, іноді із застосуванням різного інструментарію.

Всі розподілені системи повинні спілкуватися із зовнішнім середовищем. Значна частина цієї взаємодії пов'язана з перетворенням інформації та поданням її для зовнішніх користувачів, підготовкою запитів і отриманням відповідей. Компоненти розподіленої системи, що забезпечують цю діяльність, формують презентаційний шар.

Всі розподілені системи мають клієнтів, тобто сутності, які користуються сервісами, наданими цими системами. Клієнти можуть бути повністю зовнішніми по відношенню до систем і незалежними від них. У цьому випадку вони не є презентаційним шаром самих систем.

На практиці досить часто виникає подія при якій клієнт і презентаційний шар виконують однакові функції. Це типово для систем типу клієнт / сервер, в яких є програма, яка одночасно виконує роль клієнта і презентаційного шару.

Програмне забезпечення системи не тільки відображає інформацію, а й здійснює обробку даних. Ця обробка проводиться програмою, що реалізує фактичні операції, запитані клієнтом за посередництвом презентаційного шару, тобто програмою шару прикладної логіки. Залежно від складності виконуваної логіки цей шар може називатися бізнес-процесом, бізнес-логікою, бізнес-правилами або сервером.

Для роботи будь-яке програмне забезпечення системи потребує даних, які можуть розміщуватися в базах даних, файлових системах та інших репозиторіях. Програми шару управління ресурсами об'єднують всі такі елементи. Цей підхід, однак, має обмеження, оскільки він концентрується тільки на аспекті управління даними. Однак, управління потребують всі зовнішні системи, які постачають інформацію. До них входять не тільки бази даних, а й інші розподілені системи зі своїми шарами: презентаційним, прикладним і управління ресурсами. При цьому з'являється можливість рекурсивно будувати розподілені системи, що складаються з інших систем, як з компонентів.

Описані три шари – це концептуальні конструкції, які логічно поділяють функціональність більшості розподілених систем. У практичних реалізаціях вони можуть комбінуватися різними способами, створюючи при цьому різнотипні розподілені системи.

Однією із форм розподілених обчислень є грід-обчислення.

Грід-обчислення – це форма розподілених обчислень, в якій віртуальний суперкомп’ютер представлений у вигляді кластерів, з'єднаних за допомогою мережі, слабозв’язанних гетерогенних комп'ютерів, що працюють разом для виконання великої кількості завдань. Ця технологія застосовується для вирішення наукових, математичних задач, що вимагають значних обчислювальних ресурсів. Грід-обчислення використовуються також в комерційній інфраструктурі для вирішення таких трудомістких завдань, як економічне прогнозування, сейсмоаналіз, розробка та вивчення властивостей нових ліків.

Грід з точки зору мережної організації є узгодженим, відкритим і стандартизованим середовищем, яке забезпечує гнучкий, безпечний, скоординований розподіл обчислювальних ресурсів і ресурсів зберігання інформації, які є частиною цього середовища, в рамках однієї віртуальної організації.

Грід-обчислення можна організувати на базі множини застарілих моделей персональних комп'ютерів, що об’єднанні в ієрархічну локальну обчислювальну мережу Ethernet з присутністю серверів. Ця мережа може мати з'єднання з інтернетом.

В основі грід-систем лежить забезпечення стабільної роботи набору служб на основі прийнятих відкритих стандартів і керуючого програмного забезпечення для забезпечення надійного, уніфікованого доступу до географічно розподілених інформаційним і обчислювальних ресурсів, що включає окремі комп’ютери, кластери і суперкомп'ютерні центри, сховища інформації тощо (рис. 1.5).

Рис. 1.5. Узагальнена архітектура грід-системи

На сьогоднішній день відомими грід-системами є Globus Toolkit [44, 133], Legion [133] та Crisis [114].

Globus Toolkit є динамічною грід-системою та передбачає використання мінливих в часі "областей довіри". Рішення динамічного виставлення областей довіри і динамічного створення сутностей заснована на ідеї видачі сертифікатів. Тобто, політика безпеки в Globus Toolkit [44, 133] заснована на використанні криптосистем з відкритими ключами. Видача прав пов'язана з перевіркою сертифікатів. Для того, щоб довіряти пред'явнику сертифіката, потрібно довіряти лише центру видачі сертифікатів.

Legion [133] – це розподілена обчислювальна платформа для комбінування множини незалежних робочих станцій в одну систему. Всі ресурси, включаючи обчислювальні потужності, бази даних, Legion об'єднує, використовуючи один об'єктно-орієнтований мета-комп'ютер. Модулі забезпечення безпеки в цій платформі входять в Legion Runtime Library (бібліотека часу виконання), яка визначає необхідні базові об'єкти, такі як об'єкти ядер, хостів і сховищ.

Crisis [114] це одна з компонент WebOS. WebOS – це застосунок, основною метою якого є підтримка мережних застосунків під різні операційні системи. CRISIS доводиться вирішувати питання безпеки такі як аутентификація та авторизація користувачів в системі. CRISIS це система, яка базується на основі

подій і включає два основних компоненти для авторизації і аутентифікації. Перший це менеджер процесів, який приймає запити (логін або доступ до ресурсу) і менеджер безпеки, який зберігає особисті дані суб'єктів і приймає рішення за отриманими запитами.

Відомі рішення та реалізації типових розподілених систем не можуть бути використані при створення розподілених систем виявлення ЗПЗ, бо інформація про них є відкритою і зловмисник використовуватиме особливості побудови таких систем. Тому, необхідним є розробка теорії і практики створення розподілених систем виявлення ЗПЗ, які б враховували специфіку предметної області.

Таким чином, для ефективного здійснення процесу виявлення зловмисного програмного забезпечення необхідним є побудова системи, яка була б заснована на принципі децентралізації обчислювальних ресурсів та використовувала основні концепції розподілених обчислень. Реалізація такої системи дозволить увібрати в собі основні переваги розподілених обчислень [173, 184, 200, 227, 364, 398, 399], що проявляються у високій надійності системи (відсутність окресленого центру прийняття рішень дозволить усунути можливість атаки на нього), масштабованості, модульності та охопить процесом виявлення групу КС.

1.5. Визначення напрямів досліджень

Для вирішення проблеми покращення ефективності виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах необхідно здійснити розроблення теорії і практики створення розподілених систем на основі принципів децентралізації та самоорганізації і розв’язати такі наукові задачі:

1) дослідити особливості функціонування зловмисного програмного забезпечення в локальних комп’ютерних мережах та проаналізувати ефективність роботи сучасних розподілених систем виявлення, а також їх архітектуру і компоненти;

2) удосконалити модель архітектури розподіленої системи виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах, в якій синтезувати вимоги розподіленості, децентралізованості, багаторівневості та самоорганізованості, для створення на її основі розподілених систем та їх компонентів, що функціонуватимуть автономно і самостійно прийматимуть рішення про наявність зловмисного програмного забезпечення та нарощення своїх функціональних можливостей;

3) розробити модель архітектури типових компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення на основі структур Кріпке з представленням компонентів через стани, в яких вони можуть перебувати під час функціонування, для визначення стану безпеки всієї розподіленої системи та її компонентів;

4) розробити метод взаємодії компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення для підтримки її цілісності, визначення порядку передачі знань між її компонентами і використання встановлених аналітичних залежностей між рівнями безпеки програмних модулів та рівнем безпеки всієї розподіленої багаторівневої системи, на основі якого система змогла б автономно змінювати свою архітектуру та функції без втручання користувача, а також визначати стратегію своєї подальшої роботи;

5) удосконалити моделі зловмисного програмного забезпечення шляхом їх подання алгебрами поведінки, що дозволило б створити базис поведінкових сигнатур, врахувати особливості функціонування зловмисного програмного забезпечення в локальних комп’ютерних мережах та здійснити його класифікацію за типами поведінки;

6) розробити метод виявлення бот-мереж у локальних комп’ютерних мережах, який би включав здійснення активного моніторингу системних подій та організацію узгодженої взаємодії компонентів розподіленої системи при прийнятті рішень, для створення на його основі засобів, здатних інтегруватись в розподілену систему та класифікувати бот-мережі за їх поведінковими

сигнатурами, сформованими закладеними в їх компоненти функціями;

7) розробити метод виявлення файлового зловмисного програмного забезпечення в локальних комп'ютерних мережах, суть якого полягає в поєднанні роботи програмних агентів, що здійснюють виявлення зловмисного програмного забезпечення в окремих комп'ютерних системах, відповідно до імплементованих в них методів: динамічного формування поведінкової сигнатури шляхом відстеження викликів прикладного програмного інтерфейсу, знаходження поліморфного та метаморфного програмного коду, сканування виконуваних програм шляхом створення для них автономних процесів та відповідних програмних агентів у розподіленій системі;

8) розробити метод виявлення файлового зловмисного програмного забезпечення на основі динамічного формування поведінкової сигнатури шляхом відстеження викликів прикладного програмного інтерфейсу, в якому поведінкову сигнатуру формувати на основі критичних викликів прикладного програмного інтерфейсу за групами зловмисної активності з відображенням частоти їх входження та характеру взаємодії критичних функцій, для виявлення нових версій відомого зловмисного програмного забезпечення не тільки за наявністю критичних викликів, але й за їх взаємодією між собою;

9) розробити метод виявлення поліморфних та метаморфних вірусів з використанням функцій заплутування програмного коду на основі поетапного аналізу і порівняння функціональних блоків програмного об'єкта та його змінених версій, отриманих, в тому числі, від різних компонентів розподіленої системи шляхом їх взаємодії між собою;

10) розробити програмне забезпечення розподіленої багаторівневої системи та її апаратно-програмні компоненти захисту інформації для реалізації запропонованих теоретичних основ таких систем, підтвердження можливості їх практичного створення та використання в експериментальних дослідженнях для порівняння з відомими системами виявлення і впровадити розроблену розподілену багаторівневу систему виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах організацій (підприємств).

Висновки до першого розділу

Дослідження функціонування ЗПЗ у локальних комп’ютерних мережах та антивірусних засобів їх виявлення, показало наступні результати:

– розробники ЗПЗ володіють значними технічними і фінансовими засобами для його реалізації;

– ЗПЗ може використовувати різносторонні засоби для поширення, що підвищує його життєздатність і можливості для поширення та ускладнює виявлення антивірусними засобами;

– певні типи ЗПЗ мають модульну структуру, що знижує достовірність виявлення відомими антивірусними засобами;

– застосування стандартних методів (сигнатурний аналіз, метод контрольних сум тощо) не гарантує достовірного виявлення ЗПЗ у КС у зв’язку з необхідністю постійного оновлення існуючих баз вже відомого ЗПЗ;

– використання модулів шифрування в ЗПЗ не дозволяє застосовувати сигнатурний аналізатор та ускладнює виявлення через створення різних копій одного і того ж ЗПЗ;

– застосування сучасних евристичних аналізаторів вимагає значних ресурсів КС, має невисоку швидкодію та не гарантує належну достовірність результатів виявлення ЗПЗ у КС і, при цьому, можливі хибні спрацювання, кількість яких збільшується при встановленні високого рівня підозрілості в налаштуваннях системи;

– використання методів на основі контрольних сум не завжди дає однозначну відповідь стосовно того, чи відбулося інфікування КС;

– використання мережних систем виявлення ЗПЗ для покращення ефективності виявлення, через залучення адміністратора мережі до прийняття рішення, знижує оперативність в прийнятті рішення;

– відомі мережні системи побудовані переважно з використанням централізованої архітектури, що активізує зловмисників до виявлення центру для зупинки системи;

– відомі мережні системи виявлення і антивірусні засоби переважно є хост-орієнтованими і не враховують можливостей ЗПЗ виконуватись в декількох КС одночасно.

Відомі антивірусні засоби та системи виявлення ЗПЗ не забезпечують його повного виявлення. В КС залишається певний відсоток невиявленого ЗПЗ. Враховуючи актуальність виявлення цього ЗПЗ для підприємств і організацій, необхідна розробка нових методів та засобів виявлення ЗПЗ в локальних комп’ютерних мережах, особливо для нового ЗПЗ. З метою покращення ефективності виявлення ЗПЗ у локальних комп’ютерних мережах актуальним є розроблення теорії і практики створення розподілених систем виявлення ЗПЗ.

Основні результати розділу опубліковані у працях [372, 373, 369, 390, 362, 225, 378, 350-352].

РОЗДІЛ 2
РОЗПОДІЛЕНА БАГАТОРІВНЕВА СИСТЕМА ВИЯВЛЕННЯ ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ У ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ

Проведені дослідження відомих антивірусних методів та засобів вказують, що реалізація нових принципів, моделей та методів виявлення конкретних типів зловмисного програмного забезпечення шляхом створення відповідних систем виявлення ЗПЗ потребує подальшого розвитку, але досягнення підвищення достовірності виявлення нового та відомого ЗПЗ, в якому застосовуються комбіновані технології зловмисного проникнення та поширення, можливе також за умови комбінованого використання нових та відомих методів виявлення. Для того, щоб ефективно застосовувати методи та засоби виявлення зловмисного програмного забезпечення необхідно розробити систему [173, 228, 363, 364, 367, 385], яка б включала в себе достатню кількість реалізованих ефективних методів у вигляді відповідних підсистем, мала можливість до нарощування та враховувала б майбутні тенденції розвитку як антивірусних засобів, так і зловмисного програмного забезпечення.

2.1. Модель та архітектура розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення

Сучасне зловмисне програмне забезпечення представляє собою складні багатофункційні програмні системи та комплекси, які побудовані з використанням ефективних методів створення програмних засобів та методів поширення зловмисного коду. Для організації ефективної протидії таким засобам важливим є розробка системи виявлення ЗПЗ, архітектура якої була б подібною до архітектури складного комплексу ЗПЗ, але мала б більшу функціональність. Крім того, досягнення підвищення достовірності виявлення ЗПЗ в межах тільки однієї КС, яка має вихід в мережу Internet, може бути

недостатнім при протидії засобам ЗПЗ, які представлені великим програмним комплексом, що розміщений в багатьох КС в глобальній мережі. Тому, покращення ефективності виявлення ЗПЗ можливе за рахунок залучення груп КС порівняно з однією КС. В якості таких груп можуть виступати КС локальних комп'ютерних мереж (ЛКМ). Відповідно важливим є здійснення побудови таких ефективних систем для їх використання в локальних комп'ютерних мережах, де уможливлюється встановлення таких систем в усіх комп'ютерних системах мережі, а не локально. Це дозволить при проведенні аналізу зібраних типових даних з різних КС підвищити достовірність виявлення.

Виявлення зловмисного програмного забезпечення здійснюється за допомогою різноманітних засобів. Ефективність та достовірність виявлення суттєво залежать, зокрема, і від архітектури таких засобів, а також їх позиціонування та місця розміщення в локальних комп'ютерних мережах. Враховуючи, що процес виявлення ЗПЗ проводитиметься в локальних мережах, то вибір моделі функціонування системи повинен передбачати залучення інформації зі всіх комп'ютерних систем локальної мережі, тобто розміщення в всіх КС цієї системи. Це необхідно для підвищення ефективності і достовірності виявлення за рахунок врахування інформації про стан з інших КС для прийняття рішення в конкретній КС. Ці основні вимоги, що система повинна бути розміщена в мережі в кожній КС, впливають на вибір моделі її архітектури. Також, важливим для таких систем є те, щоб центр прийняття рішень системи не представлявся та ідентифікувався однозначно, бо його виявлення призведе до атаки на нього для виведення всієї системи з робочого стану. Система [380, 385] повинна бути побудована так, щоб розміщені в КС в мережі її компоненти ефективно спілкувалися між собою для обміну інформацією про стан КС з метою надання додаткової інформації для прийняття рішення. Крім того, система виявлення ЗПЗ повинна відповідним чином структуруватись, щоб мати можливість нарощуватись і її збільшення не повинно сповільнювати процес виявлення. Таким чином, сучасна система виявлення ЗПЗ в локальних комп'ютерних мережах повинна мати такі характеристики [172]:

1) розподіленість в просторі в локальній мережі у всіх КС;

2) формування єдиної системи всіма частинами, розміщеним в КС;

3) децентралізованість структури для уникнення виведення системи з ладу за умови виявлення єдиного центру прийняття рішень чи управління системою;

4) можливість прийняття рішення модулем, розміщеним в певній КС, за підтримки інших структурних частин системи;

5) багаторівневість для чіткого відокремлення різних за призначенням функціональних частин і можливості нарощування;

6) здатність до самоорганізованості при вирішенні завдань та за умови порушення цілісності системи, під час атак на інформаційно-комунікаційні ресурси мережі або ураженні системи ЗПЗ;

7) динамічне формування архітектури з наявних частин робочих КС мережі;

8) адаптивність до середовищ КС мережі та специфіки задач, вирішуваних конкретними КС;

9) можливість автоматизованої підтримки адміністратором мережі для налаштування роботи та отримання звітності;

10) автономність системи у прийнятті рішень;

11) можливість здійснення самоконтролю в роботі компонентами системи для уникнення перевантаження КС задачами розподіленої системи;

12) відповідність політиці безпеки підприємства (організації).

Оскільки, визначальними характеристиками системи буде розподіленість та багаторівневість, то назвемо її розподіленою багаторівневою системою (РБС). Основними функціями РБС є перевірка наявного програмного забезпечення та запущених процесів в КС локальної мережі на можливість віднесення до зловмисного програмного забезпечення.

Досягнення відповідності системи заданим характеристикам та покладеним на неї функціям з виявлення ЗПЗ формують вимоги до неї, основними з яких будуть такі:

1) розподіленість;

2) децентралізованість;

3) самостійність у прийнятті рішень;

4) багаторівневість;

5) самоорганізованість;

6) адаптивність.

Згідно аналізу поставленої задачі, сформованих вимог до системи виявлення ЗПЗ, на основі функцій системи та її характеристик архітектуру розроблюваної системи можна синтезувати у вигляді сукупності наступних компонентів: моделей колективного інтелекту, багатоагентних систем, розподілених систем, децентралізованих систем, самоорганізованих та адаптивних систем. Врахування таких складових в моделі системи є основою її архітектури і підвищить надійність функціонування та живучість системи в локальній комп'ютерній мережі.

Виділимо особливості кожної моделі і інтегруємо їх в модель розроблюваної системи. Архітектура розроблюваної системи представлена узагальненою схемою [173, 364] основних складових на рис. 2.1.

Враховуючи, що процес виявлення ЗПЗ відбуватиметься в локальній комп'ютерній мережі, тоді спочатку представимо (рис. 2.2) місце поширення і виявлення ЗПЗ, яке в глобальній мережі поширюється постійно і неконтрольовано. А також, зобразимо модель [380], яка відображає розподілену архітектуру системи (рис. 2.3). Поетапно ця розподілена архітектура буде наповнюватись підсистемами, в яких реалізовуватимуться інші моделі, що входять до системи.

Введемо позначення для досліджуваної розподіленої багаторівневої системи і позначимо її символом А, структурні компоненти системи – A_i, i=1,…,n, де n - кількість програмних модулів. Тоді, представлення системи через її структурні компоненти здійснимо за формулою (2.1) так:

$$A = \{A_1, \dots, A_n\} \tag{2.1}$$

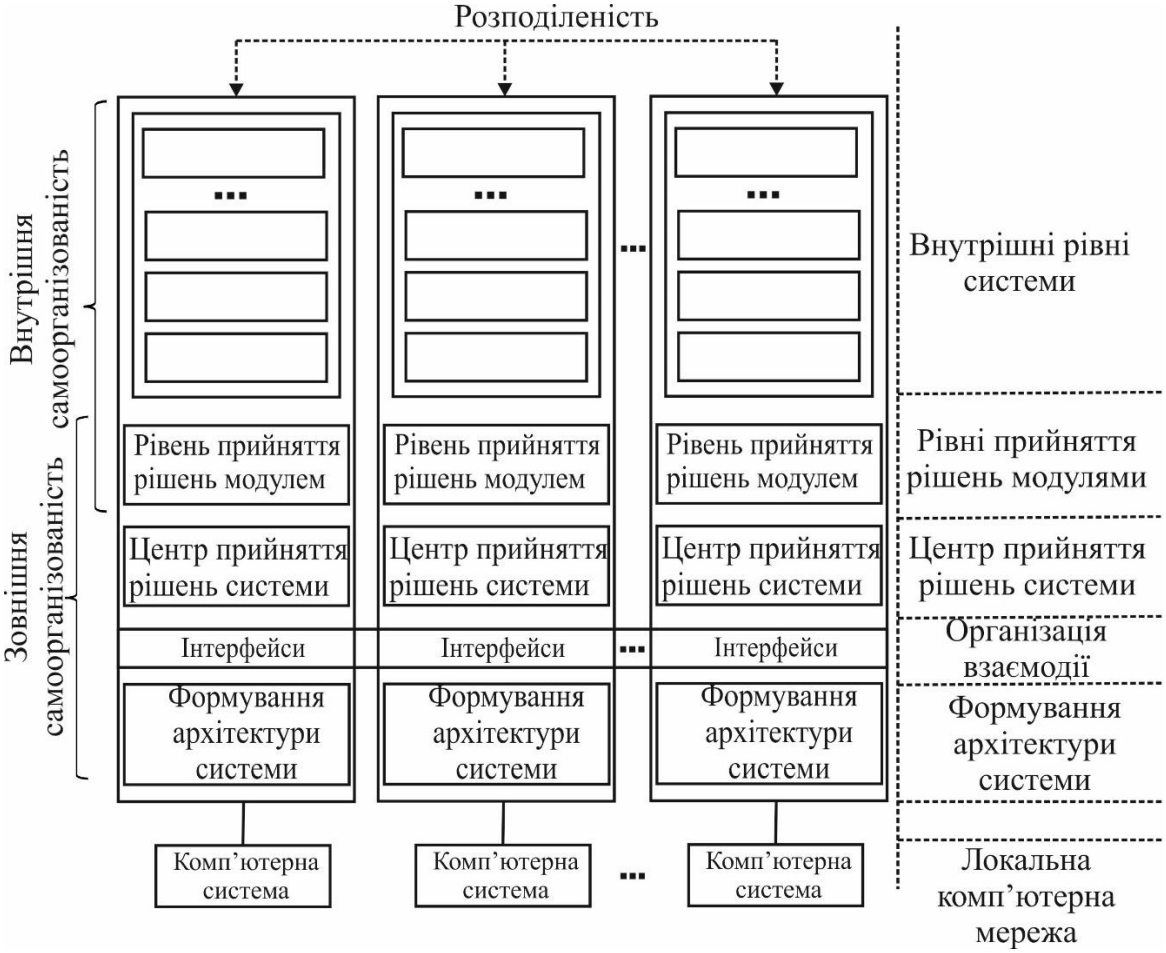


Рис. 2.1. Узагальнена схема взаємозв'язку основних компонентів в архітектурі системи

Рис. 2.2. Модель поширення ЗПЗ в фрагментах глобальної і локальної комп'ютерних мереж

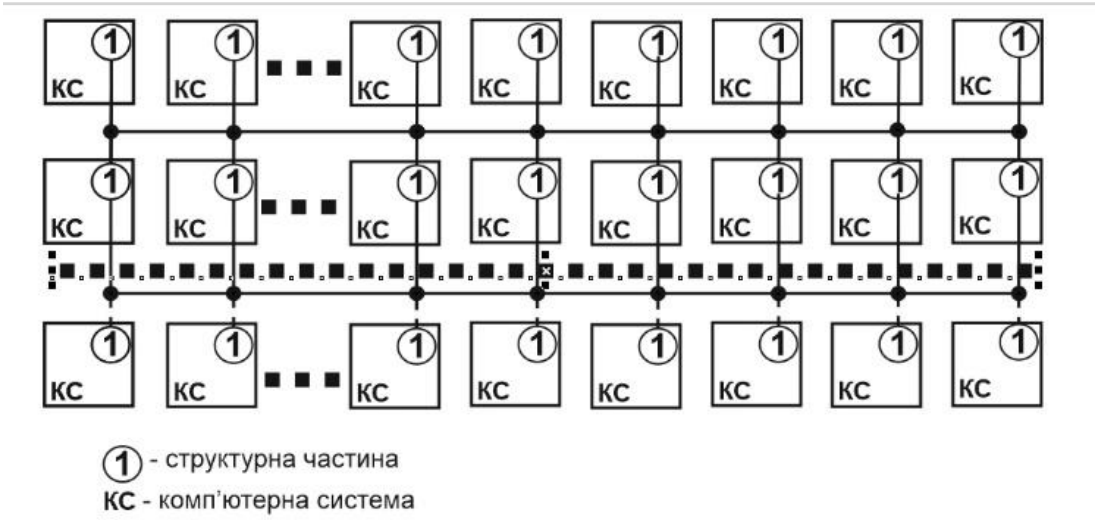


Рис. 2.3. Структура системи з врахуванням моделі розподіленої архітектури

Такий формалізований опис надалі бути використаний для оптимізації зв'язків та залежностей між елементами системи та підсистем. Елементи A_i однакові за структурою, але в процесі функціонування системи матимуть різне наповнення та можуть перебувати в різних станах.

РБС розподілена в просторі і згідно характеристичних вимог повинна бути децентралізованою, тобто система не повинна мати єдиного центру керування всіма її частинами та можливості прийняття рішень в залежності від зміни стану будь-якої КС мережі. В цьому контексті децентралізованість і самостійність прийняття рішень модулем системи розміщеним в певній КС не ототожнюються. В поняття децентралізованості системи закладемо функцію вищого рівня абстракції, завданням якої повинна бути здатність системи конкретної КС приймати рішення про початок її виконання (тобто активацію закладених функціоналів), здійснення переходу між визначеними рівнями на основі інформації отриманої з інших рівнів системи, прийняття рішення про комунікацію з іншими частинами всієї системи, отримання інформації від інших частин системи з різних КС та передача цієї інформації на відповідні рівні, завершення роботи. Для компоненти системи розміщеної в КС введемо поняття

програмного модуля системи. Самостійність в прийнятті рішення програмним модулем системи включає такі можливості: прийняття рішення про стан загрози ЗПЗ для КС на основі інтегрованої інформації з інших рівнів програмного модуля і передача цього значення на рівень децентралізації; передача на рівень децентралізації значення рівня безпеки; визначення кількості та залучення засобів відповідних рівнів програмного модуля для дослідження ЗПЗ. Таким чином, рівень децентралізації відрізнятиметься від рівня прийняття рішень тим, що на його рівні приймаються рішення про загальну організацію функціонування програмного модуля в структурі всієї системи, а на рівні прийняття рішень – про безпосереднє виконання саме основних задач дослідження ЗПЗ. На рис. 2.4 зображена взаємодія цих двох рівні та їх основні функції.

Рис. 2.4. Модель взаємодії основних функцій рівнів децентралізації та прийняття рішень

Розглянемо детальніше блоки децентралізації та прийняття рішень і їх взаємодію. Початок роботи конкретного модуля розпочинається із запуску конкретної КС, в яку він встановлений, та безпосередньо запуску цього модуля. Спочатку запускається функція стартового підблоку, в якому вирішується питання про початок і послідовність виконання функцій інших блоків програмного модуля. Першим підблоком, в який передається вказівка про

запуск, є підблок, який визначає стан загрози КС від ЗПЗ при старті. Для цього він передає вказівку на запуск внутрішніх підблоків блоку прийняття рішень. Після отримання результатів від них, тобто встановлення рівня безпеки, він передає вказівку про запуск і значення рівня безпеки на підблок, який встановлює програмний модуль на відповідний рівень безпеки та запускає визначені для цього рівня функції, тобто визначає необхідний функціонал для роботи програмного модуля. Після цього він направляє вказівку для запуску модуля, який приймає рішення про зв'язок з іншими програмними модулями всієї системи. Якщо рівень безпеки програмного модуля показує, що КС перебуває в критичному стані безпеки, то рішення про комунікацію з іншими складовими системи буде негативним і при самому надкритичному рівні передається на підблок завершення роботи програмного модуля. Якщо ж рішення позитивне, тоді на наступному кроці формується пакет для розсилки всім іншим компонентам системи і відправляється. Також, цим підблоком збирається та опрацьовується інформація отримана від інших компонентів системи. Узагальнена схема зв’язків підблоків блоків децентралізації та прийняття рішень зображена на рис. 2.5.

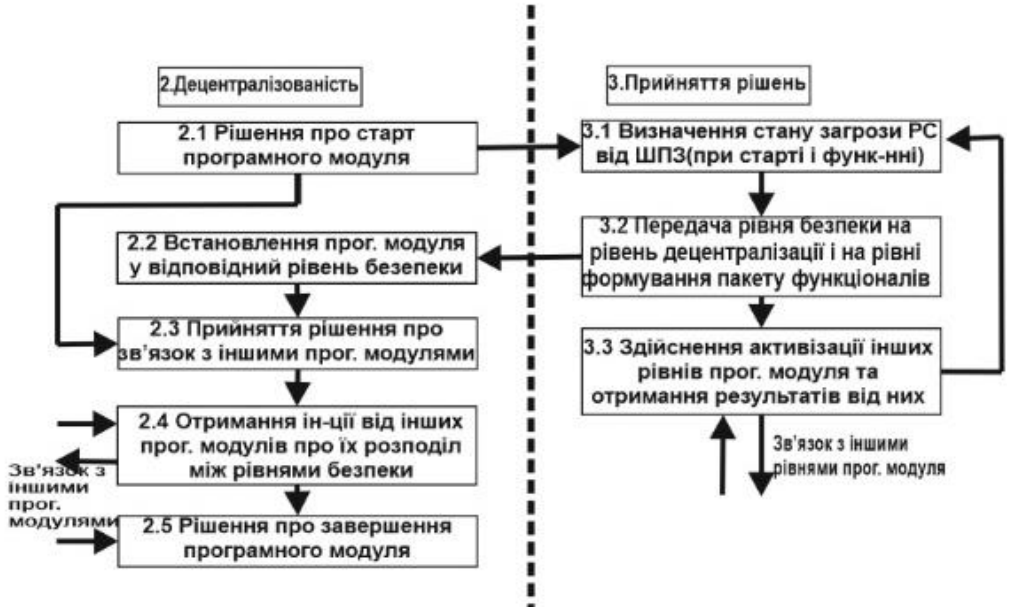


Рис. 2.5. Узагальнена модель взаємодії блоків децентралізації та прийняття рішень

Багаторівневість системи дозволить відділити процеси, які відноситимуться до функціонування системи в цілому, процеси функціонування програмного модуля і розділити з рівнями задачі з виявлення різних типів ЗПЗ та мережних атак і необхідних утиліт для вирішення цих задач. Рівні децентралізації та прийняття рішень є базовими і в багаторівневій системі зображені [173] на рис. 2.6 у взаємодії з таких же рівнів інших програмних модулів РБС. Крім того, багаторівневість дозволить проводити нарощування системи новим функціоналом, а також чітка відокремленість різних за задачами частин дозволить сумісне використання утиліт певних рівнів.

Рис. 2.6. Схема багаторівневості РБС

Функціонал, який відповідатиме за самоорганізованість програмного модуля РБС, розміщено окремим рівнем програмного модуля, який взаємодіятиме з модулем прийняття рішень. Крім того, певні рівні програмних модулів матимуть можливість здійснювати самонавчання та на його основі здійснюватиметься зміна стану безпеки. Самоорганізованість на рівні всієї РБС здійснюватиметься в підблоці блоку децентралізації і включатиме функціонування всієї системи та її переходу в різні стани в залежності від

зовнішніх змін в КС та мережі. Властивість самоорганізованості РБС тісно пов’язана з адаптивністю і проявлятиметься в зміні структури її системи та рівня організації в процесі свого життєвого циклу в результаті накопичення досвіду, збереженої інформації в пам’яті. Накопичений досвід виражатиметься в зміні параметрів, важливих для мети системи, що змінюватиме спосіб функціонування системи, і виражатиме властивість самонавчання.

РБС повинна здійснювати самоконтроль в частині завдань, які виконуватимуться її програмними модулями. Це необхідно для уникнення взаємоблокувань процесів [370, 371, 383] в комп’ютерних системах через завдання, які надходитимуть від її програмних модулів. Крім того, завдання від програмних модулів можуть перевантажувати ресурси КС, тому потрібні засоби в ПМ, які б здійснювали оцінку наближення до таких станів і призупиняли виконання частини процесів ПМ або передавали завдання на виконання іншим ПМ РБС. Для реалізації самоконтролю в РБС необхідним є розробка внутрішнього планувальника завдань в структурі кожного ПМ. Завдання самоконтролю в ПМ є частиною завдань підсистеми РБС, яка відповідає за самоорганізованість.

Формування архітектури РБС в мережі здійснюватиметься на протязі життєвого циклу її використання та передбачатиме зберігання попередніх форматів для здійснення аналізу.

Частина рівнів програмних модулів РБС матиме властивості адаптивності для підвищення ефективності виконання поставлених задач виявлення ЗПЗ. Адаптивність РБС проявлятиметься в автоматичній зміні алгоритмів свого функціонування і за потреби своєї структури з метою збереження або досягнення оптимального стану при зміні зовнішніх умов.

Інтерфейси системи поділено на такі: адміністративний, щоденний звіт, звіт за критичної ситуації, автономний міжмодульний.

Таким чином, розроблено [364, 380, 385] гнучку архітектуру розподіленої багаторівневої системи, яка має можливість розширення за рахунок її нарощення різними функціоналами виявлення ЗПЗ в локальних комп’ютерних мережах, що

дозволяє використовувати таку систему автономно. Дана архітектура системи є концептуальною і потребує деталізації рівнів програмних модулів та організації взаємозв’язку між ними.

2.2. Модель архітектури програмних модулів РБС виявлення зловмисного програмного забезпечення

Досягнення повної функційності РБС потребує розробки її моделі функціонування в локальній комп’ютерній мережі в частині організації взаємодії між її програмними компонентами, розміщеними в мережі, та спілкування між ними.

Розподілена багаторівнева система в локальній комп’ютерній мережі згідно її моделі представлена [173, 364] однаковими програмними модулями, які розміщені в кожній комп’ютерній системі. Тобто, не залежно від того скільки комп’ютерних систем в локальній мережі, кожна з них містить програмний модуль РБС. Якщо в процесі користування комп’ютерними системами виявиться, що не всі вони в мережі ввімкнені, тоді РБС складається з тих, які активні. Програмні модулі активних КС в локальній мережі формують безпосередньо РБС. Це дозволить навіть за наявності роботи двох комп’ютерних систем підтримувати виконання задач системи.

Кожен програмний модуль РБС містить на відповідних рівнях функціонал для виявлення певного типу зловмисного програмного забезпечення або атаки.

РБС містить в кожному програмному модулі рівень, який відповідає за спілкування між програмними модулями всієї системи, тобто забезпечує роботу каналів зв’язку. За допомогою цього рівня встановлюється зв'язок між програмними модулями системи в цілому. Важливим є ідентифікація одним конкретним програмним модулем решти програмних модулів для активації та роботи цілісної системи. Для цього при встановленні програмного в конкретну КС активується функціонал, що здійснює зчитування системної інформації про апаратно-програмне забезпечення КС, зберігає цю інформацію в своєму

внутрішньому сховищі, формує на її основі ідентифікаційну ознаку цього конкретного модуля, зберігає її як ідентифікатор програмного модуля і використовує її в пакетах, які розсилатимуться в інші програмні модулі РБС. При встановленні всіх програмних модулів РБС в мережі здійснюють їх активацію з метою збору всіх ідентифікаційних ознак. Кожен окремий програмний модуль в системі після повного встановлення системи міститиме характеристики та ідентифікатори всіх програмних модулів системи.

Використання такої ідентифікаційної інформації в якості окремого поля пакету дозволить однозначно ідентифікувати та розрізнювати різні КС в локальній мережі, але при цьому віддаленість КС не важлива. Врахування віддаленості відбувається на рівні підтвердження цілісності системи. Узагальнена структура пакету, що надсилається від програмних модулів системи, представлена в табл. 2.1.

Таблиця 2.1

Структура повідомлення в пакеті

Ідентифікатор програмного модуля системи	Інформація про стан програмного модуля

Отримана інформація від програмних модулів системи зберігається та використовується для обробки в структурі представленій в табл. 2.2.

Таблиця 2.2

Інформація про стани програмних модулів

Ідентифікатори програмних модулів	Стан програмних модулів
Ідентифікатор програмного модуля 1 системи	Інформація про стан програмного модуля 1
Ідентифікатор програмного модуля 2 системи	Інформація про стан програмного модуля 2
…	…
Ідентифікатор програмного модуля N системи	Інформація про стан програмного модуля N

Враховуючи, що стани програмних модулів системи можуть змінюватись протягом часу їх функціонування, то в програмних модулях будуть накопичуватись статистичні данні роботи як за добу так і більших інтервалів часу. Ця статистична інформація є корисною для уточнення прийняття рішень. А, також, на певних етапах буде необхідною для видалення частини таблиць з неї за певними правилами. Протягом робочого дня такий обмін інформацією між програмними модулями системи може бути багатократним. В зв’язку з цим необхідним буде також розробка підсистеми, яка оптимізуватиме накопичену інформацію.

Кількість рівнів програмного модуля та їх наповнення залежатимуть від завдань, які вирішуватимуться. Основними завданнями системи є проведення аналізу виконуваних файлів з метою виявлення зловмисного функціоналу і поведінки виконуваного в КС програмного забезпечення (запущених процесів) на наявність зловмисних дій. Обидві ці задачі об’єднує необхідність проведення аналізу поведінки програмного забезпечення, тобто здійснення виявлення через поведінку. Це дозволить увібрати для розгляду і дослідження достатньо широку множину всього файлового ЗПЗ, відділивши в її складовій зловмисну поведінку для аналізу. Тобто, система включатиме засоби дослідження програмного забезпечення розміщеного в КС, яке перебуватиме в активному (виконуваному) стані або в зовнішній пам’яті, спільним елементом яких буде відслідковування і аналіз їх поведінки. В програмному забезпеченні досліджуватиметься наявність таких його можливостей:

1) приховувати ознаки своєї присутності в комп'ютерній системі;

2) володіти здатністю до самокопіювання, в тому числі до створення модифікованих своїх копій;

3) мати здатність до асоціювання себе з іншими програмами;

4) мати здатність до переносу своїх фрагментів в інші області оперативної або зовнішньої пам'яті, в тому числі тих, що знаходяться у віддаленому комп'ютері;

5) отримати несанкціонований доступ до компонентів або ресурсів КС;

6) руйнувати або спотворювати код програм в оперативній пам'яті;

7) спостерігати за процесами обробки інформації та принципами функціонування засобів захисту;

8) зберігати фрагменти інформації з оперативної пам'яті в деякій області зовнішньої пам'яті;

9) спотворювати, блокувати або підміняти виведений в зовнішню пам'ять або канал зв'язку інформаційний масив, що утворився в результаті роботи прикладних програм;

10) спотворювати масиви даних, які знаходяться у зовнішній пам'яті;

11) пригнічувати інформаційний обмін в комп'ютерних мережах, фальсифікувати інформацію в каналах зв'язку;

12) нейтралізовувати роботу тестових програм та засобів захисту інформаційних ресурсів КС;

13) постійно або тимчасово змінювати ступінь захищеності секретних даних;

14) приводити в неробочий стан або руйнувати компоненти системи;

15) створювати приховані канали передачі даних;

16) ініціалізувати раніше впроваджене ЗПЗ.

Підсистеми рівнів програмних модулів РБС, враховуючи перераховані функціональні можливості програмного забезпечення та вирішення проблеми їх виявлення через дослідження і аналіз поведінки, повинні мати такі складові частини:

1) сигнатурний аналізатор;

2) евристичний та поведінковий аналізатор;

3) емулятор процесору;

4) підсистему виявлення несанкціонованих змін з використанням контрольних кодів цілісності;

5) підсистему моніторингу потенційно небезпечних дій;

6) підсистему створення несправжніх об’єктів атаки в мережі;

7) підсистему внесення невизначеності в роботу об’єктів та засобів захисту

КС мережі;

8) контроль ходу виконання програм;

9) підсистему обліку роботи КС.

Сигнатурний аналізатор виконуватиме пошук за сигнатурою програмного коду та за поведінковою сигнатурою. Для цього цим рівнем програмного модуля буде здійснюватись підтримка двох відповідних баз сигнатур. Це дозволить прискорити виявлення відомого ЗПЗ, а також підвищить достовірність пошуку сигнатур в інших КС локальної мережі за рахунок обміну інформацією з баз сигнатур.

Евристичний аналізатор здійснюватиме пошук нових вірусних програм за характерними ознаками, які виділятимуться та представлятимуться у певний масивах та структурах даних. Поведінковий аналізатор здійснюватиме дослідження програмного коду на наявність підозрілої поведінки шляхом виділення характерних ознак та порівняння їх із заданими із залученням інших програмних модулів системи.

Здійснення емуляції програмних команд програмною моделлю процесора необхідно для більш глибшого аналізу програмного коду і за потреби, також, виділення характерних ознак.

Виявлення несанкціонованих змін з використанням контрольних кодів цілісності застосовуватиметься для оперативного дослідження та визначення порушення цілісності програмного забезпечення.

Розробка підсистеми моніторингу потенційно небезпечних дій необхідна для дослідження роботи запущених процесів в конкретній КС та інших КС мережі.

Підсистема створення несправжніх об’єктів атаки в мережі потрібна для спонукання досліджуваного програмного забезпечення до прояву зловмисного функціоналу. Особливо актуальною така підсистема є для використання її в локальній мережі.

Підсистема внесення невизначеності в роботу об’єктів та засобів захисту КС мережі необхідна для ускладнення пошуку ЗПЗ необхідних об’єктів та

інформації. Це дозволить здійснити виявлення ЗПЗ за його повторюваною поведінкою при вирішенні поставленого для нього завдання.

Здійснення контролю ходу виконання програм є необхідною підсистемою для аналізу їх активності першочергово в локальній мережі.

Враховуючи, що в результаті виконання функціоналів підсистем накопичуватимуться великі обсяги різнорідної інформації, то для ефективної роботи РБС потрібні підсистеми програмних модулів для обліку роботи конкретних КС та здійснення оптимізації масивів інформації в системі в цілому.

Узагальнена схема розподілу завдань в програмних модулях системи включатиме стани, в яких перебуватиме програмний модуль конкретної КС в процесі свого функціонування, і зображена на рис. 2.7 у вигляді графу та вершин співвіднесених до різних рівнів.

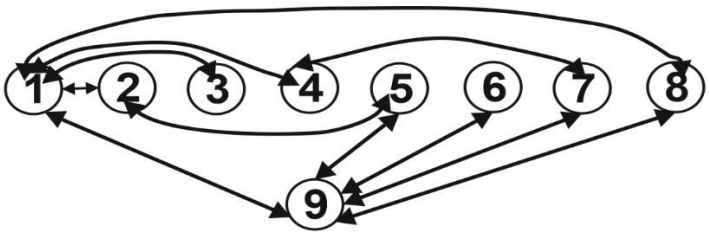


Рис. 2.7. Граф узагальнених станів програмного модуля системи

Позначення:

1 – базовий стан, моніторинг КС, визначення переходу до станів «2», «3», «4», «8»;

2 – перевірка виконуваних файлів, визначення переходу до станів «1», «5»;

3 – перевірка запущених процесів та мережної активності в КС, визначення переходу до станів «1», «6»;

4 – перевірка виконуваних файлів та запущених процесів, визначення переходу до станів «1», «7»;

5 – перевірка виконуваних файлів з використанням інших програмних модулів, повернення на рівень «2»;

6 – перевірка запущених процесів та мережних активностей і їх порівняння з іншими КС мережі, повернення на рівень «4»;

7 – обробка виконуваних файлів та запущених процесів з використанням інших програмних модулів системи;

8 – обробка та оптимізація інформації з бази пакетів програмного модуля із залученням інформації з інших програмних модулів системи, повернення на рівень «1».

Модель РБС, що враховуватиме архітектурні складові і їх стани та зв’язки між ними представимо за формулою (2.2) так:

$$M_A^S = \langle S, G_A \rangle, \tag{2.2}$$

де S – множина станів системи, G_A – орієнтований граф узагальнених станів РБС, який зображено на рис. 2.8.

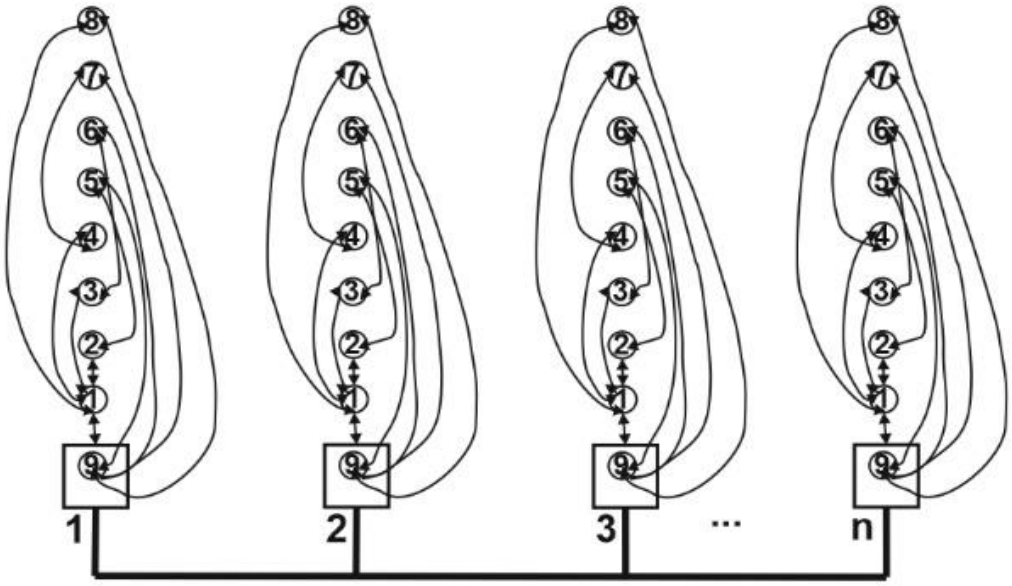


Рис. 2.8. Орієнтований граф зв’язків узагальнених станів програмного модуля системи

Позначення:

«9» – забезпечення зв’язку з іншими програмними модулями системи.

Кожен програмний модуль системи має однакову структуру і розділений на чотири рівні в залежності від функційного призначення та згрупованих в них

завдань:

1) рівень «1» включає моніторинг подій та визначення переходів до наступних рівнів, а також здійснює обробку інформації, що надходить від інших програмних модулів;

2) рівень «2» включає перевірку виконуваних файлів, перевірку запущених процесів та мережних активностей без залучення інформації з інших програмних модулів системи;

3) рівень «3» включає виконання завдань рівня «2» із залученням інформації з інших програмних модулів системи;

4) рівень «4» здійснює обробку, оптимізацію та вилучення інформації з бази програмного модуля КС.

Рівні «1», «3» та «4» обов'язково спілкуються з іншими програмними модулями системи. Програмний модуль, як правило, перебуває на рівні «1», де здійснює моніторинг подій. Якщо відбуваються зміни в КС, тоді в залежності від результатів моніторингу на його рівні приймаються рішення про перехід до вершин «2», «3» або «4» тобто на рівень «2». На цьому на рівні «2» досліджуються завдання з рівня «1» методами та засобами без залучення інших частин системи. Якщо результат дослідження виявиться негативним, тобто ЗПЗ не знайдено, тоді встановлюється, що потрібна глибша перевірка і здійснюється перехід до рівня «3», де задіюються для визначення інші частини системи в мережі. Використання засобів цього рівня залучає розподілені в мережі програмні модулі та в результаті підвищуватиме ефективність виявлення ЗПЗ. На четвертому рівні відбувається оптимізація інформації з бази програмного модуля із залученням інших частин системи, а також після отримання інформації зі всіх інших програмних модулів системи прийняття рішення про стан системи в цілому.

Таким чином, перший рівень містить засоби, які забезпечують автономність роботи програмного модуля системи. На третьому рівні досягається глибший аналіз досліджуваних об'єктів в порівнянні з другим рівнем, що підвищує ефективність системи. Четвертий рівень вирішує частину

задач із самоорганізації системи, пов’язану з оптимізацією накопиченої за час роботи інформації.

Кожен рівень та відповідні йому узагальнені підсистеми в свою чергу теж представляються наборами підрівнів, яким закладено виконання певних функціоналів. Деталізовані дії рівнів кожного програмного модуля через їх функціональні можливості представлено в табл. 2.3.

Таблиця 2.3

Деталізовані дії рівнів програмного модуля

Рівні	Вирішувані завдання	Функціональні можливості
1	2	3
«1»	Початок роботи після запуску програмного модуля; здійснення аналізу змін в структурі програмного забезпечення КС; здійснення аналізу завантажувального сектору жорсткого диску; здійснення аналізу активних процесів; отримання завдань від інших програмних модулів системи.	Визначення необхідності переходу до іншого рівня; перехід до інших рівнів на основі завдань, отриманих від інших програмних модулів системи; формування звіту про виконання віднесених до цього рівня завдань; передача звіту на рівень зв’язку з іншими КС.
«2»	Перехід з рівня «1» з отриманими параметрами завдання; дослідження виконуваного файлу засобами сигнатурного аналізатора.	Здійснення сигнатурного аналізу виконуваного файлу з використанням бази сигнатур; якщо сигнатуру виявлено, то здійснення заходів з його блокування, видача відповідного візуального повідомлення користувачам, інакше здійснення переходу на рівень для детальнішого дослідження; формування звіту про виконання віднесених до цього рівня завдань; передача звіту на рівень зв’язку з іншими КС.

Продовження таблиці 2.3

1	2	3
«3»	Перехід з рівня «2» з отриманими параметрами завдання; дослідження виконуваного файлу засобами евристичного аналізатора.	Здійснення евристичного аналізу виконуваного файлу з використанням бази евристик; якщо сигнатуру виявлено, то здійснення заходів з його блокування, видача відповідного візуального повідомлення користувачам, інакше здійснення переходу на рівень для детальнішого дослідження; формування звіту про виконання віднесених до цього рівня завдань; передача звіту на рівень зв'язку з іншими КС.
«4»	Перехід з рівня «3» з отриманими параметрами завдання; дослідження виконуваного файлу засобами поведінкового аналізатора; надсилання для обробки-дослідження заданого виконуваного файлу та отримання результатів від інших програмних модулів системи.	Здійснення поведінкового аналізу виконуваного файлу з використанням бази поведінкових сигнатур та емулятора процесору; якщо поведінкову сигнатуру виявлено, то здійснення заходів з його блокування, видача відповідного візуального повідомлення користувачам, інакше залучення інших програмних модулів системи; формування звіту про виконання віднесених до цього рівня завдань; передача звіту на рівень зв'язку з іншими КС; повернення на рівень 1 після повного опрацювання заданого виконуваного файлу.
«5»	Перехід з рівня «1» із поставленими завданнями; виявлення несанкціонованих змін з використанням контрольних кодів цілісності.	Виявлення несанкціонованих змін з використанням контрольних кодів цілісності.

Продовження таблиці 2.3

1	2	3
«6»	Перехід з рівня «1» із поставленими завданнями; моніторинг потенційно небезпечних дій.	Відслідковування виконання потенційно небезпечних команд.
«7»	Перехід з рівня «1» із поставленими завданнями; створення несправжніх об’єктів атаки в мережі.	Звернення до однієї КС як до сервера мережі та здійснення відповідної імітації.
«8»	Перехід з рівня «1» із поставленими завданнями; внесення невизначеності в роботу об’єктів та засобів захисту КС мережі.	Внесення невизначеності в роботу об’єктів та засобів захисту КС мережі.
«9»	Перехід з рівня «1» із поставленими завданнями; здійснення контролю ходу виконання програм.	Отримання доступу до запущеної виконуваної програми та підготовка до її перевірки.
«10»	Перехід з рівня «1» із поставленими завданнями; здійснення аналізу отриманої інформації від інших програмних модулів в частині прийняття рішення про стан безпеки в мережі в цілому.	Проведення обробки інформації з сховища програмного модуля та інформації про поточний стан мережі для встановлення загального висновку про стан безпеки в мережі; підготовка звіту.
«11»	Перехід з рівня «1» із поставленими завданнями; здійснення обробки накопиченої інформації.	Оптимізація та часткове видалення накопиченої інформації.

Зображення взаємозв’язку підрівнів представлено на рис. 2.9. графом з вершинами, що відповідають призначенням підрівнів.

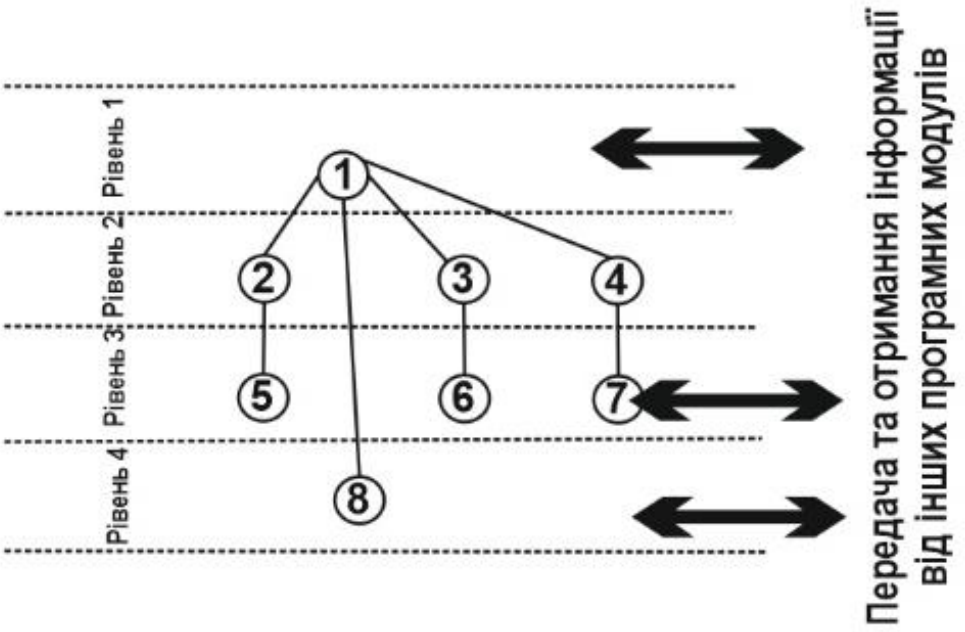


Рис. 2.9. Граф-схема взаємодії підрівнів програмних модулів системи

Представлення інформації про поточний стан програмних модулів РБС, з врахуваннях рівнів перебування, може бути представлений матрицями поточних станів. Наприклад, в табл. 2.4. наведено стани програмних модулів системи в деякий поточний момент часу.

Таблиця 2.4

Матриця поточних станів програмних модулів

Номер програмного модуля РБС	Рівень 1	Рівень 2	Рівень 3	Рівень 4
1	+			
2			+	
…	…	…	…	…
n	+			

Представлена інформація в табл. 2.4 дозволяє здійснювати аналіз системи в цілому та готувати рішення про перехід програмних модулів на інші рівні або призупинку роботи комп’ютерних систем в мережі з видачею відповідного звіту для адміністратора. Крім того, представлення інформації про програмні модулі на основі табл. 2.4 аналогічно деталізується до станів програмних модулів, що є

більш інформативним для прийняття рішень. Наприклад, представимо в табл. 2.5 структуру з інформацією про стани програмних модулів системи на основі графу з рис. 2.9.

Таблиця 2.5

Представлення програмного модуля через його стани

Номер програмного модуля РБС	Стани								
	1	2	3	4	5	6	7	8	(9)

Примітка: (9) – програмний модуль не переходить до стану 9; цей стан відображає зв’язок з іншими програмними модулями.

Аналіз інформації з матриці станів програмних модулів дозволить приймати рішення про подальші дії РБС і її складових частин. Тобто, рішення прийматимуться окремими програмними модулями системи для них та для визначення подальшої поведінки системи в цілому.

Розроблена концептуальна модель [385] архітектури програмних модулів РБС базується на принципах децентралізації, самоорганізації та багаторівневості. Вона дозволяє здійснювати наповнення всієї системи різними функціоналами виявлення ЗПЗ та побудована таким чином, що дозволяє збільшувати кількість рівнів системи без зміни її архітектури. Основою архітектури РБС [172, 364] виступають програмні модулі, які спроектовано однаковими архітектурами, але які дозволяють містити різні данні зібрані з різних КС. Для ефективної роботи РБС необхідним є розробка принципів взаємодії та узгодження роботи різних програмних модулів між собою на їх відповідних рівнях.

2.3. Модель розподіленої багаторівневої системи та принципи взаємодії програмних модулів різних комп'ютерних систем на основі децентралізації та самоорганізації

Представлення розподіленої багаторівневої системи у формі ПМ, розміщених в КС локальної мережі, та врахування характеристики децентралізованості системи вимагає розробки такого методу взаємодії програмних модулів різних КС, який би базувався на таких принципах:

1) спілкування програмних модулів між собою;

2) виконання програмними модулями поставлених завдань без будь-якого впливу ззовні системи;

3) самовідтворення РБС;

4) програмні модулі повинні впливати на себе та інші програмні модулі;

5) система та її компоненти повинні здійснювати самоконтроль;

6) РБС повинна утворювати свої власні цілі.

Врахування при проектуванні РБС цих принципів дозволить створити розподілену децентралізовану систему, яка вирішуватиме поставлені завдання за рахунок наповненого функціоналу та створюватиме власні цілі для роботи. Розробка правильної організації взаємодії модулів системи впливатиме на ефективність виконання поставлених завдань та роботу системи в цілому.

Організація взаємодії програмних модулів системи в процесі спілкування вимагатиме розгляду та опису таких подій на різних етапах їх функціонування:

1) запуск КС та відповідно його програмного модуля спричиняє необхідність виконання зчитування характеристик КС та порівняння їх з тими, що зберігаються в базі; формування пакету інформації з ідентифікатором про початок роботи програмного модуля;

2) відправлення пакету інформації з ідентифікатором про початок роботи іншим програмним модулям системи;

3) якщо не всі КС увімкнені, тоді РБС буде формуватись виключно з тих, які вже працюють;

4) модель поведінки користувачів передбачає різний час початку роботи в локальній мережі, тому обов’язково буде КС, яка була увімкнена першою та активувала програмний модуль і з неї було розіслано пакет інформації, але жодна КС не увімкнена, тому відповіді від них отримано не буде; час відправки пакету фіксуватиме програмний модуль;

5) після ввімкнення другої та інших КС мережі в них формуються пакети інформації про початок роботи програмних модулів системи і надсилаються за реєстром в інші КС мережі, в тому числі, і на першу ввімкнену КС, яка після отримання першого з них відправляє повторно свій пакет інформації про початок роботи; таким чином, вже не буде жодної першої ввімкненої КС, якій не відповів хоча б один програмний модуль іншої КС; обмін пакетів відбудеться усіх з усіма ввімкненими; якщо не всі КС за певний час будуть ввімкнені, а додадуться через тривалий час, тоді всі події з формування системи будуть ті ж, тобто розсилатимуться пакети інформації про початок роботи;

6) кожен програмний модуль РБС після сформування системи здійснює моніторинг подій своєї конкретної КС і при виникненні події, що переводить його в інший стан, який потребуватиме при виконанні завдання підтримки інших програмних модулів системи, розсилає пакет інформації з відповідним ідентифікатором, що видає завдання іншим програмним модулям, які після його виконання повертають йому результат роботи (час повернення результату, як правило є різним);

7) програмний модуль системи, перебуваючи на певному відповідному рівні, надсилає решті програмних модулів вказівку та інформацію про повне виконання завдання з відповідним ідентифікатором; кожен ПМ виконує перевірку та розсилає усім модулям системи результат; у випадку великої завантаженості КС, рівень якої встановлюється ПМ, надсилається відповідне повідомлення та видане завдання очікує в черзі; сумарний результат оцінюється кожним ПМ та розсилається в інші ПМ; обробка такої статистичної інформації в різних ПМ здійснюється за єдиним алгоритмом та приймається рішення про реагування системи в цілому на отриманий результат, бо він буде отриманий

кожним програмним модулем;

8) завершення роботи КС і її вимкнення відбувається тільки після надсилання решті ПМ системи про вимкнення пакетом з відповідним ідентифікатором.

Важливим елементом аналізу спілкування програмних модулів РБС є побудова її часової моделі у вигляді відповідного графіку від старту та безпосередньо роботи програмних модулів і системи, яка демонструватиме залежність між програмними модулями і їх станами від часу функціонування. Параметри такого представлення зображено на рис. 2.10.

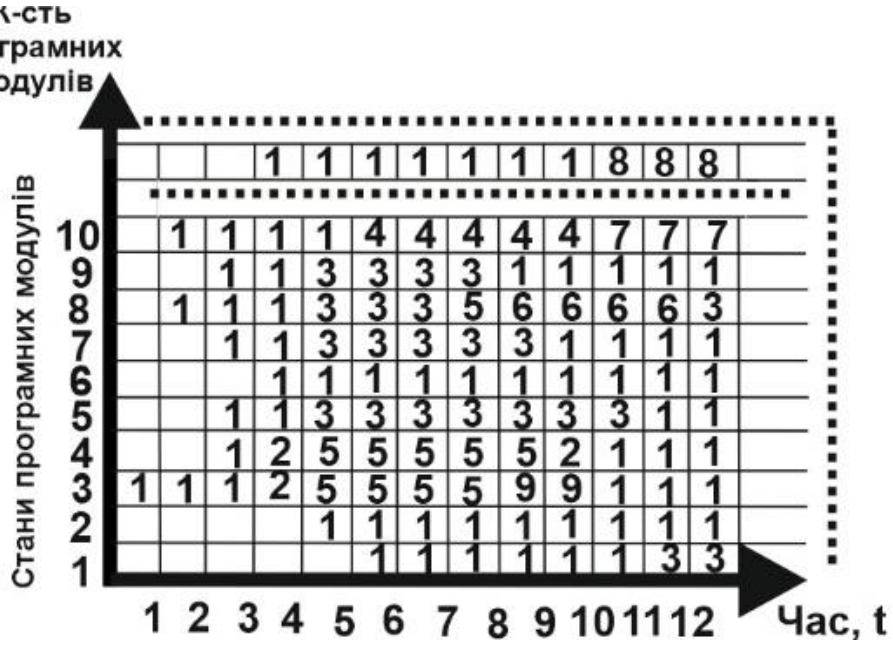


Рис. 2.10. Часова модель функціонування РБС

Відображення станів ПМ вагами критичності в матриці дозволить проводити динамічну обробку подій, які впливатимуть на наступні стани РБС. Формування такої матриці відбуватиметься за потреби, зокрема у випадку дослідження мережної активності в локальній мережі.

З рис. 2.10 видно, що часова модель поведінки РБС через моделі поведінки її ПМ розглядається не як лінійна послідовність множини обчислень, а як дерево можливих обчислень, тобто розгалужена часова модель із зворотніми зв’язками. Тому, представимо РБС моделлю на основі структури Кріпке за формулою (2.3):

$$M_{\text{РБС}} = \langle S, S_0, R, F_{\text{РБС}} \rangle, \quad (2.3)$$

де S – скінченна множина станів РБС; S_0 – множина початкових станів (причому їх може бути декілька); R – множина переходів між станами; $F_{\text{РБС}}$ - функція, що відображає кожен стан з множини S в підмножину атомів, які є істинними у відображуваних станах.

Оскільки, система є розподіленою, тоді множину станів системи представимо через підмножини станів, які відносяться до програмних модулів A_i, i=1,2,...,n, а саме: $S = \bigcup_{i=1}^{n} S_i$, тобто множини станів ПМ формуватимуть множину станів, в якій перебуватиме система. Аналогічно, множина початкових станів РБС $S_0 = \bigcup_{i=1}^{n} S_{0i}$, причому серед їх елементів не може бути однакових. З будь-якого стану $s \in S$ існує мінімум один перехід до іншого стану, тобто справедливим є відношення $R \subseteq S \times S$, що означає для будь-якого стану з множини станів існує відповідний йому стан з цієї ж множини. Нехай T_A - множина атомарних тверджень пов'язаних із станами, які є істинними тільки в цих станах. Якщо множина атомарних тверджень T_A скінчена, тоді множина її підмножин 2^{T_A} буде відображенням $F_{\text{РБС}}$ для множини S в ті підмножини атомів, які будуть істинними в $s \in S$.

Аналогічно, представимо модель архітектури структурної компоненти РБС програмного модуля M_{A_i} за формулою (2.4) так:

$$M_{A_i} = \langle S_i, S_{0i}, R_i, F_{A_i} \rangle, \quad (2.4)$$

де S_{0i} – множина початкових станів ПМ (причому їх може бути декілька); i=1,2,...,n; R_i – множина переходів між станами $s_{ij} \in S_i$; j – кількість i – их станів; F_{A_i} – функція відображення множини станів S_i в множину підмножин множини T_{A_i} з кількістю станів $2^{T_{A_i}}$.

Кожен стан обов'язково має зв'язок з деякими іншими станами. Перехід з

одного стану до іншого, якщо між ними є зв'язок, відобразимо послідовністю $s_{ij}s_{ip}$, де i – номер програмного модуля, j та p – стани цього ж модуля. Тоді, послідовності $s_{ij}s_{ip}s_{ij}s_{ih}s_{iy}s_{iu}s_{ie}s_{ik}\ldots$ означатимуть переходи ПМ з одного стану в інший на протязі часу його функціонування. РБС в процесі функціонування характеризуватиметься множиною послідовностей переходів з стану в стан програмних модулів. Побудуємо структуру Кріпке для архітектури ПМ за його діаграмою переходів, зображеною на рис. 2.11.

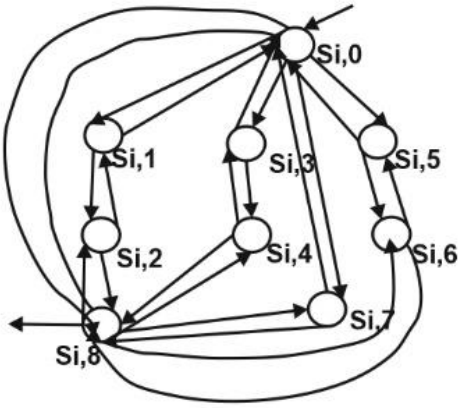


Рис. 2.11. Структура Кріпке програмного модуля A_i

В табл. 2.6 представимо стани з множини станів програмного модуля S_i двійковими послідовностями, а також кодування програмних модулів.

Таблиця 2.6

Кодовані стани ПМ

Номер програмного модуля S_i	Стани	Кодування станів	Кодування програмного модуля S_i
i	$s_{i,0}$	0000	Представлення десяткового числа і двійковим числом з 6-ма розрядами (кількість розрядів може бути змінена за потреби)
	$s_{i,1}$	0001	
	$s_{i,2}$	0010	
	$s_{i,3}$	0011	
	$s_{i,4}$	0100	
	$s_{i,5}$	0101	
	$s_{i,6}$	0110	
	$s_{i,7}$	0111	
	$s_{i,8}$	1000	

Наприклад, для i=5, тобто для п’ятого програмного модуля системи, його код визначатиметься послідовністю 000101i, тоді, відповідно множина станів буде такою: $S_5 = \{0000\mathbf{000101}, 0001\mathbf{000101}, 0010\mathbf{0001010}, 0011\mathbf{000101}, 0100\mathbf{000101}, 0101\mathbf{000101}, 0110\mathbf{000101}, 0111\mathbf{000101}, \mathbf{1}000\mathbf{000101}\}$.

Позначимо V, як множину змінних, які впливатимуть на зміну станів модулів, тоді $V = \bigcup_{i=1}^{n} V_i$, де V_i – множина змінних і-того модуля. Тоді, V' – множина змінних, які впливатимуть на зміну наступних станів модулів, тоді $V' = \bigcup_{i=1}^{n} V'_i$, де V'_i – множина змінних і-того модуля.

Отже, R_i - множина переходів між станами і-того модуля на множинах змінних та визначається так:

R_i = { 0000i: 0001i; 0000i: 0011i; 0000i: 0101i; 0000i: 1000i;
0001i: 0000i; 0011: 0000i; 0101i: 0000i; 1000i: 0000i;
0010i: 0001i; 0010i: 1000i; 1000i: 0010i; 0011i: 0100i;
0100i: 0011i; 0100i: 1000i; 1000i: 0100i; 0001i: 0111i;
0111i: 0001i; 0111i: 1000i; 1000i: 0111i; 0001i: 0010i;
0011i: 0110i; 0110i: 0011i; 1000i: 0110i; 0110i: 1000i }

Функцію F_{A_i} відображення множини станів S_i в множину підмножин множини T_{A_i} задамо таблично фрагментом (табл. 2.7).

Таблиця 2.7

Значення функції F_{A_i}

Елементи множини R_i	Кодований перехід	Стан
$r_{i,0}$	«1000j», 0000i	$s_{i,0}$
$r_{i,1}$	0000i, 0001i	$s_{i,1}$
…	…	…
$r_{i,26}$	1000i, «0000j»	$s_{i,0}$

Позначення:
i, j – номери програмних модулів РБС;
$r_{i,0}$ - $r_{i,26}$ – елементи множини R_i;
$s_{i,0}$ - $s_{i,8}$ – стани програмного модуля.

Логічні значення елементів множини R_i у вершинах графу з рис. 2.8, графу з рис. 2.11 та множини елементів, які породжуватимуть переходи і впливатимуть на процес переходів з вершини у вершину представимо в табл. А.1 додатку А.

Деталізація кожного стану залежить від наповнення РБС функціоналами. Станами, в яких здійснюватиметься підготовка інформації для надсилання іншим програмним модулям, та її тип і структура представлені в табл. 2.8.

Таблиця 2.8

Підготовча інформація про стани ПМ

Стани	Ідентифікатори	Набір параметрів	Тип інформації
$s_{i,0}$	…	…	…
…	…	…	…
$s_{i,7}$	…	…	…

В кожному програмному модулі здійснюються переходи між станами згідно табличної функції F_{A_i} за схемою зображеною на рис. 2.12.

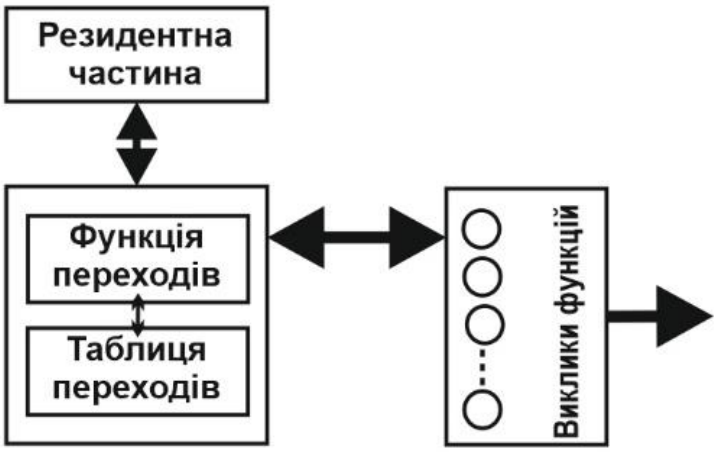


Рис. 2.12. Механізм переходів між станами ПМ

Функціонування РБС відбувається без будь-якого впливу ззовні системи, крім етапів установки системи адміністратором, додавання додатково її програмних модулів та перезапуску у випадку виникнення критичної події, що зупиняє роботу КС та системи. За умови виникнення певних визначених подій РБС призупиняє роботу процесів в КС та видає повідомлення користувачам і

адміністратору. При виконанні основних поставлених завдань участь адміністратора та користувачів не передбачена.

Самовідтворення РБС здійснюється при початку роботи кожного з програмних модулів, додаванні кожного нового програмного модуля, видаленні певного програмного модуля. Ці події зберігають систему і дозволяють вирішувати поставлені на неї завдання. Цей рівень розгляду РБС представляє цю властивість як таку, що відноситься до існування системи. Описані події по організації спілкування між ПМ показують принцип утворення РБС та її самовідтворення в процесі функціонування її частин. Мінімальна кількість працюючих КС, в яких встановлені програмні модулі, не менше двох. Але використання дуже малої кількості ПМ системи не є ефективним, бо тоді для підвищення достовірності виявлення ЗПЗ недостатньою є кількість інформації від різних КС локальної мережі. Наявність тільки одного ПМ в складі РБС, тобто однієї ввімкненої КС, не дозволить використати можливості по переходу в інші стани з метою детальнішого дослідження поведінки ЗПЗ, а використовуватиме лише можливості на рівні однокористувацьких антивірусних програмних засобів. Самовідтворення РБС на рівні її структурних частин відбуватиметься шляхом переведення ПМ, які тривалий час перебувають в одному з станів, до базового стану. А також, РБС здійснюватиме аналіз статистики роботи ПМ КС для оптимізації передавання завдань з метою залучення до їх вирішення найбільш активних КС.

Формування РБС включатиме такі етапи:

1) при встановленні програмного модуля системи в кожну КС локальної мережі необхідна процедура первинної активації всієї системи, яка передбачатиме за вказівкою адміністратора мережі початок роботи і обмін інформацією між програмними модулями про всі КС, в які встановлена система, визначення ідентифікатора кожного програмного модуля та прописування цих даних у внутрішні бази програмних модулів, що крім характеристик КС також включатиме і час їх початку роботи; активація кожного програмного модуля системи відбуватиметься не в одну мить, а в різні часові проміжки, зокрема і збір

характеристик конкретних КС, що впливатиме на початок роботи всієї системи, тому поки не буде отримано в кожній КС підтвердження про завершення встановлення всіх програмних модулів системи, то робота системи не розпочинається; після введення відповідної вказівки адміністратором програмні модулі і система в цілому розпочинають автономну роботу; така послідовність дій при активізації виділяє наступні події: запуск програмного модуля в КС, введення логіна і пароля адміністратора, зчитування характеристик системи, формування ідентифікатора програмного модуля, прописування характеристик та ідентифікатора у внутрішню базу, відправка адміністратором від КС першого пакету в інші КС, отримання пакетів від інших програмних модулів, перевірка адміністратором отримання пакетів зі всіх визначених КС та вказівка-підтвердження адміністратором кожному модулю про завершення етапу активації системи; події цього етапу вважатимемо стартовим станом; при цьому в пакеті відображається відповідний ідентифікатор, що сигналізує про першу активацію системи адміністратором;

2) доповнення системи новими програмними модулями здійснюватиметься за допомогою адміністратора безпосередньо з тієї КС, яка додається в локальну мережу (нова або була на ремонті), при цьому в пакеті відображається відповідний ідентифікатор, що сигналізує про додавання в систему адміністратором ще однієї КС; ця подія заставляє інші програмні модулі системи після отримання вказівки долучити цей програмний модуль і надіслати пакет про свої данні, після отримання пакетів від всіх програмних модулів адміністратор активує новий програмний модуль, який стає частиною системи, при цьому всі решта модулів створюють нову сторінку в своїй базі і прописують оновленні данні про структуру системи та використовують їх в подальшій роботі; при збереженні даних про структуру системи враховується час активації кожного програмного модуля;

3) вилучення КС з локальної мережі призводить до втрати ПМ системи; виявлення цієї події іншими ПМ може відбутись при коректному вимкненні КС, тоді ПМ цієї КС передає пакет з відповідним ідентифікатором; якщо ж відбувся

аварійний вихід КС, тоді інформація про відсутність частини системи переданою не буде, а виявиться під час спроб інших ПМ системи спілкуватися з цим ПМ.

Програмні модулі РБС при перебуванні на третьому і четвертому рівнях для вирішення поставлених завдань залучатимуть технології самонавчання, що впливатиме на зміну їх роботи при вирішенні таких же завдань на наступних етапах життєвого циклу; при цьому вони, передаючи завдання для обробки на інші ПМ, та результати їх виконання впливатимуть на інші програмні модулі; в цілому вплив ПМ, як структурних частин системи, на інші ПМ дозволить еволюціонувати РБС.

РБС на рівні її структурних частин ПМ здійснюватиме самоконтроль, що проявлятиметься в періодичній перевірці комплектності системи, здійсненню аналізу наявності ПМ, які тривалий час перебувають в одному й тому ж стані та потребують автоматичного зняття поточних завдань з виконання та переведення до іншого стану, обробка баз ПМ для оптимізації та розподіл ПМ на декілька груп згідно аналізу їх станів протягом тривалого проміжку часу. Всі ці завдання можуть суттєво завантажити КС. Оскільки завдання в КС представляються процесами, то аналіз їх виконання зі сторони РБС та прийняття відповідного рішення про продовження їх роботи вирішуватиме завдання здійснення самоконтролю. Крім того, ці процеси, які генеруватиме своїми завданнями РБС, можуть призвести не тільки до зниження продуктивності КС, але і до взаємоблокування.

Відомі методи уникнення взаємоблокувань вимагають для своєї реалізації різних обсягів ресурсів. Крім того, перевантаження операційної системи цими завданнями призводить до загального зменшення продуктивності КС [370]. В наслідок цього більшість виробників операційних систем, які, зокрема, використовуються не тільки побутовими користувачами, але і організаціями та підприємствами, відмовляються від реалізації в складі операційних систем методів уникнення взаємоблокувань. Операційні системи, які використовуються в організаціях та підприємствах, можуть відрізнятись за особливостями внутрішніх алгоритмів керування ресурсами КС, типами підтримуваних

апаратних платформ, областями використання. Всі ці особливості можуть вплинути на виконання КС поставлених задач і при функціонуванні РБС не повинні нею сповільнюватись та блокуватись. Тому, важливою вимогою до можливостей РБС є така, щоб плануванням виконання процесів операційною системою, які задані користувачем, продовжувало виконуватись без суттєвого впливу зі сторони РБС. Для процесів, які формуватимуться як завдання РБС, повинен бути встановлений порядок їх надходження, згідно з яким би вони не могли призвести до блокування КС. Це одна із задач по самоорганізації РБС в частині здійснення самоконтролю. Для її вирішення розглянемо життєвий цикл процесів комп'ютерної системи [371]. Класичне представлення станів процесу відповідною діаграмою не відображає можливості наближення до стану взаємоблокування. Тому, для кожного процесу введемо поняття його сигнатури [337], до якої включимо сукупність характеристик, які в загальній кількості різних процесів виступатимуть основою для прогнозування їх попадання до стану взаємоблокування або сповільнення роботи КС. Такими характеристиками [371] можуть бути ідентифікатор процесу, ідентифікатор батьківського процесу, ідентифікатор користувача, якому належить процес, пріоритет процесу, квоти процесу, дескриптори відкритих процесом файлів. Такі характеристики для процесів, отримані в певний момент часу, будуть унікальними, тобто не буде жодної сигнатури, яка б повторилась двічі. Отже, життєвий цикл процесу можна подати у вигляді послідовності станів, через які проходить цей процес. Перехід із стану в стан відбувається через зміну певних параметрів, якими характеризується процес. Зміна параметрів процесу відбувається з певних причин: дії ОС, дії інших процесів, виконання власного програмного коду. Стан кожного окремого процесу буде впливати на стан КС в цілому. Якщо розглядати проблему із взаємоблокуванням процесів, то в нього можуть потрапляти процеси, які взаємодіють між собою і породжені РБС. Вважатимемо, що використання компонентів РБС переважно буде відбуватись в однопроцесорних КС. Тоді, процеси почергово отримуватимуть центральний процесор і відповідно будуть в певних станах. У стан взаємоблокування вони, як правило,

потраплятимуть зі станів «заблокований» або «очікуючий». Перед входження у взаємоблокування процес перебуватиме в певному граничному стані [337], після якого настане стан «взаємоблокування». РБС при породженні великої групи процесів повинна врахувати можливість настання взаємоблокування спрогнозувавши його. Внутрішній планувальник кожного ПМ при виконанні великої кількості завдань повинен здійснити прогнозування потрапляння процесів у стан взаємоблокування. Для вирішення цього завдання в РБС реалізовано алгоритм прогнозування [383] потрапляння процесів в стан взаємоблокування на основі використання системи нечіткого логічного висновку.

Важливим елементом самоорганізації РБС є розробка в ній механізмів для утворення своїх власних цілей. До таких цілей віднесемо такі: динамічне формування системи; розподіл та співвіднесення всіх структурних частин за групами завантаженості; обробка критичних подій в системі; здійснення самоконтролю кожним ПМ; колективне виконання завдань, розв’язуване одним ПМ; обробка та оптимізація накопичених статистичних даних.

Основою побудованої моделі РБС [380] є її структурні компоненти, які представляються програмними модулями, що можуть перебувати в різних станах. Перехід між станами ПМ здійснюється на основі визначеної множини переходів. Взаємодія та спілкування між ПМ базується на основі їх перебування в певних станах під час експлуатації. РБС [363, 385] є реагуючою системою, яка здійснюватиме моніторинг визначених подій. Кожен програмний модуль містить резидентний механізм, рушійні механізми для переходу між станами, переходи між якими задаються підмножинами переходів, данні для яких формуватимуться у його внутрішніх рівнях.

Розроблена модель [364, 380] архітектури типових компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення базується на використанні структур Кріпке. Таке представлення компонентів через стани, в яких можуть перебувати програмні модулі під час функціонування, дає змогу враховувати для визначення стану безпеки всієї

розподіленої системи та її компонентів перебування їх в різних станах. Розроблена модель передбачає можливість збільшення кількості рівнів системи без зміни її архітектури. Основою архітектури РБС виступають програмні модулі з однаковими архітектурами, але при цьому кожен з них може самостійно приймати рішення на основі різних даних зібраних у локальній комп'ютерній мережі.

2.4. Метод взаємодії компонентів розподіленої багаторівневої системи виявлення ЗПЗ у локальних мережах

Метод взаємодії компонентів розподіленої системи виявлення ЗПЗ встановлює порядок здійснення комунікації між частинами системи та обміну знаннями між ними. Він застосовуватиметься для вирішення задач верхнього рівня організації взаємодії, тобто тільки для організації взаємодії частин системи і представлення її цілісною. Для вирішення проблеми з безпосереднього виявлення ЗПЗ в локальних комп'ютерних мережах застосовуватимуться методи, які відноситимуться до нижчого рівня системи, що включатимуть архітектурні особливості розподіленої системи і технології виявлення ЗПЗ. Типова схема розміщення методів в системі та їх взаємозв'язку зображена на рис. 2.13.

1. Метод виявлення файлового ЗПЗ	…	1. Метод виявлення файлового ЗПЗ
2. Метод виявлення мережного ЗПЗ	…	2. Метод виявлення мережного ЗПЗ
…	…	…
Метод взаємодії компонентів РБС		
1	…	n
Комп'ютерні системи		

Рис. 2.13. Схема розміщення методів в системі та їх взаємозв'язку

Вважатимемо, що запуск програмного модуля РБС в конкретній КС здійснено успішно, тобто з програмного забезпечення РБС запущена та частина, яка відноситься і відповідає за запуск тільки в КС. Ця запущена і активна функція є стартовою, в задачі якої входить активація програмного модуля РБС в КС і подальший запуск з неї функцій, що реалізують збірку РБС в мережі. Для розгляду основних кроків методу вважатимемо, що не менше, ніж в двох КС мережі стартова функція виконалась успішно. Крім того, вважатимемо що програмний модуль вже було інстальовано раніше успішно і цей запуск не є інсталяційним. Час обороту між двома різними програмними модулями встановлюється при інсталяції програмного модуля шляхом відправки першого повідомлення всім решті модулям та отриманням відповіді від них. Тоді, подальші процеси, що протікатимуть в мережі, які пов’язані з функціонуванням системи, представимо такими кроками методу взаємодії компонентів розподіленої системи:

Крок 1. Визначення станів програмних модулів РБС.

1.1. Визначення стану кожного програмного модуля в КС, в яких встановлена РБС. Виконання цього кроку відбувається на етапах початку роботи ПМ при його первинній інсталяції, при щоденному завантаженні КС, в процесі функціонування КС та при завершенні роботи КС. Визначення стану програмного модуля в КС включає перевірку КС, її програмного забезпечення та безпосередньо рівня активності самого ПМ.

1.2. Занесення інформації, отриманої на кроці 1.1, у внутрішні бази кожного програмного модуля системи.

1.3. Порівняння результатів сканування КС, отриманих на кроці 1.1, з попередніми результатами сканувань за певний період, які зберігаються в базі сканувань. Якщо результати сканування не збігаються з попередніми, то КС блокується і видається відповідне повідомлення на екран. Якщо основні параметри сканувань збігаються, тоді ПМ продовжує роботу.

1.4. Підготовка і формування пакету повідомлення про свій стан кожним модулем системи.

1.5. Занесення повідомлення пакету, сформованого на кроці 1.5, у базу повідомлень ПМ (відправлених та отриманих).

1.6. Надсилання пакетів повідомлень про свій стан у всі решту компонентів РБС згідно заданого реєстру програмних модулів системи і комп'ютерних систем, в яких вони розміщені. Номери та адреси програмних модулів РБС занесені в базу програмних модулів решти КС.

Крок 2. Обробка відповідей від ПМ КС на відправлені пакети.

2.1. Отримання відповідей на надіслане повідомлення про стан програмного модуля. Якщо надісланий пакет було доставлено успішно, то про це надсилається відповідна відповідь, причому вона обов'язково надходить окремо після відправленого першого пакету від всіх компонентів системи. Очікування відповіді програмним модулем системи відбувається за заданим інтервалом часу, який розраховується при первинній інсталяції системи та враховує технічні можливості мережі по швидкості передачі пакетів, а також можуть бути введені певні критерії, виконання яких вказує на необхідність очікування повідомлення на надісланий пакет, а не перехід на наступний крок. Певна кількість КС, які містять ПМ РБС можуть бути вимкненими, тому ПМ не активні, і це теж враховується при оцінці стану цілісності РБС та її структури в певні моменти часу.

2.1.1. Отримання відповідей від кожного, зареєстрованого в РБС, програмного модуля з решти КС про успішне отримання ними пакету з повідомленням про свій стан, тобто отримання позитивного результату на виконаний крок 1.6 від усіх ПМ РБС.

2.1.2. Відповіді про успішне отримання пакету з повідомленням про стан програмного модуля, який надіслав це повідомлення всім решті ПМ з РБС, надійшли тільки від певної кількості ПМ, а від деяких ПМ взагалі не отримано ніякої відповіді.

2.1.3. Не отримано відповіді від жодного ПМ системи у встановлені часові вимоги.

2.2. Занесення отриманої інформації на кроці 2.1 до бази повідомлень

модуля.

2.3. Фактичне формування або підтвердження цілісності РБС з активних програмних модулів. На кожен пакет, який формується після ввімкнення КС і надсилається в решту КС, обов'язковою є відповідь про його отримання та включення ПМ до реєстру активних ПМ РБС.

Крок 3. Обробка програмним модулем невизначеностей, пов'язаних з відсутністю відповідей на відправлені пакети.

3.1. Якщо відповідь на відправлений пакет із заданого програмного модуля певної КС не отримано взагалі на протязі певного інтервалу часу або визначення такого факту за іншими критеріями, тоді здійснити сканування необхідного порту заданої комп'ютерної системи.

3.2. Якщо отримано відповідь, що пакет не доставлено через збої в системі передачі, тоді повторити надсилання пакету на визначену КС, тобто виконати крок 1.6 для одного ПМ.

Крок 4. Сканування заданого порту КС. Номер порту через який буде здійснюватись обмін повідомленнями між програмними модулями визначено при налаштуванні під час встановлення програмного модуля в КС. Також, додатково може бути встановлено ще декілька портів як резервних для підвищення живучості системи.

4.1. Якщо сканування порту успішне, тобто порт є доступним, тоді робиться відмітка в базі відправлених та отриманих повідомлень і ПМ за цією КС переходить в стан очікування пакету від неї.

4.2. Інакше, тобто якщо сканування неуспішне і показує, що порт недоступний, тобто що порт закрито, тоді робиться відмітка в базі відправлених та отриманих повідомлень і ПМ за цією КС переходить в стан очікування пакету від неї. Досліджувана КС може бути вимкнена. В цьому випадку здійснити почергово сканування резервних портів. Якщо відповідь негативна, тоді ПМ переходить до стану очікування результатів від такої КС, інакше робиться звірка такого результату з іншими активними ПМ, які вже сформували РБС.

Крок 5. Оцінка стану програмного модуля та її звірка між рештою ПМ РБС

на етапі обміну повідомленнями. Якщо відбуваються такі події для ПМ, який надсилав пакети на всі КС: за відправленим пакетом відповіді не отримано; після сканування порту відповіді у вигляді результату не отримано або отримано, що відкритий. Тоді ПМ здійснює звірку свого результату з результатами решти КС.

5.1. Сформувати і задати запит у вигляді пакету решті програмних модулів, крім досліджуваного, які повинні надіслати підтвердження про свою роботу, про стан того досліджуваного програмного модуля.

5.2. Для здійснення дослідження за вказівкою одного вибраного ПМ РБС, решта ПМ надсилають йому по одному пакету про свій стан та обробляють відповіді від нього.

5.3. Надсилання результатів дослідження вибраного програмного модуля запитуваному модулю.

5.4. Обробка результатів дослідження вибраного програмного модуля.

5.4.1. Якщо програмні модулі отримали таку ж відповідь від досліджуваного модуля, як і програмний модуль, що активізував цю подію перевірки, тоді всі ПМ РБС вважають, що досліджуваний модуль поки не активний і продовжують очікувати пакету від нього, коли КС ввімкнеться. Досліджувана КС може бути виключена, тоді всі ПМ отримають однакову відповідь.

5.4.2. Якщо програмні модулі отримали різні відповіді від досліджуваного модуля або певна частина отримала, а інша не отримала, або інші відповіді, як і програмний модуль, що активізував цю подію перевірки, тоді всі програмні модулі повідомляють адміністратору про цю подію, видають повідомлення на екран своєї КС, записують в свій реєстр позаштатних ситуацій та зменшують РБС на один програмний модуль.

Крок 6. Визначення стану розподіленої багаторівневої системи. Після певного часу, встановленого адміністратором, програмні модулі визначають стан РБС з певною періодичністю за часом або за настанням критичних подій в КС.

6.1. Визначення свого стану кожним ПМ окремо.

6.2. Підготовка і формування пакету з повідомленням про свій стан кожним ПМ системи і надсилання його від кожного ПМ решті активних ПМ.

6.3. Проведення підтвердження про успішну доставку пакету кожним ПМ від інших ПМ.

6.4. Обчислення за формулами (2.12) та (2.39) стану РБС кожним ПМ на основі отриманих даних зі всіх активних ПМ РБС. За формулою (2.12) обчислення стану безпеки на першому етапі і обчислення стану безпеки за формулою (2.39) на другому етапі.

6.5. Обмін пакетами з повідомленнями про стан РБС від кожного ПМ з рештою ПМ системи.

6.6. Аналіз і обробка отриманих результатів кожними ПМ РБС.

6.6.1. Всі ПМ обчислили стан РБС однаково, тоді система продовжує роботу. Результати звірки про стан РБС збігаються, тоді надсилається відповідне повідомлення від кожного програмного модуля всім решті модулів. Ця інформація заноситься у внутрішній реєстр подій.

6.6.2. Якщо виявиться, що хоча б один з програмних модулів відповість всім решті, що він отримав від певного модуля результат відмінний від свого і їх, тоді він вказує в повідомленні для всіх програмних модулів номер того програмного модуля і його результат, який відмінний від загального результату. В цій ситуації всі програмні модулі, крім того в якого відмінні від інших результати, відправляють йому команди для блокування КС та виведення повідомлення про причину блокування на екран, а самі блокують надходження будь-яких пакетів від нього і вилучають його з реєстру модулів системи. РБС продовжить роботу, але кожен ПМ відобразить на екран КС повідомлення про номер та стан ПМ, якого вилучено.

6.6.3. Якщо виявиться, що програмний модуль отримав від всіх однакове значення стану системи, але воно не збігається з розрахованим ним і при цьому він надіслав своє значення всім, тоді цей програмний модуль блокує КС та видає відвідне ситуації повідомлення на екран.

6.6.4. Якщо від частини ПМ РБС пакети на надійшли, тоді здійснення

запуску процедури (крок 3) визначення причини не отримання повідомлень. Після обробки таких подій продовження роботи ПМ.

Крок 7. Прийняття рішення про подальшу роботу РБС в цілому на основі дослідження її стану програмними модулями після виконання кроку 6. Визначення рівня кожного ПМ системи на основі рівнів її ПМ, розподіл їх за групами ризику бути ураженими, тобто рівнями визначених загроз, та прийняття рішення про подальшу роботу системи.

7.1. Якщо стан РБС, визначений на кроці 6, встановлено як такий, що ступінь безпеки складає 0–30 %, тоді здійснити блокування ПМ всіх КС і повідомити адміністратору. Така подія може відбутись також за умови наявності в системі великої кількості ПМ, які перебувають на рівнях «2» або «3» тривалий час.

7.2. Якщо стан РБС, визначений на кроці 6, встановлено як такий, що ступінь безпеки складає 30-75 %, тоді здійснити блокування тільки тих КС, ПМ яких вказує на віднесення КС до рівня 0–30 %, повідомити адміністратору і перерахувати стан системи на ту кількість КС, які залишились.

7.3. Якщо стан РБС, визначений на кроці 6, встановлено як такий, що ступінь безпеки складає 75–100 %, тоді дослідити ті КС, в яких ступінь безпеки складає менше 75 % тривалий час. Якщо перевищено ліміт часу на подолання загрози, тоді здійснити блокування ПМ тієї КС, повідомити адміністратору і продовжити роботу без вилученого ПМ.

7.4. Якщо стан РБС, визначений на кроці 6, встановлено як такий, що ступінь безпеки складає 75–100 % і після дослідження тих КС, в яких ступінь безпеки складає менше 75 % тривалий час, їх не виявлено, то продовжити роботу.

Крок 8. Вилучення активного програмного з РБС в результаті вимкнення КС.

8.1. Якщо одна з КС вимикається, тоді про це її програмний модуль повідомляє решту модулів і тільки після цього відбувається вимкнення.

Крок 9. Події, які активують методи виявлення зловмисного програмного

забезпечення, впливають на зміну стану ПМ РБС. При цьому здійснюється дослідження інших КС на наявність подібних активностей та обмін отриманими результатами.

9.1. При переході ПМ на рівень «2» застосовується метод виявлення файлового зловмисного програмного забезпечення.

9.2. При переході програмного модуля на рівень «3» застосовується метод виявлення мережного зловмисного програмного забезпечення.

Крок 10. Обробка та оптимізація статистичних даних накопичених в системі кожним модулем окремо.

10.1. При невеликій завантаженості КС і відсутності інших завдань, які пов’язані з необхідністю перебування програмного модуля на рівні «2» або «3», та тривале за часом перебування в стані «1», ПМ переходить до рівня «4» та здійснює дослідження накопичених статистичних даних. Зокрема, він відслідковує та аналізує порядок запуску ПМ в порівнянні з іншими програмами КС та час його старту за певний період часу. Обробка таких даних включає обчислення основних статистичних параметрів: визначення середнього значення, дисперсії та середнього квадратичного відхилення.

10.2. Здійснення оптимізації накопичених статистичних даних в базах програмних модулів після успішного завершення кроку 10.1.

10.3. Якщо ж на кроці 10.1 відбулось обчислення і виявлено значне відхилення, тоді оптимізація даних не здійснюється, блокується КС та видається повідомлення адміністратору.

Крок 11. Обмін знаннями в середині розподіленої багаторівневої системи.

11.1. Отримані результати одним програмним модулем РБС, які стосуються виявлення і локалізації ЗПЗ, формуються в пакет і надсилаються іншим ПМ в мережі, які застосовують ці результати для перевірки своїх КС.

Крок 12. Сумісне виконання завдань компонентами РБС.

12.1. Колективне виконання завдань, пов’язаних з виявленням ЗПЗ, здійсненням збільшення обчислювальних потужностей для програмного модуля шляхом відправки частини задач в інші КС для дослідження програмних

об’єктів, в яких є підозрілі поведінки. Зокрема, залучення в ПМ інших КС емуляторів роботи процесора для спонукання до різних проявів ЗПЗ.

12.2. Підготовка і надсилання іншим ПМ результатів, які отримані на кроці 12.1.

Крок 13. Робота РБС в складі всього одного програмного модуля.

13.1. Якщо РБС залишається в складі всього одного ПМ, в зв’язку з коректним завершенням роботи інших ПМ, тоді вона переходить до обмеженого використання своїх можливостей і може переходити з першого рівня на другий, в якого обмежені можливості.

13.2. Після наступного нового запуску РБС в цьому ПМ здійснюється обов’язкова перевірка його стану за умови, якщо він переходив до рівня «2», залишаючись одним в системі. Таким чином, кожен останній працюючий ПМ при новому запуску РБС так перевіряється.

Крок 14. Поповнення РБС новими модулями.

14.1. Комп’ютерні системи, які ввімкнуться пізніше, перебудують систему, розширивши її. Кожен програмний модуль КС здійснить розсилку пакетів іншим ПМ системи за правилами з кроку 1.

14.2. Щойно встановлені ПМ розширюють систему, то після кроку первинної інсталяції виконують крок 14.1.

Взаємозв’язок кроків методу взаємодії компонентів РБС зображено на рис. 2.14. Кроки методу можуть виконуватись не послідовно, а в залежності від настання певних подій в системі, які вимагатимуть реагування на них.

Кроки 7-12, 14 пов’язані з кроком 1 і можуть викликатись після повного його виконання або повертатись після повного свого виконання до нього. Така їх взаємопов’язаність спричинена тим, що РБС є реагуючою системою на певні події що протікають в КС. Настання таких подій відслідковується динамічно в КС та змінює стан ПМ, який як компонента РБС повідомляє про зміну свого стану решті компонентів системи, що відображається в постійній зміні станів і рівнів РБС.

Крок 13 відображає здатність РБС бути в складі всього одного модуля

після виконання кроку 1 або після виконання кроку 8.

Перехід з кроку 4 до кроку 1 здійснюється не одним і тим же ПМ, а двома різними ПМ РБС.

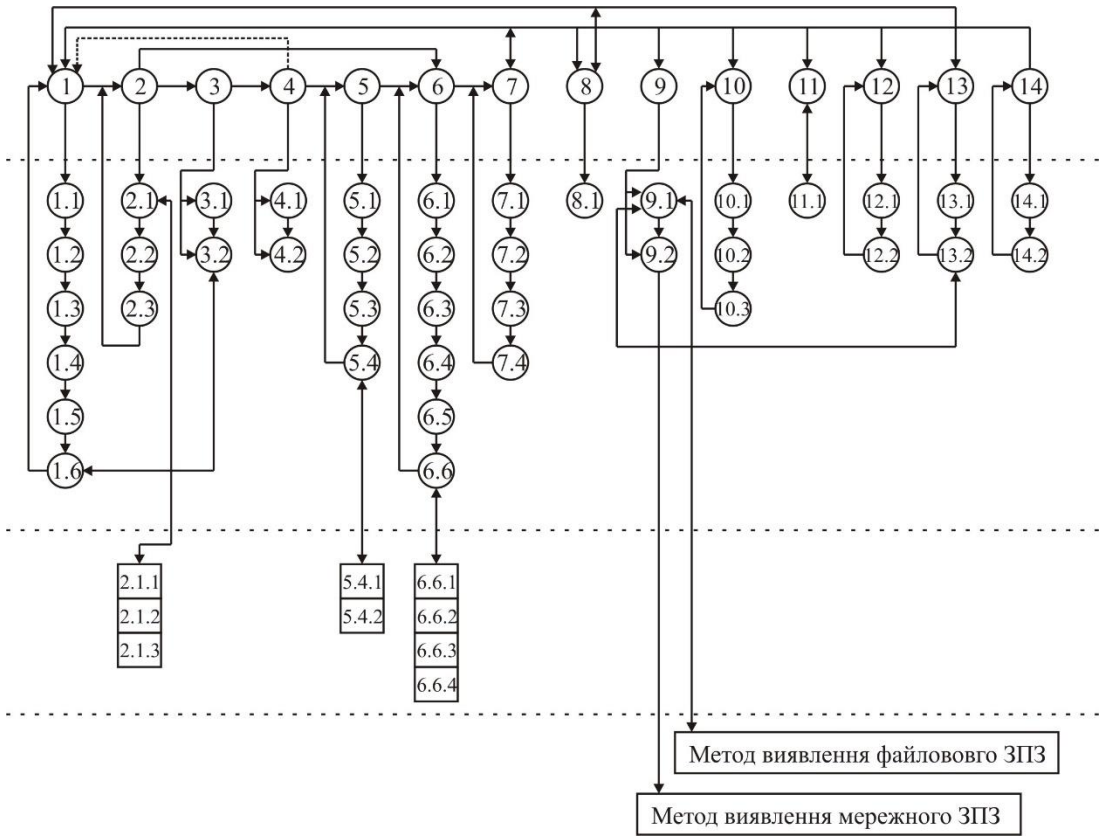


Рис. 2.14. Схема взаємозв’язку кроків методу

Для виконання кроку 6.4 проаналізуємо стани ПМ та встановимо їх взаємозв’язок з станом системи в цілому. Кількість станів ПМ в розроблюваній РБС фіксована і становить вісім, але за потреби розширення системи новими підсистемами може бути більше або бути узагальненою на будь-яку їх скінчену кількість. Кожен ПМ може бути в поточний момент часу тільки в одному із станів, тому для РБС, яка складається з n компонентів, кількість можливих станів у поточний момент часу дорівнює 8^n. При невеликому значенні n ці стани можна однозначно відносити до безпечних або небезпечних. Але при достатньо великому значенні n кількість різних варіантів зростає і тому необхідним є розробка функційних залежностей, які б дозволяли РБС використовуючи їх

однозначно ідентифікувати свій стан. Не тільки значення n впливає на розрахунки, а також, і різний час ввімкнення різних КС, що робить РБС дуже динамічною та в зв‘язку з цим потребує організації внутрішнього самоконтролю. За потреби додавання нових можливостей виявлення ЗПЗ РБС програмні модулі можуть мати не вісім а більше станів, що теж впливатиме на складність визначення стану РБС. Якщо k – кількість тих ПМ РБС, які розміщені в КС, що не ввімкнені, тоді кількість ПМ в системі n-k і відповідно кількість можливих станів РБС дорівнює 8^{n-k}. При цьому неувімкнені КС представлені в системі ПМ відображаємо такими, що мають невизначений стан, який позначимо нулем. Тоді кількість станів менша, ніж загальна кількість можливих станів РБС. Але з додаванням цього неіснуючого стану, якщо його враховувати, тобто відображати реакцію РБС на неактивні ПМ, то загальна кількість можливих станів системи буде 9^n. Кількість станів РБС без активних ПМ дорівнює $9^n - 8^{n-k}$. Оскільки кількість станів РБС скінчене число, тоді введемо і позначимо E_m – m – й стан РБС, де $m = 1, 2, \dots, d$. РБС може перебувати в одному із станів $E_1, E_2, \dots, E_d$, число яких скінчене. Крім того, кожен стан системи відображається через її ПМ і представляється матрицею E_m за формулою (2.5):

$$E_m = \begin{pmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,8} \\ \cdots & \cdots & \cdots & \cdots \\ q_{n,1} & q_{n,2} & \cdots & q_{n,8} \end{pmatrix}, \quad (2.5)$$

де $q_{i,j}$ – відображення результату перебування i – того ПМ в j – стані.

Це означає, що всі стани E_m є детермінованими, тобто такими, відомості про які можна задати при їх створенні. Але проблемою є кількість таких станів, яка залежить від кількості КС в локальній мережі. В процесі роботи РБС може містити не повний набір ПМ, бо не всі КС можуть бути увімкнені або будуть вилучені з системи, тому кількість станів може бути менша від максимально розрахованої. Але навіть при меншій кількості ПМ і враховуючи не активні, кількість всеможливих варіантів все-таки визначена, тобто кожен стан

визначений за умови перебування ПМ у восьми зазначених станах. Перехід системи з стану E_m до стану E_q пов‘язаний зі зміною станів чи стану ПМ. Така зміна може бути викликана подіями, на які відреагує ПМ певної КС, або можливим є здійснення аналізу стану РБС через певні фіксовані чи визначені розрахунками інтервали часу, покаже його незмінність, або в результаті додавання чи вилучення ПМ. Зміна станів ПМ в КС-ах відбувається через вплив випадкових процесів по відношенню до РБС, але які заставляють РБС змінювати свій стан. РБС може повертатись до попередніх станів. Ймовірність переходу з стану E_m до стану E_q не залежить від стану E_m і від того коли і як РБС прийшла в цей стан. Процес, який здійснив вплив на зміну стану системи, може протікати в КС незалежно від системи, а на стани РБС впливатимуть попередні події (бо ПМ зберігає інформацію про перебування в станах) і поточні. Тому, розглядатимемо не ймовірність переходу РБС з стану в стан, а виділимо рівні ймовірностей «бути не в критичному стані» ПМ, оскільки ця характеристика зібрана з усіх ПМ та інтегрована в одне число визначатиме рівень РБС. Враховуючи це в РБС зміна станів, кількість яких дискретна і скінчена, пов’язана з неперервним часом, тобто момент переходів випадковий. Опис моделі РБС через її стани на основі ланцюгів Маркова з неперервним часом є неможливим, бо хоча спільним є те, що зміни станів системи відбуваються в будь-які випадково можливі моменти часу, але відмінним є те, що для ланцюгів Маркова обов’язковим є незалежність попередніх випробувань, тоді як РБС залежить від того якими були початкові стани ПМ та наступні перед тим, в якому перебуває в поточний момент часу.

Також, врахуємо крок 1.3 методу про стани програмного модуля, які досліджуватимуться, як окремого компонента системи і визначатимуть стан саме ПМ з використанням попередньої статистики з його періоду функціонування. Для розгляду оцінки стану ПМ на першому кроці здійснюється сканування КС. Під таке сканування потрапляють апаратні засоби, програмне забезпечення КС, програмне забезпечення модуля РБС. В табл. 2.9 представлено об’єкти

сканування та відображена їх часова статистика.

Таблиця 2.9

Результати сканування

Перелік об’єктів сканування							
Апаратне забезпечення		Програмне забезпечення			Програмне забезпечення ПМ РБС		
Назва	Час	Назва	Час	Розмір	Назва	Час	Розмір
…	…	…	…	…	…	…	…
Назва	Час	Назва	Час	Розмір	Назва	Час	Розмір

Для програмного забезпечення відображається час встановлення та розмір. Для апаратного забезпечення в базі ПМ зберігається час останнього сканування. Якщо виявлено нове програмне забезпечення, тоді здійснюється оновлення реєстру ПМ і видається повідомлення на екран. Для уникнення таких подій адміністратор при встановленні нового програмного забезпечення відомості про нього заносить в ПМ та прописує шлях до нього в базі ПМ. Після сканування здійснюється аналіз нових результатів сканування і збережених при попередніх скануваннях.

Оцінюватимемо РБС та її компоненти програмні модулі такими характеристиками: рівень безпеки РБС, рівень безпеки програмного модуля, рівень безпеки стану ПМ. Ці три характеристики є пов’язаними між собою, а саме: рівень безпеки РБС формується на основі рівнів безпеки програмних модулів, а рівні безпеки ПМ формуються на основі рівнів безпеки станів ПМ. Рівні безпеки станів ПМ отримуються на основі результатів застосування методів та підсистем ПМ РБС виявлення ЗПЗ в КС, які оцінюються відповідними ймовірностями ураження.

Приймемо за рівень загрози стану програмного модуля ймовірність ураження об’єктів КС зловмисним програмним забезпеченням та визначимо його за формулою (2.6):

$$R_{z,s,i} = k_{s,i} \cdot p_{s,i} , \qquad (2.6)$$

де $R_{z,s,i}$ – рівень загрози для i – того стану; $k_{s,i}$ – коефіцієнт загрози бути ураженим ЗПЗ, значення якого встановлюється з відрізку $[0; 1]$ в залежності від того які функційні навантаження закладено у певний i – стан, s – це позначення стану; z – позначення загрози; $p_{s,i}$ – ймовірність бути ураженим ЗПЗ. Рівень безпеки стану ПМ виразимо через його рівень загрози за формулою (2.7):

$$R_{b,s,i} = 1 - R_{z,s,i}\ , \tag{2.7}$$

де $R_{b,s,i}$ – рівень безпеки для i – того стану; b – позначення безпеки.

За рівень загрози програмного модуля приймемо ймовірність ураження КС зловмисним програмним забезпеченням, який визначається на основі рівнів загроз станів цього програмного модуля, та визначимо його за формулою (2.8):

$$R_{z,j} = \sum_{s=1}^{8} k_{s,j} \cdot\ p_{s,j}\ , \tag{2.8}$$

де $R_{z,j}$ – рівень загрози ПМ; $k_{s,j}$ – коефіцієнт загрози бути ураженим ЗПЗ s – го стану ПМ, значення якого встановлюється з відрізку $[0; 1]$ в залежності від того які функційні навантаження закладено у певний s – стан; z – позначення загрози; $p_{s,j}$ – ймовірність бути ураженим ЗПЗ. Рівень безпеки програмного модуля виразимо через його рівень загрози за формулою (2.9):

$$R_{b,j} = 1 - R_{z,j}, \tag{2.9}$$

де $R_{b,j}$ – рівень безпеки ПМ; b – позначення безпеки; $R_{z,j}$ – рівень загрози ПМ.

Рівень безпеки РБС визначимо за формулою (2.10):

$$R_{b,\text{РБС}} = \frac{\sum_{l=1}^{n} R_{b,l}}{n}, \tag{2.10}$$

де $R_{b,\text{РБС}}$ – рівень безпеки РБС; b – позначення безпеки; l – ий програмний модуль РБС; n – кількість модулів РБС; $R_{b,l}$ – рівень безпеки ПМ. Відповідно рівень безпеки РБС на основі співвідношень (2.8) та (2.9) можна виразити за

формулою (2.11):

$$R_{b,\text{РБС}} = \frac{\sum_{l=1}^{n}(1-R_{z,l})}{n} = 1 - \frac{\sum_{l=1}^{n} R_{z,l}}{n} = 1 - R_{z,\text{РБС}}, \quad (2.11)$$

де $R_{b,\text{РБС}}$ – рівень безпеки РБС; b – позначення безпеки; z – позначення загрози; l – ий програмний модуль РБС; n – кількість модулів РБС; $R_{z,\text{РБС}}$ – рівень загрози РБС; $R_{z,l}$ – рівень загрози ПМ. Зведемо формули (2.11) та (2.8) підстановкою і отримаємо деталізовану формулу (2.12) для обчислення рівня безпеки РБС:

$$R_{b,\text{РБС},1} = \frac{\sum_{l=1}^{n}(1-\sum_{s=1}^{m} k_{s,l}\cdot p_{s,l})}{n}, \quad (2.12)$$

де $R_{b,\text{РБС},1}$ – рівень безпеки РБС, визначений на першому етапі; b – позначення безпеки; l – номер програмного модуля РБС; n – кількість програмних модулів РБС; $k_{s,l}$ – коефіцієнт загрози (табл. 2.11) бути ураженим ЗПЗ s – го стану ПМ, значення якого встановлюється з відрізку $[0;1]$ в залежності від того, які функційні навантаження закладено у певний s–ий стан; $p_{s,l}$ – ймовірність бути ураженим ЗПЗ; m – кількість станів ПМ.

Отримання ПМ даних від решти активних ПМ РБС для оцінки рівня безпеки системи обробляються в декілька етапів. Першим етапом обробки таких даних є їх представлення одномірним вектором, компонентами якого будуть номери станів програмних модулів РБС, а саме: $p_i = (s_1, s_2, \dots, s_n)$, причому стан самого ПМ теж буде представлений відповідною компонентою вектора. На основі так сформованого вектора здійснюємо первинну швидку оцінку рівня безпеки РБС. Для цього визначаємо кількість програмних модулів РБС за кожним із можливих станів, тобто задаємо функційне відношення формулою (2.13):

$$f(i) = \sum_{\substack{j=1, \\ s_j=i}}^{n} s_j, \quad (2.13)$$

де i – кількість станів ПМ; n – кількість ПМ РБС; s_j – число, яке відображає поточний стан ПМ.

Відповідно аналіз значень функції $f(i)$ показує рівень безпеки РБС на першому етапі оцінювання. Наприклад, якщо $f(1) = n$, тобто всі модулі РБС перебувають в стані 1, що означає відсутність проявів ЗПЗ, тоді рівень безпеки РБС найвищий. Також, якщо $f(1) + f(8) = n$, тобто всі модулі РБС перебувають в станах 1 або 8, що означає відсутність проявів ЗПЗ, тоді рівень безпеки РБС теж найвищий. При найвищих рівнях безпеки подальша перевірка для уточнення рівня безпеки системи в цілому або окремих її частин не проводиться. Якщо виявиться, що різні ПМ РБС перебувають в різних станах одночасно, тоді здійснюється виділення ПМ, стани яких вважаються критичними до ураження, а відповідно і рівень безпеки їх КС нижчий, і проводиться подальше дослідження системи в цілому та виділених її компонентів. В табл. 2.10 представлено приклад можливого згрупованого за станами розподілу.

Таблиця 2.10

Визначення згрупованого розподілу

Номер стану	Кількість ПМ в певному стані
1	$f(1) = \sum_{j=1,}^{n} s_j$, для всіх $s_j = 1$
2	$f(2) = \sum_{j=1,}^{n} s_j$, для всіх $s_j = 2$
3	$f(3) = \sum_{j=1,}^{n} s_j$, для всіх $s_j = 3$
4	$f(4) = \sum_{j=1,}^{n} s_j$, для всіх $s_j = 4$
5	$f(5) = \sum_{j=1,}^{n} s_j$, для всіх $s_j = 5$
6	$f(6) = \sum_{j=1,}^{n} s_j$, для всіх $s_j = 6$
7	$f(7) = \sum_{j=1,}^{n} s_j$, для всіх $s_j = 7$
8	$f(8) = \sum_{j=1,}^{n} s_j$, для всіх $s_j = 8$

Для проведення статистичної обробки компонент одномірного вектора

задамо коефіцієнти $k_{s,j}$ загрози бути ураженим ЗПЗ для кожного стану ПМ, які залежатимуть від того, який тип ЗПЗ вони пробують виявити. Якщо прийняти, стани в яких не буде здійснюватись виявлення ЗПЗ такими, що не представляють загрози об'єктам КС і системі в цілому, тоді їх коефіцієнт дорівнює нулеві. Такими станами розробленої системи є стани один і вісім. Для решти станів, в яких буде здійснюватись пошук ЗПЗ, розподіл коефіцієнтів проведемо в залежності від складності типу ЗПЗ. Наприклад, представимо в табл. 2.11 можливі коефіцієнти загроз для різних станів ПМ.

Таблиця 2.11

Коефіцієнти загроз для різних станів

Рівні загроз станів	Стани ПМ, s	Коефіцієнти загроз	Унормовані значення коефіцієнтів загроз, $k_{s,j}$
0	1, 8	0	0
1	2, 3	0,2	1/15
2	4	0,4	2/15
3	5	0,6	1/5
4	6	0,8	4/15
5	7	1	1/3

Отримавши всі значення для розрахунків, кожен програмний модуль окремо здійснює розрахунки для першого етапу. На основі значень компонент вектору з табл. 2.10 розраховується середнє значення, дисперсія та середньоквадратичне відхилення. Середнє значення для РБС на основі сукупності станів ПМ знаходимо використовуючи данні табл. 2.10 за формулою (2.14):

$$s_{\text{с,РБС}} = \frac{\sum_{s=1}^{8} s \cdot f(s)}{\sum_{s=1}^{8} f(s)}. \quad (2.14)$$

Дисперсію значень для РБС знаходимо за формулою (2.15):

$$D_{\text{с,РБС}} = \frac{\sum_{s=1}^{8} (s - s_{\text{с,РБС}})^2 \cdot f(s)}{\sum_{s=1}^{8} f(s)}. \quad (2.15)$$

Середньоквадратичне відхилення для РБС знаходимо за формулою (2.16):

$$\sigma_{\text{РБС}} = \sqrt{D_{\text{с,РБС}}} \ . \tag{2.16}$$

За результатами таких обчислень програмні модулі вибирають частину ПМ РБС, які перебувають в тих станах, відхилення від середнього значення до яких найменше, і встановлюють їх кількість k та фіксують, які саме ПМ формують центр РБС в поточний момент $A_{\text{ц,РБС}} = \{\cup A_j \mid A_j \subseteq A,\ j = 1, 2, \dots n,\ |\text{стан}(A_j) - s_{\text{с,РБС}}| <= \sigma_{\text{РБС}}\}$, тобто переважно які задачі вирішує система. Якщо виявиться, що РБС працює над виявленням ЗПЗ одного типу на декількох різних КС і таких КС досліджується багато в порівнянні з усіма активними в поточний момент, а також якщо при цьому ще рівень безпеки є низьким, тоді РБС здійснює перехід до глибшого дослідження процесів в КС локальної мережі шляхом опрацювання інформації з їх ПМ про час перебування в кожному із станів з моменту останньої активації. Якщо центр РБС складається переважно з станів, в яких не здійснюється виявлення ЗПЗ, а вирішуються інші задачі, тоді РБС може досліджувати ті КС, які не увійшли до центру, або продовжувати роботу на основі задач, які вирішують її ПМ окремо. Враховуючи необхідність досягнення однозначності у визначенні подальших дій для РБС після першого етапу досліджень свого стану і знаходження центру в поточний момент, визначимо на основі результатів з формування центру подальші дії РБС за такою функцією, представленою формулою (2.17):

$$g\left(R_{b,\text{РБС}}, k, s, s_{\text{с,РБС}}\ \right) = \begin{cases} 0, & \text{якщо виконується умова 1} \\ 1, & \text{якщо виконується умова 2} \\ 2, & \text{якщо виконується умова 3} \end{cases}. \tag{2.17}$$

Враховуючи, що умови для задання функції g є складеними, то представимо їх в табл. 2.12.

Таблиця 2.12

Значення функції $g\left(R_{b,\text{РБС}}, k, s, s_{\text{с,РБС}}\right)$

Умови для функції g	Значення умов для функції g		
	Значення рівня безпеки РБС $R_{b,\text{РБС}}$	Умови, які пов'язані з кількістю ПМ	Умови, які пов'язані з середньоквадратичним відхиленням
1	2	3	4
Умова 1	$R_{b,\text{РБС}} > 0{,}75$	$\frac{k}{n} > 0{,}5$	$\min(\frac{s_{\text{с,РБС}}}{s} - 1) < 1$ для $s = 1$ або для $s = 8$
	$R_{b,\text{РБС}} > 0{,}75$	$\frac{k}{n} > 0{,}5$	$\min(\frac{s_{\text{с,РБС}}}{s} - 1) < 1$ для $s = 2$ або для $s = 3$ або $s = 4$
	$R_{b,\text{РБС}} > 0{,}75$	$\frac{k}{n} < 0{,}5$	$\min(\frac{s_{\text{с,РБС}}}{s} - 1) < 1$ для $s = 5$ або для $s = 6$ або $s = 7$
	$0{,}5 < R_{b,\text{РБС}} < 0{,}75$	$\frac{k}{n} < 0{,}5$	$\min(\frac{s_{\text{с,РБС}}}{s} - 1) < 1$ для $s = 2$ або для $s = 3$
Умова 2	$0{,}5 < R_{b,\text{РБС}} < 0{,}75$	$\frac{k}{n} > 0{,}5$	$\min(\frac{s_{\text{с,РБС}}}{s} - 1) < 1$ для $s = 2$ або для $s = 3$
	$0{,}5 < R_{b,\text{РБС}} < 0{,}75$	$\frac{k}{n} < 0{,}5$	$\min(\frac{s_{\text{с,РБС}}}{s} - 1) < 1$ для $s = 2$ або для $s = 3$ або $s = 4$
	$0{,}25 < R_{b,\text{РБС}} < 0{,}5$	$\frac{k}{n} > 0{,}5$	$\min(\frac{s_{\text{с,РБС}}}{s} - 1) < 1$ для $s = 1$ або для $s = 8$

Продовження таблиці 2.12

1	2	3	4
Умова 3	$R_{b,\text{РБС}} > 0{,}75$	$\frac{k}{n} > 0{,}5$	$\min(\frac{s_{\text{с,РБС}}}{s} - 1) < 1$ для $s = 6$ або для $s = 7$
	$0{,}5 < R_{b,\text{РБС}} < 0{,}75$	$\frac{k}{n} > 0{,}5$	$\min(\frac{s_{\text{с,РБС}}}{s} - 1) < 1$ для $s = 5$ або для $s = 6$ або для $s = 7$
	$0{,}25 < R_{b,\text{РБС}} < 0{,}5$	$\frac{k}{n} > 0{,}5$	$\min(\frac{s_{\text{с,РБС}}}{s} - 1) < 1$ для $s = 4$ або для $s = 5$ або для $s = 6$ або для $s = 7$
	$0{,}25 < R_{b,\text{РБС}} < 0{,}5$	$\frac{k}{n} < 0{,}5$	$\min(\frac{s_{\text{с,РБС}}}{s} - 1) < 1$ для $s = 5$ або для $s = 6$ або для $s = 7$
	$R_{b,\text{РБС}} < 0{,}25$	-	-
	інші випадки		

Всього таких випадків може бути 64, бо є чотири випадки для рівня безпеки, два випадки для кількості ПМ, які входять до центру РБС в поточний момент часу, вісім випадків для віднесення центру до одного зі станів за рахунок дослідження його відхилення.

При виконанні умови 1, тобто якщо $g\left(R_{b,\text{РБС}}, k, s, s_{\text{с,РБС}}\right) = 0$, РБС продовжує роботу в режимі коли її ПМ працюють в тих станах, в яких були. При цьому жодних дій по обробці ситуацій в певних відібраних КС не проводиться.

При виконанні умови 2, тобто якщо $g\left(R_{b,\text{РБС}}, k, s, s_{\text{с,РБС}}\right) = 1$, РБС продовжує роботу в режимі коли її ПМ працюють в тих станах, в яких були. А також, РБС зразу відмічає програмні модулі для яких потрібне додаткове

уточнення стосовно задач, які виконують в поточний момент часу.

При виконанні умови 3, тобто якщо $g\left(R_{b,\text{РБС}}, k, s, s_{\text{с,РБС}}\right) = 2$, РБС переходить до другого етапу уточнення свого стану на основі залучення часових характеристик станів всіх ПМ.

Представимо результати в поточний момент часу з програмних модулів РБС з врахуванням перебування в станах та час перебування в кожному стані кожного ПМ в табл. 2.13.

Таблиця 2.13

Результати перебування ПМ в поточні моменти часу

	Стани																$\sum$ часу
	1		2		3		4		5		6		7		8		
ПМ	стан	час	стан	час	стан	час	стан	час	стан	час	стан	час	стан	час	стан	час	
1	$w_{1,1}$	$t_{1,1}$	$w_{1,2}$	$t_{1,2}$	$w_{1,3}$	$t_{1,3}$	$w_{1,4}$	$t_{1,4}$	$w_{1,5}$	$t_{1,5}$	$w_{1,6}$	$t_{1,6}$	$w_{1,7}$	$t_{1,7}$	$w_{1,8}$	$t_{1,8}$	
…																	
n	$w_{n,1}$	$t_{n,1}$	$w_{n,2}$	$t_{n,2}$	$w_{n,3}$	$t_{n,3}$	$w_{n,4}$	$t_{n,4}$	$w_{n,5}$	$t_{n,5}$	$w_{n,6}$	$t_{n,6}$	$w_{n,7}$	$t_{n,7}$	$w_{n,8}$	$t_{n,8}$	
$\sum$																	

Результати представлені в табл. 2.13 задамо матрицями: W – матриця станів кожного програмного модуля РБС від початку поточного запуску, T – матриця відображення для програмних модулів часу перебування в кожному стані. Зокрема, $w_{1,1}$ – кількість перебувань ПМ з номером один в першому стані, $w_{i,s}$ - кількість перебувань ПМ з номером i в стані s, $i = 1, 2, \ldots, n$, $s = 1, 2, \ldots, 8$, $t_{i,s}$ – сумарний час перебування ПМ з номером i в стані s. На відміну від одновекторного представлення ПМ РБС, яке надає данні лише поточних станів програмних модулів, таке представлення через матриці W та T надає інформацію про програмні модулі за увесь час їх роботи від моменту останнього ввімкнення КС. Така інформація дозволить оцінити час витрачений на задачі, які вирішувались ПМ системи і відповідно РБС в цілому.

Для оцінки стану загрози розглядатимемо результати безпеки всіх ПМ РБС

через їх стани представлені матрицею $P_{s,\text{РБС}}$ згідно формули (2.18):

$$P_{s,\text{РБС}} = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,8} \\ \cdots & \cdots & \cdots & \cdots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,8} \end{pmatrix}, \quad (2.18)$$

де $p_{s,j}$ – ймовірність бути ураженим ЗПЗ; s – номер стану ПМ; $s = 1, \dots, 8$; $j = 1, 2, \dots, n$; n – кількість ПМ РБС.

Значення $p_{s,j}$ отримуються на основі результатів функціонування закладених в програмні модулі підсистем виявлення певних типів ЗПЗ.

Визначимо суми елементів рядків, стовпців та всіх відповідно з сформованих матриць $P_{s,\text{РБС}}$, W та T, які представлені в табл. 2.14 і формулою (2.9), за формулами (2.19) – (2.24):

$$W_{j,\text{ПМ}} = \sum_{s=1,}^{8} w_{s,j}, \quad (2.19)$$

$$W_{s,\text{Стан}} = \sum_{j=1,}^{n} w_{s,j}, \quad (2.20)$$

$$T_{j,\text{ПМ}} = \sum_{s=1,}^{8} t_{s,j}, \quad (2.21)$$

$$T_{s,\text{Стан}} = \sum_{j=1,}^{n} t_{s,j}, \quad (2.22)$$

$$P_{j,\text{ПМ}} = \sum_{s=1,}^{8} p_{s,j}, \quad (2.23)$$

$$P_{s,\text{Стан}} = \sum_{j=1,}^{n} p, \quad (2.24)$$

де $W_{j,\text{ПМ}}$ – сума кількостей станів ПМ протягом часу від останнього запуску, причому ПМ може потрапляти в деякі стани багато разів; $W_{s,\text{Стан}}$ – сума кількостей в стані s протягом роботи РБС від початку останнього запуску, тобто кількість перебувань РБС в стані s з врахування перебування у всіх активних програмних модулях; $T_{j,\text{ПМ}}$ – час роботи ПМ від його останнього запуску; $T_{s,\text{Стан}}$ – сума часу усіх модулів РБС від моменту останнього запуску; $P_{j,\text{ПМ}}$ – сума ймовірностей бути ураженим ЗПЗ певного j – того модуля РБС; $P_{s,\text{Стан}}$ – сума ймовірностей бути ураженим ЗПЗ в стані s.

Рівень безпеки РБС в цілому на цьому етапі її дослідження визначимо як комплексну величину, що залежатиме від кількох, обчислених за різними критеріями, рівнів безпеки РБС, визначення кожного з яких здійснимо за окремим критерієм.

Нехай вважатимемо, що ймовірності бути ураженим ЗПЗ впливатимуть не тільки на програмні модулі або їх вплив на ці модулі несуттєвий, що не дозволяє визначити стан РБС як критичний. Такий випадок можливий, коли дослідження на першому етапі через визначення середнього значення і його подальшого використання було невисоким через невеликий час роботи від останнього запуску ПМ чи через необхідність його усереднення для восьми станів. Але може виявитись, що багато програмних модулів тривалий час перебувають чи перебували в одному і тому ж стані, а використання критеріїв першого етапу їх не виділяє. Тому, щоб врахувати такі граничні особливості необхідно виділити ймовірності бути ураженим ЗПЗ для РБС у певних визначених станах і розробити критерії для оцінки таких випадків. Введемо для кожного стану кожного модуля величину, яка буде визначатись через ймовірність ураження, і задамо її так: $q_{s,j} = 1 - p_{s,j}$ для всіх модулів і станів. Тоді, ймовірність бути ураженою в стані s для РБС залежить від ймовірностей бути ураженими в стані s для програмних модулів. Події перебування частини програмних модулів РБС в одному і тому ж стані є сумісними подіями і при цьому перебування одного модуля в тому ж стані що і інший є незалежними, бо немає впливу одного модуля на інший. Вплив може проявлятись тільки при вирішенні спільних завдань, але перехід до стану вирішується програмним модулем за потреби і не залежить від вказівок від інших модулів. Тому, значення ймовірність бути ураженою в стані s для РБС виразимо за формулою (2.25):

$$P_{s,\text{РБС}} = 1 - \prod^{n}_{\substack{j=1, \\ q_{s,j}>0}} q_{s,j}, \tag{2.25}$$

де $P_{s,\text{РБС}}$ - ймовірність бути ураженою в стані s для РБС.

Тоді, визначимо ймовірність бути ураженою для РБС на основі ймовірностей її станів за формулою (2.26):

$$P_{\text{РБС},1} = \sum_{s=1}^{8}(P_{s,\text{РБС}} \cdot k_s), \quad (2.26)$$

де $P_{\text{РБС},1}$ - ймовірність бути ураженою для РБС на основі ймовірностей її станів.

Важливою характеристикою, яка впливає на рівень безпеки РБС є час перебування в критичних станах. Якщо час перебування в певному стані дуже тривалий по відношенню до часу інших станів та часу ПМ в цілому, тоді така подія повинна бути врахована при оцінці поточного рівня безпеки РБС. Елементи матриці W можуть бути більшими одиниці, бо ПМ може декілька разів перебувати в одному й тому ж стані, тобто повертатись в нього багатократно. Врахування кількості перебувань ПМ, як компоненти РБС, характеризуватиме потребу РБС вирішувати одне і те ж завдання через свій ПМ. Тому, здійснимо нормування елементів матриці W порядково, тобто встановимо співвідношення між кількістю перебувань ПМ в своїх станах. Здійснимо унормування елементів в рядках за формулою (2.27):

$$w'_{\text{ПМ},s,j} = \frac{w_{s,j}}{\sum_{s=1}^{8} w_{s,j}}, \quad (2.27)$$

де $w'_{\text{ПМ},s,j}$ - унормоване значення $w_{s,j}$ s – го стану в j – му рядку, тобто в j – му ПМ.

Аналогічно з матрицею W, виконаємо такі ж дії по унормуванню часу перебування для кожного програмного модуля в певному його стані. Здійснимо унормування елементів в рядках за формулою (2.28):

$$t'_{\text{ПМ},s,j} = \frac{t_{s,j}}{\sum_{s=1}^{8} t_{s,j}}, \quad (2.28)$$

де $t'_{\text{ПМ},s,j}$ – унормоване значення $t_{s,j}$ s – го стану в j –му рядку, тобто в j – му ПМ.

Ймовірність бути ураженим для ПМ і системи в цілому на основі її

компонентів з врахуванням часу перебування в певному стані відносно кожного її ПМ та відповідно кількості перебувань задамо формулою (2.29):

$$P_{\text{РБС},2} = \frac{\sum_{j=1}^{n}\sum_{\substack{s=1,\\ t'_{\text{ПМ},s,j}>0,\\ w'_{\text{ПМ},s,j}>0}}^{8} (t'_{\text{ПМ},s,j} \cdot w'_{\text{ПМ},s,j})}{n}, \quad (2.29)$$

де $P_{\text{РБС},2}$ – ймовірність бути ураженою для РБС на основі ймовірностей сформованих часом перебування в певних станах та кількостей перебувань в розрізі програмних модулів РБС. Обчислення добутку для певного стану здійснюється за умови, що ПМ хоча б один раз перебував в ньому. Якщо ж ПМ був в певному стані хоча б один раз, тоді і час його перебування буде більше нуля, що необхідно для отримання значення доданку. В знаменнику знаходиться число, що дорівнює кількості активних ПМ в РБС в поточний момент часу. Оскільки ПМ активні, тоді вони мають показники часу і кількості перебування в певних станах, значення яких не нулеві.

Час перебування РБС в певному стані на основі сумарної величини перебування її компонент в цьому ж стані є важливою характеристикою, яка в залежності від рівня загрози бути ураженим у визначених станах дозволяє визначити ймовірність бути ураженим для РБС на основі часу перебування в станах. З метою визначення числової величини данної характеристики здійснимо обчислення сум стовпців матриці W та матриці T і проведемо їх унормування за формулами (2.30) – (2.35):

$$w_{s,\text{Стан}} = \sum_{j=1}^{n} w_{s,j}, \quad (2.30)$$

$$t_{s,\text{Стан}} = \sum_{j=1}^{n} t_{s,j}, \quad (2.31)$$

$$w_{s,\text{РБС}} = \sum_{s=1}^{8} w_{s,\text{Стан}}, \quad (2.32)$$

$$t_{s,\text{РБС}} = \sum_{s=1}^{8} t_{s,\text{Стан}}, \quad (2.33)$$

$$w'_{s,\text{Стан}} = \frac{w_{s,\text{Стан}}}{w_{s,\text{РБС}}}, \quad (2.34)$$

$$t'_{s,\text{Стан}} = \frac{t_{s,\text{Стан}}}{t_{s,\text{РБС}}}, \quad (2.35)$$

де $w_{s,\text{Стан}}$ і $t_{s,\text{Стан}}$ – кількість перебувань та час в певному стані s відповідно; $w_{s,\text{РБС}}$ і $t_{s,\text{РБС}}$ – кількість перебувань та час в станах для всієї РБС; $w'_{s,\text{Стан}}$ і $t'_{s,\text{Стан}}$ – унормовані значення кількості перебувань та часу в певному стані s відповідно.

Тоді, визначимо характеристику $P_{\text{РБС,3}}$ ймовірності бути ураженим для РБС на основі врахування часу перебування в певному стані та відповідно кількості перебувань в кожному із станів за формулою (2.36):

$$P_{\text{РБС,3}} = \sum_{s=1}^{8} w'_{s,\text{Стан}} \cdot t'_{s,\text{Стан}}. \quad (2.36)$$

А також, визначимо характеристику $P_{\text{РБС,4}}$ ймовірності бути ураженим для РБС на основі врахування часу перебування в певному стані та відповідно кількості перебувань в кожному із станів за формулою (2.37) з використанням вагових коефіцієнтів станів:

$$P_{\text{РБС,4}} = \sum_{s=1}^{8} w'_{s,\text{Стан}} \cdot t'_{s,\text{Стан}} \cdot k_s. \quad (2.37)$$

На основі числових величин характеристик РБС знайдемо інтегроване значення рівня її безпеки за формулою (2.38), вважаючи частку кожної з характеристик рівноцінною:

$$R_{b,\text{РБС,2}} = \frac{\sum_{i=1}^{4} P_{\text{РБС},i}}{4}, \quad (2.38)$$

де $R_{b,\text{РБС}}$ – інтегрований рівень безпеки РБС, який отриманий на другому етапі дослідження; b – позначення безпеки.

Деталізоване представлення на основі формули (2.38) інтегрованого значення рівня безпеки РБС задано і визначається за формулою (2.39):

$$R_{b,\text{РБС},2} = \frac{1}{4}\cdot\left(\sum_{s=1}^{m}\left(1-\prod_{\substack{j=1,\\ p_{s,j}<1}}^{n}\left(1-p_{s,j}\right)\right)\cdot k_s + \sum_{j=1}^{n}\sum_{\substack{s=1,\\ t_{s,j}>0,\\ w_{s,j}>0}}^{m}\left(\frac{t_{s,j}}{\sum_{s=1}^{m} t_{s,j}}\cdot \frac{w_{s,j}}{\sum_{s=1}^{m} w_{s,j}}\right) + \sum_{s=1}^{m}\left((1+k_s)\cdot\frac{\sum_{j=1}^{n} w_{s,j}}{\sum_{s=1}^{m}\sum_{j=1}^{n} w_{s,j}}\cdot\frac{\sum_{j=1}^{n} t_{s,j}}{\sum_{s=1}^{m}\sum_{j=1}^{n} t_{s,j}}\right)\right), \quad (2.39)$$

де $R_{b,\text{РБС},2}$ – рівень безпеки РБС, визначений на другому етапі; b – позначення безпеки; s – номер програмного модуля РБС; n – кількість програмних модулів РБС; m – кількість станів ПМ; k_s – коефіцієнт загрози бути ураженим ЗПЗ s – того стану ПМ, значення якого встановлюється з відрізку $[0;1]$ в залежності від того, які функційні навантаження закладено у певний s–ий стан; $p_{s,j}$ – ймовірність бути ураженим ЗПЗ; $w_{s,j}$ - кількість перебувань ПМ з номером j в стані s; $i = 1, 2, \dots, n$; $s = 1, 2, \dots, m$; $t_{s,j}$ – сумарний час перебування ПМ з номером j в стані s n – кількість ПМ РБС.

Значення $p_{s,j}$ отримуються на основі результатів функціонування закладених в програмні модулі підсистем виявлення певних типів ЗПЗ.

Використання розробленого методу [353, 364, 365, 374, 386] дозволяє організовувати підтримку цілісності системи та здійснення передачі знань, отриманих окремими структурними компонентами системи програмними модулями іншим компонентам. Отримані аналітичні залежності рівнів безпеки програмних модулів і РБС дозволяють здійснювати зміну архітектури РБС динамічно та впливати на подальші її дії без втручання користувача. Розроблений метод є основою для розробки зв'язуючої частини програмного забезпечення розподіленої багаторівневої системи виявлення ЗПЗ.

Висновки до другого розділу

Запропоновані принципи та моделі розробки розподілених систем є важливими для теорії і практики створення ефективних систем виявлення

зловмисного програмного забезпечення в локальних комп’ютерних мережах, побудованих на основі децентралізації та самоорганізації.

Архітектура розподіленої багаторівневої системи [172, 173, 363, 364, 385] дозволяє здійснювати її наповнення різними функціоналами виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах. РБС відноситься до реагуючих систем, яка постійно здійснюватиме моніторинг запущених процесів та виконуваних програм в локальних комп’ютерних мережах. Об’єктами для дослідження зі сторони РБС є наявне програмне забезпечення та запущені процеси в КС локальної мережі на можливість віднесення до зловмисного програмного забезпечення. Удосконалена модель архітектури розподіленої системи виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах базується на комплексному врахуванні вимог розподіленості, децентралізованості, багаторівневості та самоорганізованості, що дозволяє створювати на її основі розподілені системи та їх компоненти, які функціонуватимуть автономно і самостійно прийматимуть рішення про наявність зловмисного програмного забезпечення та нарощення своїх функціональних можливостей.

Розроблена модель [364, 380] архітектури типових компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення на основі структур Кріпке з представленням компонентів через стани, в яких вони можуть перебувати під час функціонування, дає змогу враховувати перебування їх в різних станах і є основою для визначення стану безпеки всієї розподіленої системи та її компонентів. Вона дозволяє здійснювати збільшення кількості рівнів системи без зміни її архітектури. Основою архітектури РБС виступають програмні модулі з однаковими архітектурами, але при цьому кожен з них може самостійно приймати рішення на основі різних даних зібраних з різних КС локальної мережі.

Розроблений метод взаємодії [353, 364, 365, 374, 386] компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення на основі підтримки її цілісності та визначення порядку передачі

знань між її компонентами і використання встановлених аналітичних залежностей між рівнями безпеки програмних модулів та рівнем безпеки всієї розподіленої багаторівневої системи, що дозволяє системі автономно змінювати свою архітектуру та функції без втручання користувача, а також визначати стратегію своєї подальшої роботи. Метод є основою для розробки зв’язуючої частини програмного забезпечення, яка організовує взаємодію компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах.

Основні результати розділу опубліковані у працях [163, 172, 173, 201–203, 223, 227, 313, 337, 363–365, 367, 368, 370, 371, 380, 383–385].

РОЗДІЛ 3
ТЕОРЕТИЧНІ І МЕТОДОЛОГІЧНІ ОСНОВИ ПОБУДОВИ СИСТЕМ ВИЯВЛЕННЯ ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ

Функціонування зловмисного програмного забезпечення здійснюється в розподілених обчислювальних системах комп'ютерних мереж. Для розробки нових моделей та методів виявлення зловмисного програмного забезпечення необхідним є формалізація безпосередньо зловмисного програмного забезпечення, інших програмних об'єктів, а також апаратних засобів комп'ютерних систем та мереж, як єдиного інформаційного простору.

3.1. Формалізація характерних властивостей зловмисного програмного забезпечення на основі теорії алгебр

Інформаційний простір містить такі основні компоненти: інформаційні ресурси, засоби інформаційної взаємодії, інформаційну інфраструктуру. Іншими словами, необхідно формалізувати компоненти інформаційного простору з метою представлення зловмисного програмного забезпечення і середовища їх функціонування для подальшої побудови моделей зловмисного програмного забезпечення і розробки методів та засобів їх виявлення.

Інформаці́йна інфраструкту́ра включає програмно-технічні засоби комп'ютерних мереж, які забезпечують організацію взаємодії інформаційних потоків, функціонування та розвиток засобів інформаційної взаємодії та інформаційного простору організації. В якості середовища функціонування зловмисного програмного забезпечення і його виявлення розглядатимемо корпоративну мережу, яка складається з групи локальних мереж організації. Середовищем перебування зловмисного програмного забезпечення в корпоративній мережі будуть засоби пам'яті та процесори. Тому, необхідно при формалізації інформаційного простору враховувати ці апаратні засоби.

Корпоративна мережа [393] є частиною відкритого інформаційного простору і, як правило, є частиною Internet простору. Зловмисне програмне забезпечення може функціонувати і поширюватись в одній комп'ютерній системі, при розширенні обчислювальних ресурсів до локальної комп'ютерної мережі – в комп'ютерних системах локальної мережі і при розширенні обчислювальних ресурсів до глобальних комп'ютерних мереж – в комп'ютерних системах глобальних мереж. Таким чином, інформаційних простір, в якому може функціонувати ЗПЗ має підпростори на рівні корпоративних мереж та на рівні окремих комп'ютерних систем, де зберігаються ті ж можливості для розвитку і поширення ЗПЗ, які є на рівні глобальної мережі. Враховуючи можливість порівняння результатів функціонування на поширення ЗПЗ в різних комп'ютерних системах в локальній мережі на відміну від розгляду тільки однієї комп'ютерної системи, тоді найменшим підпростором будемо вважати корпоративну мережу і результати отримані в ній можна поширити на фрагменти всього простору тобто глобальної мережі, розглядаючи її як складену з локальних мереж.

Позначимо множину всього зловмисного програмного забезпечення V, яке перебуває в комп'ютерних системах локальних мереж. Тобто розглядатимемо те ЗПЗ, яке за певних обставин та на протязі певного часу експлуатації локальних комп'ютерних мереж, проникло в комп'ютерні системи, змогло пройти певні системи захисту і функціонує там. Представимо ЗПЗ у локальних комп'ютерних мережах, особливістю якого є втілення у виконувані файли, завантажувальний сектор жорсткого диску, оперативний запам'ятовуючий пристрій та поширення мережею своїх копій, алгебраїчною системою типу $\tau = (\alpha,\ \beta)$ за формулою (3.1):

$$\mathfrak{A}_V = \langle V,\ \ \Omega_F,\ \ \Omega_P \rangle, \tag{3.1}$$

де $\Omega_F = \{F_0,\ F_1,\ F_2, \dots, F_{\alpha_1}, \dots\}$ – множина операцій заданих на множині V для кожного $\alpha_1 = 0, 1, 2, \dots$; $\Omega_P = \{P_0,\ P_1,\ P_2, \dots, P_{\beta_1}, \dots\}$ – множина предикатів

заданих на множині V для кожного $\beta_1 = 0, 1, 2, ...;\ \alpha = 1,\ \beta = 1$ – арності операцій, тому тип системи $\tau = (1,\ 1)$.

Елементами множини $v_j \in V\ \ (j = 1, 2, ...)$ вважатимемо всі об'єкти файлової системи, завантажувального сектору диску, оперативної пам'яті, мережні пакети, які відносяться до розглядуваного ЗПЗ. Елементи $v_0 \in V_0 \subseteq V$ є одиничними елементами, тобто такими що містять єдиний функціонал, вміст якого полягає у необхідності здійснення самокопіювання з метою поширення, але без конкретного функціонального наповнення для виконання технічно цих дій. Решта операцій представлені іншими функціями. Ці елементи, що формують множину V_0 є породжуючими для решти різних елементів множини V. Функції з множини Ω_F виконуються на елементах v_0, що формує інші об'єкти, які належатимуть множині V, а також можуть виконуватись на інших елементах множини V, які не належать множині V_0. Функції з множини Ω_F не завжди успішно виконуватимуться по відношенню до елементів з множини V, тому для представлення ЗПЗ в локальних мережах вибрано також множину предикатів, яка відображатиме результат успішного/неуспішного виконання функцій.

Функції $F_{\alpha_1}\ (\alpha_1 = 0, 1, 2, ...)$ з множини Ω_F визначимо, як такі що здійснюватимуть відображення елементів з множини V на неї. Їх конкретне визначення залежатиме від поділу множини Ω_F на підмножини за різними характеристичними властивостями ЗПЗ. Предикати $P_{\beta_1}\ (\beta_1 = 0, 1, 2, ...)$ з множини Ω_P визначимо, як такі що будуть істинними при успішному виконанні операцій і хибними - в іншому випадку.

Множину Ω_F представимо її підмножинами $\Omega_{F_{s,t}}$, які відображатимуть такі характеристичні для ЗПЗ властивості та закладені в його функціонал особливості:

1) зберігання знань про механізм місцерозміщення своїх наступних копій;

2) пошук місця в пам'яті для розміщення своєї копії;

3) знання про механізми втілення у виконувані програми;

4) механізми запису в оперативну пам'ять;

5) приховування свого перебування в комп’ютерних системах;

6) пошук інших вузлів мережі для свого поширення;

7) механізми для формування і відправки мережних пакетів;

8) подолання механізмів захисту;

9) техніки запису своїх копій в головний завантажувальний сектор;

10) виконання деструктивних дій.

Ці характеристичні властивості ЗПЗ пов’язані з системними викликами, які відносяться до роботи з файлами, оперативною пам’яттю та командами роботи в мережі: створення, відкриття, закриття, видалення, читання, записування, додавання, знаходження, отримання атрибутів і встановлення атрибутів, команди доступу до ОП, команди для роботи в мережі. Відповідність характеристичних ознак ЗПЗ системним викликам представлено в табл.3.1.

Таблиця 3.1

Характеристичні ознаки ЗПЗ

Характеристичні властивості ЗПЗ	створення	відкриття	закриття	видалення	читання	записування	додавання	знаходження	отримання атрибутів	встановлення атрибутів	команди доступу до ОП	команди для роботи в мережі
1	2	3	4	5	6	7	8	9	10	11	12	13
зберігання знань про механізм місцерозміщення своїх наступних копій								+	+			
пошук місця в пам’яті для розміщення своєї копії					+							

Продовження таблиці 3.1

1	2	3	4	5	6	7	8	9	10	11	12	13
знання про механізми втілення у виконувані програми		+	+	+	+	+						
механізми запису в оперативну пам’ять											+	
приховування свого перебування в комп’ютерних системах											+	
пошук інших вузлів мережі для свого поширення												+
механізми для формування і відправки мережних пакетів												+
подолання механізмів захисту	+	+	+	+	+	+	+	+	+	+	+	+
техніки запису своїх копій в головний завантажувальний сектор	+	+	+	+	+	+						
виконання деструктивних дій	+	+	+	+	+	+	+	+	+	+	+	+

Реалізація характеристичних властивостей ЗПЗ пов’язана з системними викликами та командами для роботи в мережі визначатиме наповнення функцій з множини Ω_F і залежатиме від них, що дозволить ідентифікувати такі дії.

Під новими копіями ЗПЗ вважатимемо збіг ЗПЗ за семантикою, а не тільки за синтаксисом. Тому, при поширенні ЗПЗ у випадку зміни синтаксису важливими є особливості функціоналу незалежно від синтаксису коду.

Місцем можливого розміщення ЗПЗ в комп’ютерних системах локальних комп’ютерних мереж може бути оперативна пам'ять, зовнішня пам'ять та мережні пакети. Представлення місця перебування зловмисного програмного забезпечення в комп’ютерних системах зображено на рис. 3.1. Таке виділення трьох основних складових необхідно для побудови моделей ЗПЗ, які б стали основою для розробки нових методів їх виявлення. Наявність ЗПЗ в зовнішній

пам’яті є характерним для усіх його різновидів. Перебування ЗПЗ в мережних пакетах для частини ЗПЗ є обов’язковим, оскільки характеризує механізми його поширення. Для іншої частини ЗПЗ перебування в мережних пакетах не є обов’язковим, тобто їх поширення може відбуватись іншими шляхами, зокрема і через носії зовнішньої пам’яті. Використання ЗПЗ для свого перебування можливе тільки при ввімкненій працюючій КС. Для певної частини ЗПЗ використання оперативної пам’яті є обов’язковим місцем розміщення та функціонування, а для іншої тільки місцем на час виконання. Врахування в моделях ЗПЗ місця їх перебування є важливим елементом, який може бути використаний при їх виявленні, оскільки розробники ЗПЗ закладають в нього знання про їх місцезнаходження в КС. Тобто ЗПЗ володіє техніками перевірки свого місцезнаходження, що є важливим при розробці його моделей. Елементи місць розміщення ЗПЗ потребуватимуть деталізації в залежності від їх функційного призначення та технічних характеристик, що впливатиме на моделі ЗПЗ і потребуватиме їх деталізації і уточнення.

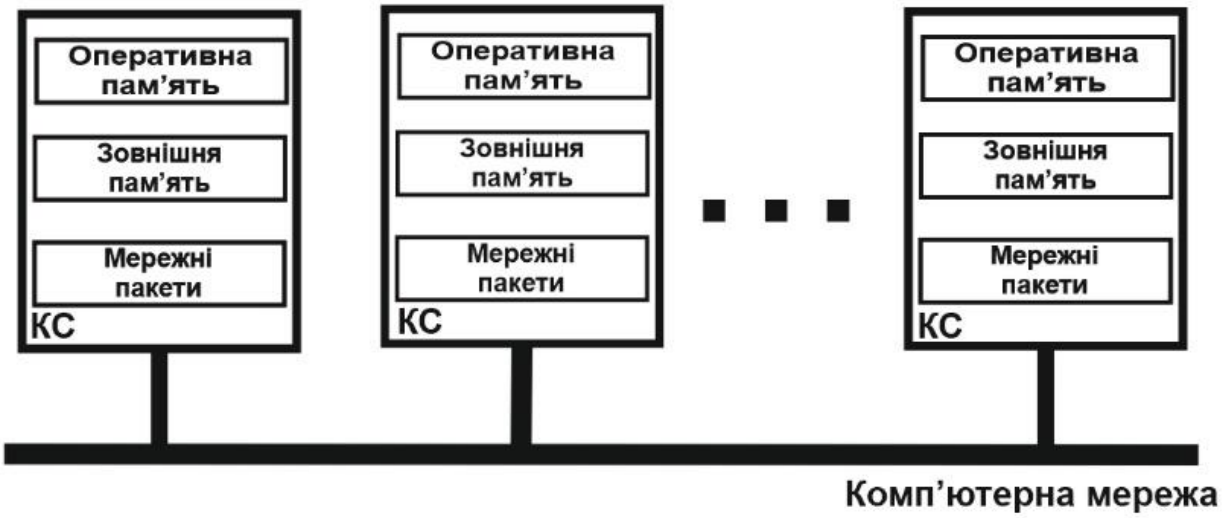


Рис.3.1. Місця перебування ЗПЗ

Розглядаючи ЗПЗ з точки зору його місця розміщення та пошуку ним його для зберігання себе і своїх копій при поширенні, представимо модель ЗПЗ за цією характерною властивістю. Виділимо в множині всіх програм підмножину

зловмисного програмного забезпечення для якого характерна властивість полягає у збереженні закладеної у ЗПЗ інформації про механізми поширення в частині перебування їх в зовнішній пам'яті, оперативній пам'яті і мережних пакетах. Здійснимо її формалізоване представлення для використання в процесі пошуку ЗПЗ.

Виділимо в множині Ω_F підмножини Ω_{F_p} таким чином, щоб $\Omega_F = \bigcup_{p=1}^{k} \Omega_{F_p}$, де k – кількість характеристичних властивостей ЗПЗ. Тоді, алгебраїчну систему для першої властивості, що характеризує знання про місцезнаходження ЗПЗ при p=1 для всієї локальної мережі задамо за формулою (3.2):

$$\mathfrak{A}_{V,1} = \langle V,\ \Omega_{F_1},\ \Omega_{P_1} \rangle, \tag{3.2}$$

де Ω_{F_1}– множина операцій заданих на множині V; Ω_{P_1}– множина предикатів заданих на множині V.

Нехай $V = \bigcup_{s=1}^{n} V_s$, тобто виділимо в кожній КС локальної мережі підмножину ЗПЗ V_s, де $s = 1, 2, \dots, n$. Для виділених підмножин з множини V в момент часу t=0 буде справедливе твердження $\bigcap_{s=1}^{n} V_s = \emptyset$. Дійсно, з самого початку початкового встановлення програмного забезпечення у всі КС мережі в них немає ЗПЗ. В процесі збільшення часу їх роботи, тобто при t > 0, ймовірність появи ЗПЗ певного на різних КС мережі зростає, тому справедливим може бути твердження $\bigcap_{s=1}^{n} V_s \neq \emptyset$. Задамо алгебру для першої властивості, що характеризує знання про місцезнаходження ЗПЗ при p=1 для однієї з КС мережі за формулою (3.3):

$$\mathfrak{B}_{V_{s,1}} = \langle V_s,\ \Omega_{F_1} \rangle, \tag{3.3}$$

де s – кількість вузлів ЛКМ; Ω_{F_1} – множина функцій заданих на множині V, яка впливає на місце розміщення наступних копій ЗПЗ.

Ця множина функцій здійснює відображення копії ЗПЗ в певний об'єкт

зовнішньої пам'яті, оперативного запам'ятовуючого пристрою та мережного пакету. Якщо $v_{s,j,l} \in V_s$, де j – це номер елементу ЗПЗ, l – номер версії j елементу, тоді $F_{1,k} \in \ \Omega_{F_1}$, де k – кількість функцій в множині Ω_{F_1}, k $\in$ N. Ці функції $F_{1,k}$ здійснюють відображення елементу $v_{s,j,l}$ в множину V_s, тобто $F_{1,k}(\ v_{s,j,l}\ ) = \ v_{s,j,l+1}$, де $v_{s,j,l+1} \in V_s$, якщо наступна копія ЗПЗ створюватиметься в тій же КС мережі. Але наступна копія може створюватись і в іншій КС мережі, тому функцію $F_{1,k}$ можна задати за формулою (3.4):

$$F_{1,k}(\ v_{s,j,l}\ ) = \begin{cases} v_{s,j,l+1}, \text{ якщо } v_{s,j,l+1} \in V_s \\ v_{s\prime,j,l\prime}, \text{ якщо } v_{s\prime,j,l\prime} \notin V_s \end{cases}, \tag{3.4}$$

де s' - номер КС в мережі відмінний від s; l' - номер копії ЗПЗ відмінний від l'.

Для функції $F_{1,k}$ існує, також, обернена функція $F_{1,k}^{-1}$, яка встановлює відповідність між елементами множини V за формулою (3.5):

$$\begin{aligned} F_{1,k}^{-1}(\ v_{s,j,l+1}) &= \ v_{s,j,l}\ , \\ F_{1,k}^{-1}(\ v_{s\prime,j,l\prime}) &= \ v_{s,j,l}. \end{aligned} \tag{3.5}$$

Для КС з номером s у випадку появи j – того елементу множини V через час t від початку її функціонування в мережі можуть бути створені копії безпосередньо в цій же s – ій КС або інших КС локальної мережі. Тому, результат поширення одного елементу ЗПЗ можна відобразити послідовностями в табл. 3.2. Дане представлення залежить від часу та вузла ЛКМ, тому його можна інтерпретувати як часову модель поширення ЗПЗ в ЛКМ. Зобразимо на рис. 3.2 представлені в таблиці елементи, які відповідають копіям одного з елементів множини V, графом станів, де у вершинах розмістимо ці елементи, а дуги відповідатимуть за зв'язки між попередніми та наступними копіями елементів.

Таблиця 3.2

Елементи ЗПЗ в часовому представленні

Номер КС	Час, t									
	0	1	2	3	4	…	t_1	…	t_z	…
1				$v_{1,j,0}$ $(v_{s\prime,j,0\prime})$		…			$v_{1,j,1}$	…
…	…	…	…	…	…	…	…	…	…	…
s-2					$v_{s-2,j,0}$ $(v_{s\prime,j,0\prime})$	…	$v_{s-2,j,1}$			…
s-1				$v_{s-1,j,0}$ $(v_{s\prime,j,0\prime})$		…	$v_{s-1,j,1}$ $(v_{s+1\prime,j,0\prime})$			…
s		$v_{s,j,0}$			$v_{s,j,1}$	…	$v_{s,j,l}$	…	$v_{s,j,l+1}$	…
s+1			$v_{s+1,j,0}$ $(v_{s\prime,j,0\prime})$			…				…
s+2			$v_{s+2,j,0}$ $(v_{s\prime,j,0\prime})$			…			$v_{s+2,j,1}$	…
…	…	…	…	…	…	…	…	…	…	…
n						…	$v_{n,j,0}$ $(v_{s\prime,j,1\prime})$			…

Зображені дугами графа на рис. 3.2 переходи, які здійснюватимуться між елементами, між різними вузлами ЛКМ міститимуть проміжні рівні, що відповідають за формування, пересилання та обробку мережних пакетів. Також, в дугах є ще один рівень переходів, який відображає виконання операції поширення копій із використанням оперативного запам’ятовуючого пристрою. Часову діаграму для графу з рис. 3.2 зобразимо на рис. 3.3.

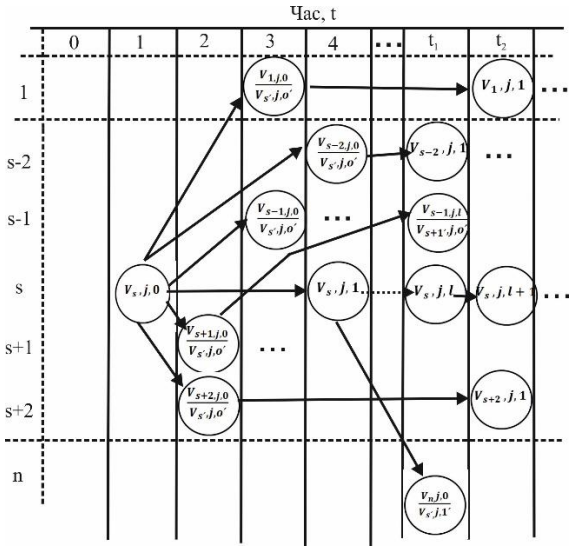


Рис.3.2. Граф поширення ЗПЗ в часі та в ЛКМ, представлений станами для одного з елементів множини V

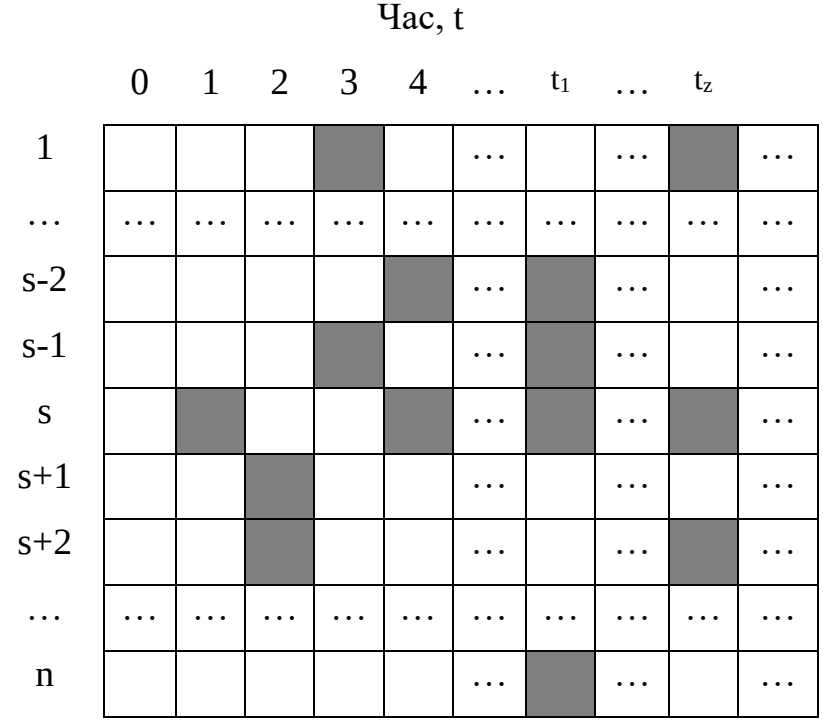


Рис.3.3. Часова інтерпретація поширення елементу $v_{s,j,l}$ з множини V

Зведення даних про поширення копій зі всіх КС локальної мережі, наприклад, зображено на рис. 3.4.

Час, t																					
t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}	t_{19}	t_{20}	t_{21}	t_{22}	…
																						…

Рис.3.4. Зведена часова інтерпретація поширення елементу $v_{s,j,l}$ з множини V

Наслідком з такої інтерпретації поширення елементу $v_{s,j,l}$ з множини V є можливість обчислення середнього часу поширення для одного елементу та багатьох за формулою (3.6):

$$\Delta t = \frac{t_z - t_0}{\sum_{i=1}^{r} l_i}, \tag{3.6}$$

де l_i – кількість копій елементу $v_{i,j,l}$ з множини V в кожному вузлі ЛКМ; i – кількість КС в мережі; t_0 – початковий час; t_z – поточний час роботи.

Також, цю формулу можна узагальнити для всіх елементів з множини V, які можуть поширюватись в ЛКМ, тоді середній час поширення для всіх елементів визначається за формулою (3.7):

$$\Delta t = \frac{t_z - t_0}{\sum_{j=1}^{q}\sum_{i=1}^{r} l_{i,j}}, \tag{3.7}$$

де $l_{i,j}$ – кількість копій j – того елементу $v_{i,j,l}$ з множини V в кожному вузлі ЛКМ; i – кількість КС в мережі; j- тий елемент з множини V; t_0 – початковий час; t_z – поточний час роботи.

Крім того, з формули (3.6) можна виразити швидкість поширення ЗПЗ в ЛКМ за певний час за формулою (3.7):

$$w = \frac{\sum_{j=1}^{q}\sum_{i=1}^{r} l_{i,j}}{t_z - t_0}. \tag{3.8}$$

Цю швидкість w можна використати для оцінки прогнозу поширення ЗПЗ на протязі деякого часу.

Важливим, також, є дослідження кількості поширення елементів з множини V в конкретних вузлах мережі в позиціонуванні їх від місця поширення та від певної копії одного елементу, тобто визначення кількості поширених копій у вузлах ЛКМ одного елементу $v_{i,j,l}$ з множини V. Матрицею суміжності представимо в табл. 3.3 залежність породжених копій елементів у вузлах ЛКМ від копій елементів з різних вузлів ЛКМ.

Матриця є несиметричною, бо граф поширення ЗПЗ в ЛКМ є орієнтованим. Очевидно, що копії елементів, які поширені з однієї КС можуть поширити свої копії теж на цю ж КС. Тому, за умови поширення ЗПЗ протягом тривалого часу можливою є наявність всіх ненулевих елементів матриці. Такі матриці для одного або більше елементів з множини V, отримані багатократно на протязі певного часу, дозволяють визначати рівень безпеки розподіленої багаторівневої системи виявлення ЗПЗ в ЛКМ.

Таблиця 3.3

Залежність породжених копій елементів у вузлах ЛКМ

Номер КС	1	2	…	n	Разом:
1	$l_{1,1}$	$l_{1,1}$	…	$l_{1,n}$	$\sum_{j=1}^{n} l_{1,j}$
2	$l_{1,1}$	$l_{2,2}$	…	$l_{2,n}$	$\sum_{j=1}^{n} l_{2,j}$
…	…	…	…	…	…
n	$l_{n,1}$	$l_{n,2}$	…	$l_{n,n}$	$\sum_{j=1}^{n} l_{n,j}$
Разом:	$\sum_{i=1}^{n} l_{i,1}$	$\sum_{i=1}^{n} l_{i,2}$	…	$\sum_{i=1}^{n} l_{i,n}$	$\sum_{j=1}^{n}\sum_{i=1}^{n} l_{i,j}$

Розглянемо ЗПЗ відносно місця розміщення без врахування часу, протягом якого воно поширювалось і буде поширюватись. Особливою ознакою виділимо кількість об’єктів, які можуть бути розміщені в пам’яті кожної КС мережі. Ця кількість є скінченою і залежить від обсягу пам’яті, причому в різних КС вона може бути різною. Для КС з номером s у випадку появи j – того елементу множини V через час t від початку її функціонування в мережі можуть бути створені копії безпосередньо в цій же s – ій КС або інших КС локальної мережі. Тому, результат поширення одного елементу ЗПЗ можна відобразити послідовностями в табл. 3.4, які на відміну від часової моделі можуть бути розміщеними не спочатку, а в певній частині пам’яті і поширюватись в об’єкти як до так і після об’єкту з $v_{s,j,l}$. Зображення у вигляді графу подано на рис.3.5.

Задамо узагальнене представлення даних з табл. 3.3 матрицею суміжності. Для цього введемо позначення об’єктів пам’яті змінними та їх представлення у вигляді лінійних многочленів з коефіцієнтами. Дійсно, кожному об’єкту, відомості про який представлено в таблицях файлових систем можна поставити у взаємно-однозначну відповідність змінну з індексом. Оскільки кількість таких об’єктів скінченна, тоді многочлен буде мати скінченну кількість доданків. Побудуємо коефіцієнти змінних многочлена в такий спосіб, щоб вони містили

інформацію про ЗПЗ, його походження та різні його інші атрибути.

Таблиця 3.4

Елементи ЗПЗ в пам’яті

Номер КС	Кількість об’єктів в пам’яті									
	k_1	k_2	k_3	k_4	k_5	…	k_m	…	$k_{z(s)-1}$	$k_{z(s)}$
1				$v_{1,j,0}$ $(v_{s\prime,j,0\prime})$		…			$v_{1,j,1}$	
…	…	…	…	…	…	…	…	…	…	
s-2					$v_{s-2,j,0}$ $(v_{s\prime,j,0\prime})$	…	$v_{s-2,j,1}$			
s-1				$v_{s-1,j,0}$ $(v_{s\prime,j,0\prime})$		…	$v_{s-1,j,1}$ $(v_{s+1\prime,j,0\prime})$			
s	$v_{s,j,1}$			$v_{s,j,0}$		…	$v_{s,j,l}$	…	$v_{s,j,l+1}$	
s+1			$v_{s+1,j,0}$ $(v_{s\prime,j,0\prime})$			…				
s+2		$v_{s+2,j,1}$				…	$v_{s+2,j,0}$ $(v_{s\prime,j,0\prime})$			
…	…	…	…	…	…	…	…	…	…	
n		$v_{n,j,0}$ $(v_{s\prime,j,1\prime})$				…				

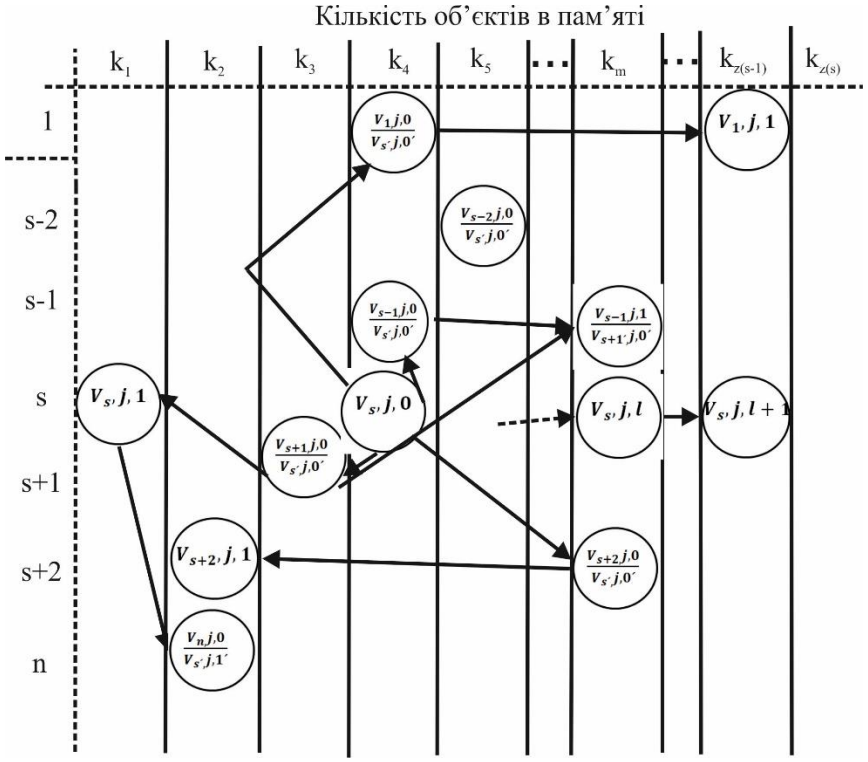


Рис.3.5. Граф поширення ЗПЗ у ЛКМ, представлений копіями для одного з елементів множини V

Виділимо з них такі та введемо їх позначення: (1) номер об’єкта з цього ж місця перебування; (2) номер об’єкта цієї ж КС але іншого місця перебування; (3) номер об’єкта з іншої КС, з якого надійшло ЗПЗ; (4) номер КС, з якої надійшов пакет з ЗПЗ. Для їх відображення коефіцієнтами використаємо базис i, j, k в такий спосіб: $\alpha_{1,s,p} + \alpha_{2,s,p}\, i + \alpha_{3,s,p}\, j + \alpha_{4,s,p}\, k$, де $\alpha_{x,s,p}$ - x-й коефіцієнт, що вибирається з множини $\{1; 2; 3; 4\}$, s – номер КС в мережі, p – номер об’єкту, для якого задано коефіцієнт. Крім того, для спрощення представлення таких коефіцієнтів, в яких зберігаються відомості про ЗПЗ, здійснимо їх подання в матричному вигляді за формулою (3.9):

$$\begin{pmatrix} \alpha_{1,s,p} & \alpha_{2,s,p}\, i \\ \alpha_{3,s,p}\, j & \alpha_{4,s,p}\, k \end{pmatrix} \quad \text{або} \quad \begin{pmatrix} \alpha_{1,s,p} & \alpha_{2,s,p} \\ \alpha_{3,s,p} & \alpha_{4,s,p} \end{pmatrix}. \tag{3.9}$$

Позначимо $A_{s,p} = \alpha_{1,s,p} + \alpha_{2,s,p}\, i + \alpha_{3,s,p}\, j + \alpha_{4,s,p}\, k$ і, тоді, в результаті отримуємо таке представлення об’єктів в мережі за формулою (3.10):

$$\begin{matrix} A_{1,1}x_{1,1} & +A_{1,2}x_{1,2} & +A_{1,3}x_{1,3} & \dots & +A_{1,p_1}x_{1,p_1} \\ A_{2,1}x_{2,1} & +A_{2,2}x_{2,2} & +A_{2,3}x_{2,3} & \dots & +A_{2,p_2}x_{2,p_2} \\ \dots & \dots & \dots & \dots & \dots \\ A_{n,1}x_{n,1} & +A_{n,2}x_{n,2} & +A_{n,3}x_{n,3} & \dots & +A_{n,p_n}x_{n,p_n} \end{matrix}, \tag{3.10}$$

де кожен рядок-вираз відображає стан об’єктів в КС, а всі рядки – стан об’єктів в локальній мережі, причому p_s – кількість об’єктів в КС, s – кількість КС в мережі. Для представлення матрицею відомостей про об’єкти необхідно доповнити всі послідовності виразів до $\max p_s$ доданками, в яких змінні $x_{s,p_{s+j}} = 0$, причому $p_s + j \le \max(p_s)$ для всіх $j = 0, 1, \dots,\ \max(p_s) - p_s$. Таким чином, отримаємо таку матрицю відомостей про об’єкти в мережі:

$$\begin{pmatrix} \begin{pmatrix} \alpha_{1,1,1} & \alpha_{2,1,1} \\ \alpha_{3,1,1} & \alpha_{4,1,1} \end{pmatrix} & \begin{pmatrix} \alpha_{1,1,1} & \alpha_{2,1,1} \\ \alpha_{3,1,1} & \alpha_{4,1,1} \end{pmatrix} & \begin{pmatrix} \alpha_{1,1,1} & \alpha_{2,1,1} \\ \alpha_{3,1,1} & \alpha_{4,1,1} \end{pmatrix} & \dots & \begin{pmatrix} \alpha_{1,1,1} & \alpha_{2,1,1} \\ \alpha_{3,1,1} & \alpha_{4,1,1} \end{pmatrix} \\ \begin{pmatrix} \alpha_{1,2,1} & \alpha_{2,2,1} \\ \alpha_{3,2,1} & \alpha_{4,2,1} \end{pmatrix} & \begin{pmatrix} \alpha_{1,2,1} & \alpha_{2,2,1} \\ \alpha_{3,2,1} & \alpha_{4,2,1} \end{pmatrix} & \begin{pmatrix} \alpha_{1,2,1} & \alpha_{2,2,1} \\ \alpha_{3,2,1} & \alpha_{4,2,1} \end{pmatrix} & \dots & \begin{pmatrix} \alpha_{1,2,1} & \alpha_{2,2,1} \\ \alpha_{3,2,1} & \alpha_{4,2,1} \end{pmatrix} \\ \dots & \dots & \dots & \dots & \dots \\ \begin{pmatrix} \alpha_{1,n\,1} & \alpha_{2,n,1} \\ \alpha_{3,n\,1} & \alpha_{4,n,1} \end{pmatrix} & \begin{pmatrix} \alpha_{1,n,1} & \alpha_{2,n,1} \\ \alpha_{3,n,1} & \alpha_{4,n,1} \end{pmatrix} & \begin{pmatrix} \alpha_{1,n,1} & \alpha_{2,n,1} \\ \alpha_{3,n,1} & \alpha_{4,n,1} \end{pmatrix} & \dots & \begin{pmatrix} \alpha_{1,n,1} & \alpha_{2,n,1} \\ \alpha_{3,n,1} & \alpha_{4,n,1} \end{pmatrix} \end{pmatrix}. \tag{3.11}$$

Враховуючи, що можливими місцями перебування ЗПЗ можуть бути основна та вторинна пам’ять і мережні пакети в кожній КС мережі, то задамо їх множиною $Q = \{0;\ 1;\ 2\}$, де елемент «0» відповідатиме за перебування в основній пам’яті, елемент «1» – вторинній пам’яті, елемент «2» – мережному пакеті. Знаходження копії елементу множини V в мережному пакеті може бути в момент часу, коли вже здійснено формування пакету та відбувається його пересилання і отримання, тому саме в цей момент часу копія не перебуватиме в основній чи вторинній пам’яті. Позначивши місце перебування M_q, де $q \in Q$, введемо відповідну функцію за формулою (3.12):

$$R_{1,k}\left(v_{s,j,l}\right) = \begin{cases} 0, \text{якщо } v_{s,j,l} \in M_0 \\ 1, \text{якщо } v_{s,j,l} \in M_1 \\ 2, \text{якщо } v_{s,j,l} \in M_2 \end{cases}, \tag{3.12}$$

де $v_{s,j,l}$ – елемент з множини V в кожному вузлі ЛКМ.

Далі використовуємо значення цієї функції $R_{1,k}\left(v_{s,j,l}\right)$ в матриці відповідності кожній КС мережі місця перебування елементу множини V і такі матриці будуємо для кожного j – го елементу.

Задамо алгебру для властивості, що характеризує пошук ЗПЗ місця в пам’яті для розміщення своєї копії в комп’ютерних системах. При цьому встановимо p=2 для однієї з КС мережі за формулою (3.13):

$$\mathfrak{B}_{V_{s,2}} = \langle V_s,\ \Omega_{F_2} \rangle, \tag{3.13}$$

де s – кількість вузлів ЛКМ; Ω_{F_2} – множина функцій заданих на множині V, яка здійснює пошук ЗПЗ місця в пам’яті для розміщення своєї копії в комп’ютерних системах.

Введемо предикати на множині V таким чином, що вони відображатимуть результат виконання відповідних функцій в множину $\{0; 1\}$, тобто наявність зв’язку між елементами $v_{s,j,l}$ та $v_{s,j,l+1}$, за формулою (3.14):

$$P_{1,k}(v_{s,j,l}, v_{s,j,l+1}) = \begin{cases} 1, \text{ якщо } F_{1,k}(v_{s,j,l}) = v_{s,j,l+1} \\ 0, \text{ якщо } F_{1,k}(v_{s,j,l}) \neq v_{s,j,l+1} \end{cases}, \quad (3.14)$$

де s - номер КС в мережі; l - номер копії ЗПЗ; $v_{s,j,l} \in V_s$.

Тоді, задамо модель за формулою (3.15):

$$\mathfrak{M}_{V,1} = \langle V; \quad \Omega_{P_{1,k}} \rangle, \quad (3.15)$$

де $\Omega_{P_{1,k}}$ - множина предикатів, заданих на множині V.

Розділимо всі функції, які задані на множині V і виконують дії по втіленню ЗПЗ у виконувані програми, на підмножини так: функції запису в початок виконуваної програми із збереженням її функціоналу, функції запису в середину виконуваної програми із збереженням її функціоналу, функції запису в кінець виконуваної програми із збереженням її функціоналу, функції запису в різні частини виконуваної програми із збереженням її функціоналу, функції запису в початок виконуваної програми без збереженням її функціоналу, функції запису в середину виконуваної програми без збереженням її функціоналу, функції запису в кінець виконуваної програми без збереженням її функціоналу, функції запису в різні частини виконуваної програми без збереженням її функціоналу. Задамо алгебру для цієї властивості, що характеризує механізм втілення ЗПЗ у виконувану програму при p=3 для однієї з КС мережі за формулою (3.16):

$$\mathfrak{B}_{V_{s,3}} = \langle V_s, \ \Omega_{F_3} \rangle, \quad (3.16)$$

де s – кількість вузлів ЛКМ; Ω_{F_3} – множина функцій заданих на множині V, яка здійснює втілення своїх копій ЗПЗ у виконувані програми.

Цю множину функцій Ω_{F_2} розділимо на вісім підмножин за способом втілення у корисну програму $\Omega_{F_3} = \bigcup_{r=1}^{8} \Omega_{F_{3,r}}$. Процес втілення у вибрану виконувану програму здійснюється шляхом виконання відповідної функції $\Omega_{F_{3,r}}$ $(r = 1,2,\ldots,8)$ з врахуванням структури виконуваної програми та типу

операційної системи. Відмінність між корисною виконуваною програмою і програмою, в яку втілено ЗПЗ, відобразиться в структурі і кожного разу при використанні типової функції буде однаковим. Тобто, результатом успішного виконання функції буде типова послідовність дій, результат якої відобразиться в результуючому коді виконуваної програми. Тип операційної системи, в якій можуть активуватись функції $\Omega_{F_3,r}$ ($r = 1,2,\ldots,8$) теж закладено їх розробниками і враховується при пошуку об'єктів для втілення. Введемо для відображення типу операційної системи множину $O = \{o_1, o_2, \ldots, o_g\}$, де g – кількість типів.

Додамо до цієї множини функцій Ω_{F_3} підмножини за способом втілення не в корисну програму, а формуванням окремого файлового об'єкту. Для таких функцій приймемо $r = 9$ і, тоді, відповідна підмножина буде $\Omega_{F_{3,9}}$.

Задамо алгебру для властивості, що характеризує механізми запису ЗПЗ в оперативну пам'ять при p=4 для однієї з КС мережі за формулою (3.17):

$$\mathfrak{B}_{V_{s,4}} = \langle V_s, \ \Omega_{F_4} \rangle, \tag{3.17}$$

де s – кількість вузлів ЛКМ; Ω_{F_4} – множина функцій заданих на множині V, яка здійснює запис ЗПЗ в оперативну пам'ять.

В множину функцій входять ті функції, які відносяться до того ЗПЗ, яке постійно перебуває в оперативній пам'яті, і позначимо його підмножиною $\Omega_{F_{4,1}}$. До другої підмножини віднесемо ті функції, які переводять об'єкти з вторинної пам'яті в оперативну пам'ять, і після цього вони поширюватимуться вже перебуваючи там. Позначимо цю підмножину $\Omega_{F_{4,2}}$. До третьої підмножини $\Omega_{F_{4,3}}$ віднесемо всі об'єкти з вторинної пам'яті, які при поширенні використовуватимуть оперативну пам'ять, і ці функції виконуватимуть дії по реалізації механізмів перенесення та виконання в ній команд ЗПЗ.

Задамо алгебру для властивості, що характеризує механізми приховування ЗПЗ свого перебування в комп'ютерних системах при p=5 для однієї з КС мережі за формулою (3.18):

$$\mathfrak{B}_{V_{s,5}} = \langle V_s,\ \Omega_{F_5} \rangle, \tag{3.18}$$

де s – кількість вузлів ЛКМ; Ω_{F_5} – множина функцій заданих на множині V, яка здійснює приховування ЗПЗ свого перебування в комп'ютерних системах.

Задамо алгебру для властивості, що характеризує механізми пошуку інших вузлів мережі для свого поширення при p=6 для однієї з КС мережі за формулою (3.19):

$$\mathfrak{B}_{V_{s,6}} = \langle V_s,\ \Omega_{F_6} \rangle, \tag{3.19}$$

де s – кількість вузлів ЛКМ; Ω_{F_6} – множина функцій заданих на множині V, яка здійснює пошуку інших вузлів мережі для свого поширення.

Задамо алгебру для властивості, що характеризує механізми для формування і відправки мережних пакетів в комп'ютерних системах при p=7 для однієї з КС мережі за формулою (3.20):

$$\mathfrak{B}_{V_{s,7}} = \langle V_s,\ \Omega_{F_7} \rangle, \tag{3.20}$$

де s – кількість вузлів ЛКМ; Ω_{F_7} – множина функцій заданих на множині V, яка здійснює для формування і відправки мережних пакетів в комп'ютерних системах.

Задамо алгебру для властивості, що характеризує механізми подолання систем захисту в комп'ютерних системах при p=8 для однієї з КС мережі за формулою (3.21):

$$\mathfrak{B}_{V_{s,8}} = \langle V_s,\ \Omega_{F_8} \rangle, \tag{3.21}$$

де s – кількість вузлів ЛКМ; Ω_{F_8} – множина функцій заданих на множині V, яка здійснює подолання систем захисту в комп'ютерних системах.

Задамо алгебру для властивості, що характеризує техніки запису своїх

копій в головний завантажувальний сектор в комп'ютерних системах при p=9 для однієї з КС мережі за формулою (3.22):

$$\mathfrak{B}_{V_{s,9}} = \langle V_s,\ \Omega_{F_9} \rangle, \tag{3.22}$$

де s – кількість вузлів ЛКМ; Ω_{F_8} – множина функцій заданих на множині V, які здійснюють запис своїх копій в головний завантажувальний сектор в комп'ютерних системах.

Задамо алгебру для властивості, що характеризує реалізацію виконання деструктивних дій в комп'ютерних системах при p=10 для однієї з КС мережі за формулою (3.23):

$$\mathfrak{B}_{V_{s,10}} = \langle V_s,\ \Omega_{F_{10}} \rangle, \tag{3.23}$$

де s – кількість вузлів ЛКМ; $\Omega_{F_{10}}$ – множина функцій заданих на множині V, які здійснюють виконання деструктивних дій в комп'ютерних системах.

Ці деструктивні дій відмінні від функцій реалізованих у властивостях 1-9.

Для заданих множин функцій алгебр, що відображають формальні властивості ЗПЗ в процесі його поширення, введемо множини предикатів $\Omega_{P_{s,k}}$ так, що вони відображатимуть результат успішного/неуспішного виконання відповідних функцій в множину $\{0; 1\}$, де s - номер КС в мережі. Тоді, задамо моделі, які відповідатимуть розглядуваним властивостям наступним чином:

$$\mathfrak{M}_{V,k} = \langle V;\quad \Omega_{P_{s,k}} \rangle, \tag{3.24}$$

де $\Omega_{P_{s,k}}$ - множина предикатів, заданих на множині V; k – номер властивості ЗПЗ.

Сукупність розроблених алгебр [387] є основою для системного розподілу інформації про характерні особливості ЗПЗ в процесі свого життєвого циклу. Використання таких характеристик дозволить здійснювати виявлення ЗПЗ шляхом аналізу особливостей, які проявлятимуться при виконанні функцій. Тобто, виконання кожної функції на множині ЗПЗ здійснюватиметься типовим

способом, знання про який використовуватиметься при виявленні.

Формалізовані властивості ЗПЗ представлені розробленими алгебрами задано моделями, які дозволили створити удосконалену модель ЗПЗ в локальних мережах на основі алгебраїчної системи, яка на відміну від класичної моделі Коена [64, 65], моделі Адлемана [1, 2] і моделі Бонфанте [36-38] деталізована до рівнів властивостей ЗПЗ, дозволяє представити ЗПЗ через механізми його поширення в плоскій моделі пам'яті, особливістю якої є розгляд паралельних середовищ поширення в пам'яті різних КС в локальній мережі. Це надасть змогу формалізовано представити ЗПЗ в локальних комп'ютерних мережах з метою його ідентифікації згідно характеристичних властивостей.

3.2. Алгебри поведінки зловмисного програмного забезпечення

Розробка алгебраїчних структур алгебри та алгебраїчних моделей надала можливість здійснити узагальнення та формалізацію предметної області для структурування ЗПЗ [388] за правилами його побудови на етапі створення. Ці алгебраїчні структури міститимуть наповнення конкретними знаннями функціонування ЗПЗ в процесі його життєвого циклу і будуть основою для розробки методів їх виявлення, а також за рахунок перенесення результатів отриманих в теорії алгебраїчних структур досягти оптимізації рішень та відповідно ефективності при розробці практичних методів, спрямованих на пошук конкретних підмножин зловмисного програмного забезпечення.

Для відображення різних особливостей ЗПЗ, які проявляються в його різних типах, представимо множину ЗПЗ V підмножинами $V_l \subseteq V$, де $l = \overline{1,w}$, $V = \bigcup_{l=1}^{w} V_l$, де w – кількість типових поділів зловмисного програмного забезпечення за певними ознаками чи критеріями. Все зловмисне програмне забезпечення, яке може функціонувати в локальних комп'ютерних мережах, можна реалізувати за допомогою базових операторів, які допускаються для виконання конкретного комп'ютерною системою, і їх множину позначимо через P_1, та базових предикатів, які відповідають елементарним заданим відношенням

між даними програми. Базовими операторами вважатимемо виклики бібліотечних функцій, базовий елемент програми або автономний фрагмент програми. Базові предикати приймають значення істинне або хибне.

Множину всіх підмножин множини P_2 позначимо через $\mathfrak{P}(P_2)$ і, тоді, елементи з цієї множини відповідають всеможливим значенням базових предикатів. Важливим при цьому представленні програми через базові оператори та предикати є те, що для створення програми одні і ті ж базові оператори можуть використовуватись багатократно, а не одноразово. Крім того, до зловмисного програмного забезпечення, також, віднесемо завершені фрагменти програм, які призначенні для виконання допоміжних дій при підготовці зловмисних дій. Базові оператори та предикати, які розглядатимемо при подальших випадках, вважатимемо співвіднесеними до мови програмування найбільш наближеної до апаратури комп'ютерної системи або обладнання комп'ютерної мережі.

Графічне представлення множини ЗПЗ, множини базових операторів та предикатів і співвідношення між ними, на прикладі однієї програми зображено на рис. 3.6.

Елементи мови програмування позначимо через p_j, j= $\overline{1,m}$, $p_j \in$ P, $P = P_1 \cup P_2$. Елементи множини ЗПЗ позначимо з врахуванням їх приналежності конкретній підмножині за формулою (3.25):

$$\begin{aligned} V_1 &= \{V_{11}, V_{12}, \dots V_{1n_1}\}; \\ V_2 &= \{V_{21}, V_{22}, \dots V_{2n_2}\}; \\ &\dots\dots\dots\dots\dots\dots\dots\dots \\ V_n &= \{V_{n1}, V_{n2}, \dots V_{nn_k}\}, \end{aligned} \tag{3.25}$$

де n_j– кількість елементів ЗПЗ j-ого класу; j=$\overline{1,n_k}$. На рис. 3.6 відображено відношення представлення програмних об'єктів на різних рівнях деталізації.

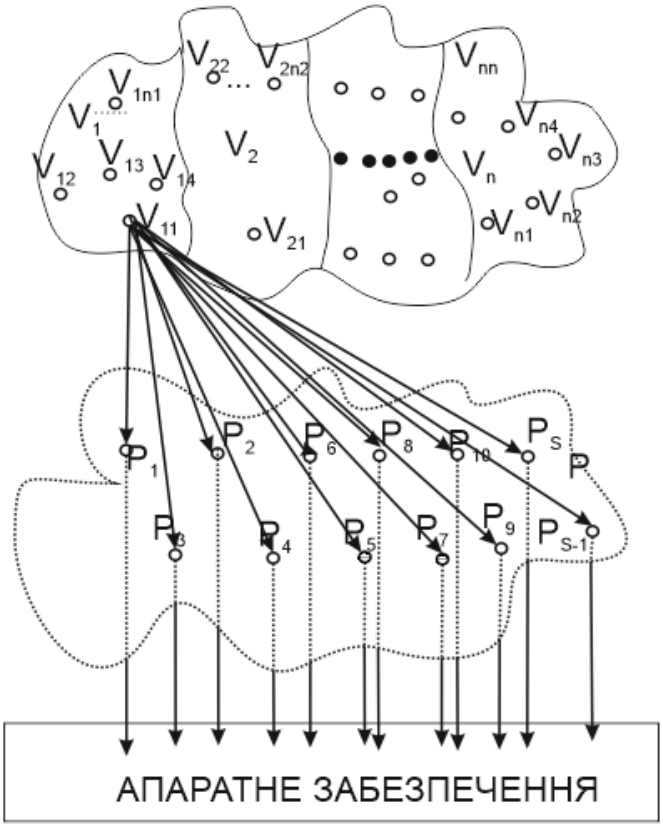


Рис. 3.6. Представлення програмних об’єктів на різних рівнях деталізації

Для конкретної зловмисної програми v_{ij} зображеної на рис. 3.6 зв’язками з елементами множини P іншого ЗПЗ та аналогічних елементів з V для мови кількість та послідовність використаних елементів з P можна представити в табл. 3.5.

Таблиця 3.5

Кількість та послідовність використаних елементів з P

V_{11}	α_1	α_2	α_2	……	α_s
	p_1	p_2	p_3	……	p_s

де α_k, k= $\overline{1,s}$– кількість входжень елементу p_k в структуру v_{ij}, якщо враховувати збереження послідовності, тоді представлення буде у вигляді строгої послідовності. Як в першому, так і в другому поданні елементу множини V можна виділити унікальну послідовність, як правило представлену байтами, за

якою можна ідентифікувати елемент ЗПЗ фактично встановлюючи відповідність шаблону, якщо таке ЗПЗ вже відоме і з нього отримано шаблон. Використання такого шаблону можливо на практиці, якщо він є незмінним при кожному тиражуванні (розмноженні) елементу ЗПЗ. Якщо ж при розповсюдженні ЗПЗ змінює свій код, тоді використання такого шаблону неефективне.

Кількість елементів p_i, i=$\overline{1,s}$ множини P скінченна і враховуючи особливості архітектури сучасних процесорів є не дуже великою. Але через можливість багатократного використання одних і тих же елементів p_j, тобто коефіцієнти α_j, j=$\overline{1,s}$ можуть бути великими, то представлення елементів $v_{ij} \in V$ може бути дуже різноманітним і як наслідок зводиться до комбінації з повтореннями: $P_{\alpha'}(\alpha_1,\alpha_2,\ldots\alpha_s)=\frac{\alpha'!}{\alpha_{1!},\alpha_{2!},\ldots\alpha_{s!}}$, де $\alpha'=\alpha_1+\alpha_2+\ldots+\alpha_s$. На оцінку можливого результату значення $P_{\alpha'}$ впливатимуть величини α_j, які можуть бути великими, але скінченними і як наслідок, буде справедливим співвідношення (формула (3.26)):

$$\left|\lim_{\substack{\alpha_{j\to q}\\ 1\le j\le s}} \frac{(\alpha_1+\alpha_2+\cdots+\alpha_s)!}{\alpha_1!\times\alpha_2!\times\ldots\times\alpha_s!}\right| \le \frac{(s\times q)!}{s\times q!}, \qquad (3.26)$$

де q = max$\{\alpha_j\}$, j=$\overline{1,s}$.

Як правило, елементи ЗПЗ прагнуть створювати з використанням мінімізованої кількості команд. Тому, величина q вважається невеликою. Але використання сучасних середовищ програмування та урізноманітнення засобів приховування зловмисного коду призводить до суттєвого збільшення величини q і як наслідок величини P_L. Це суттєво впливає на збільшення елементів унікального шаблону та відповідно на час пошуку через ускладнення обчислювального процесу.

Бази сигнатур зловмисних програм (шаблонів), які сформовані в сучасних антивірусних засобах поповнювалися більше 20 років і містять досить великі об'єми даних. Створення нових антивірусних засобів на основі сигнатурного

методу не дозволить конкурувати з тими, які вже накопичили свої бази сигнатур, а враховуючи зняття обмежень на величину q це стає неперспективним. Сигнатурний метод можна віднести до точних методів, якщо ставити задачу знаходження повного збігу з шаблоном.

В зв'язку з неможливістю поповнення бази сигнатур шаблонами відомих вірусів, розробка нових систем виявлення ЗПЗ на основі такого методу не перспективна, тому актуальним напрямком дослідження є розробка нових та вдосконалення існуючих моделей, методів та систем виявлення ЗПЗ, які дозволяють здійснювати пошук нового ЗПЗ не за їх сигнатурами, а використанням бази сигнатур було б допоміжним засобом для прискорення пошуку відомого ЗПЗ.

Формалізуємо область множини P та відношення між її величинами за допомогою алгебраїчних систем. Введемо операцію об'єднання (∪) на множині $\mathfrak{P}(P)$ і будемо розглядати її як об'єднання всеможливих підмножин множини P (формула (3.27)):

$$\mathfrak{S}_P = < \mathfrak{P}(P) ; \cup; \subseteq >. \qquad (3.27)$$

Дійсно, елементами множини $\mathfrak{P}(P)$ є всеможливі підмножини множини P, тоді їх об'єднання утворює множину, в якій елементи, що повторюються замінюються в цій множині один раз і, тоді, утворений перелік фактично буде однією з підмножин множини $\mathfrak{P}(P)$. В якості нулевового елементу виступає порожня множина.

Операція включення (⊆) відповідатиме за те, чи є серед затребуваних до виконання інструкцій всі елементи з P чи ні. Елемент ЗПЗ може включати порожню множину. Може виявитись, що серед елементів множини V є такі, що в їх структурі використані елементи, яких немає в P.

Алгебраїчна структура $\mathfrak{S}_P$ має тип <2, 2>, бо операції, задані на множині $\mathfrak{P}(P)$ є двохмісними. Операції ∪ та ⊆ є головними. Множиною функцій є $\sum_{F_T} = \{\cup\}$, а множиною предикатів - $\Omega_{\mathfrak{P}(P)_T} = \{\subseteq\}$. Виходячи з алгебраїчної

структури $\mathfrak{S}$ задано алгебру компонентів зловмисних програм $\mathfrak{B}_T = < \mathfrak{P}(P); \cup >$ та модель $\mathfrak{M}_T = < \mathfrak{P}(P); \subseteq >$.

Потужність множини $\mathfrak{P}(P)$ позначимо $|\mathfrak{P}(P)|$ і вона є для алгебраїчної системи $\mathfrak{B}_T$ її порядком. Потужність $\mathfrak{P}(P)$ є скінченною, бо множина P - скінчена. Це обмеження є необхідною умовою для доцільності побудови моделей пошуку ЗПЗ за критерієм ефективності. Дійсно, якщо б множина P була нескінченною і ця її нескінченність досягалась би за рахунок композиції елементів, тоді вона була б злічена, і як наслідок потужність розглядуваної множини $\mathfrak{P}(P)$ породженої множиною P мала потужність континууму.

Введення таких алгебраїчних структур дозволяє формалізувати простір, в якому перебувають розглядувані елементи з множини ЗПЗ, і є основою для дослідження різних типів ЗПЗ шляхом додавання нових операції над елементами множини $\mathfrak{P}(P)$. Представимо елементи $v_{ij} \in V_i$, $V_i \subseteq V$ для i=$\overline{1,n}$, j=$\overline{1,n_i}$ через елементи множини $\mathfrak{P}(P)$ наступним набором: $v_{ij} = (\{p_1\}; \{p_1, p_2\}; \dots)$.

Множина V формується з елементів, які обов'язково мають хоча б одну з ознак – властивостей, що відносяться до ЗПЗ. За цими властивостями, як за відношенням, множина V поділяється на класи – підмножини $V_{i,}$ i $= \overline{1,n}$. Всі елементи множини V входять до множини всіх програм, але за виділеними ознаками вони формують лише множину V, яка є підмножиною множини всіх програм. До складу множини V за виділеними ознаками можуть відноситись і не тільки елементи множини ЗПЗ, які мають згідно вимог до свого функціоналу функції, які є серед виділених ознак. Якщо S множина ознак – властивостей, тоді $\mathfrak{P}(S)$ - множина підмножин множини S і $V = \cup \mathfrak{P}(S)$, тому можна ввести алгебраїчну структуру за формулою (3.28):

$$\mathfrak{S}_V = < V; \cup; \subseteq >, \qquad (3.28)$$

де $\cup$ - операція об'єднання, яка задана на множині V; $\subseteq$ - операція включення, тобто якщо програма має одну з ознак – властивостей, тоді вона означає включення до множини V всіх програм, в яких є ознака з множини S.

Для достовірної класифікації програми з множини V необхідні методи, які б враховували ці ознаки з S. Є частина корисних програм (наприклад, архіватори), які мають подібні властивості і можуть бути помилково віднесеними в множину V.

Розглянемо множину V_1, в яку входять всі елементи ЗПЗ з ознаками-властивостями і задамо алгебраїчну структуру за формулою (3.29):

$$\mathfrak{S}_{\mathrm{V}_1} = <\ \mathrm{V}_1;\ F_1;\ P_1\ >. \tag{3.29}$$

Задамо функцію F_1: $V_1 \to P$, де предикат P, за формулою (3.30):

$$F_1(v_{1i}, p_j) = \begin{cases} 0, \text{не наявний } p_j \text{ в } v_{1i} \\ 1, p_j \text{ міститься в } v_{1i} \end{cases}, \tag{3.30}$$

Функція F_1 може бути задана декількома способами:

F_{11}: $\{P_1\}$ α_{11}

$\{P_2\}$ α_{21}

…… ……

$\{P_n\}$ α_{n1}

$\{P_1, P_2\}$ $\to$ $\alpha_{121}, \alpha_{221}$

$\{P_1, P_3\}$ $\alpha_{122}, \alpha_{322}$

…… …….

$\{P_1, P_{2,} P_3\}$ $\alpha_{131}, \alpha_{23}, \alpha_{33}$

……. ……..

Тобто $F_{11}(v_{1j}) = (\alpha_{11}; \alpha_{21}; \ldots \alpha_{n1}; \alpha_{121}; \alpha_{221}; \alpha_{122}; \alpha_{322}; \ldots \alpha_{131}; \alpha_{23}; \alpha_{33}; \ldots)$, що означає представлення послідовного входження кожного елементу з $\mathfrak{P}(P)$ у $v_{1j} \in V_1$, i=$\overline{1, n_1}$ числом. Для встановлення взаємно-однозначної відповідності елементи з P попередньо строго впорядковуються і, тоді, в подальшому кожен елемент з v_{1j} порівнюється з ними. В результаті за числами $\alpha_{i_1(,i_2\ ,i_3\ ,i_4\ ,\ldots,i_2 n_1\ )}$ можна відтворити v_{1j}. Взявши фрагмент або фрагменти такої

числової послідовності отримаємо сигнатуру, представлену в числовому вигляді.

Друга функція може не включати кількість входжень p_i , i $= \overline{1,s}$, а ставитиме у відповідність v_{1j} відповідні елементи j безпосередньо: F_1 (v_{1i}) = (pi_1, pi_2,…. pi_t), де t – кількість елементів у V_{1i}.

Третій варіант функції можна побудувати так:

$$F_{13} : v_{1i} \to p_1, \alpha_1$$
$$p_2, \alpha_2$$
$$\ldots\ldots\ldots$$
$$p_s, \alpha_s,$$

тоді $F_{13}(v_{1i})$ =((p_1; α_1),(p_2; α_2),…(p_s;α_s,)), тобто на основі визначення кількості входжень кожного з елементів p_j в $v_{1i} \in V$.

Функції F_{11}, F_{12} однозначно визначають елементи, з яких складаються $v_{1i} \in V_1$.

Функція F_{13} не завжди є взаємно – однозначною, особливо у випадках, коли кількість елементів p_i в $v_{1i} \in V_1$ є мінімальною, тоді входження p_i можуть бути однаковими для різних $v_{1i} \in V_1$, а також і для корисних програм.

Елементи $v_{1i} \in V_1$ можуть мати однакові властивості, але бути представлені різними наборами $p_j \in$ P. Тоді поділимо множини V_i на підмножини за ознакою чи групою ознак. Тобто на множинах V_i можна ввести відношення еквівалентності за ознаками – властивостями. Якщо розглядати множину V з введеним на ній відношенням еквівалентності, то класи еквівалентності формуватимуть множини V_i , i=$\overline{1,n}$, $\bigcap_{i=1}^{n} V_i = \emptyset$.

На практиці для знаходження V_i , i=$\overline{1,n}$ таких, що їх перетин є порожньою множиною, тобто однозначно віднести елементи v_{ij} до певної множини V_i не завжди є вирішуваною задачею і тому, відношення еквівалентності потрібно вводити за умови укрупнення ознак – властивостей.

Для здійснення класифікації введемо відношення еквівалентності на

кожній з множин $V_i, i = 1, n$ і представимо їх відповідними алгебрами за формулами (3.31) і (3.32):

$$\mathfrak{B}_S = < V,\ \cup, \leftrightarrow >, \tag{3.31}$$

$$\mathfrak{B}_{V_i} = < V_i,\ \cup, \leftrightarrow >, \tag{3.32}$$

де множини $V_i \subseteq V$ для всіх i=$\overline{1,n}$, $\bigcap_{i=1}^{n} V_i = \emptyset$.

Розроблені алгебраїчні системи та алгебри з введеними операціями на множині ЗПЗ є основою для створення поведінкових сигнатур ЗПЗ з метою їх формалізованого представлення в системах виявлення. Особливістю розроблених алгебраїчних систем є структуризація ЗПЗ [388] за типами, яка дозволяє здійснювати їх розподіл і віднесення до підмножин на основі характеристичних властивостей ЗПЗ для проведення ідентифікації та класифікації.

3.3. Типи загроз та їх представлення алгебрами поведінки

Розглянемо види загроз, які можуть бути здійснені в локальній мережі. Їх аналіз пов'язаний з вимогами, які висуваються до безпеки комп’ютерних систем в мережі: конфіденційність, цілісність, доступність та аутентичність. Для порушення цих вимог розробники ЗПЗ закладають в нього механізми здійснення загроз у вигляді таких атак: переривання, перехоплення, зміна, підробка. В локальних мережах здійснення таких атак або їх комбінацій відбувається по відношенню до апаратного забезпечення, програмного забезпечення, ліній зв’язку та даних. На рис. 3.7 зображено об’єкти в комп’ютерних системах локальних мереж, які можуть бути піддані атакам за певним типами загроз.

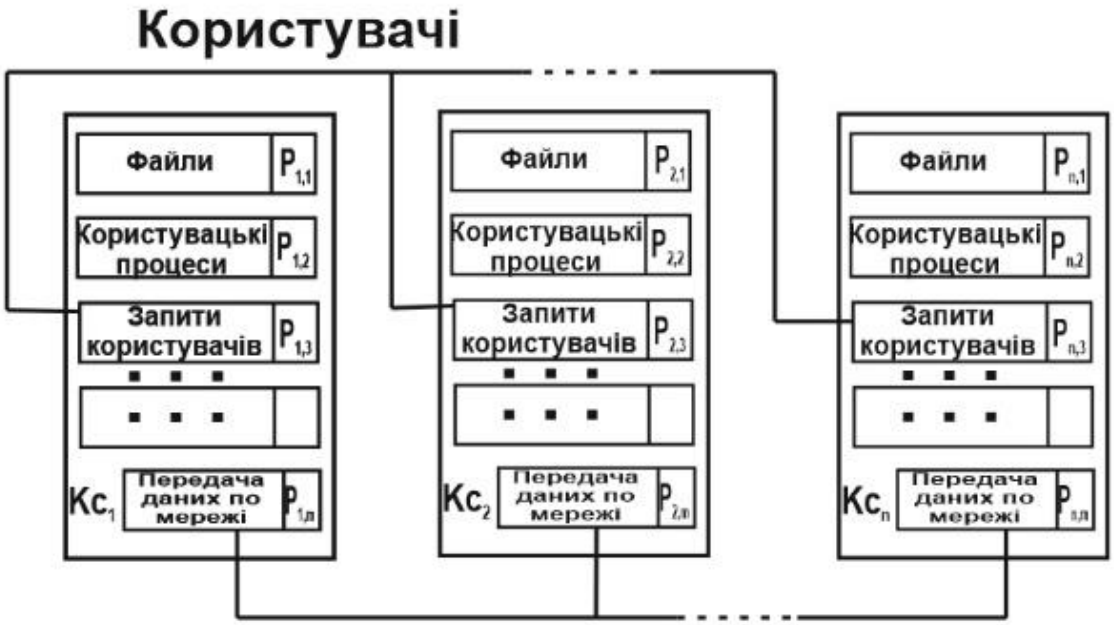


Рис.3.7. Об’єкти комп’ютерних систем, які можуть бути піддані атакам

Позначення:

$i = 1, n$;

$P_{i,1}$ – множина файлів i – ої комп’ютерної системи;

$P_{i,2}$ – множина користувацьких процесів i – ої комп’ютерної системи;

$P_{i,3}$ – множина запитів користувачів i – ої комп’ютерної системи;

$P_{i,4}$ – множина мережних пакетів i – ої комп’ютерної системи.

Функціонування комп’ютерних систем в локальних мережах пов’язане з обробкою, зберіганням та поширенням інформації. Саме при виконанні цих дій можливим є здійснення атак, узагальнені види яких виділимо наступним чином:

1) $Z_{i,1}$ – множина несанкціонованих змін;

2) $Z_{i,2}$ – множина підроблених об’єктів, розміщених в систему в результаті атаки;

3) $Z_{i,3}$ – множина перехоплень зі сторони зловмисника засобами програм або комп’ютерів;

4) $Z_{i,4}$ – множина переривань, яка здійснена для виведення з ладу компонентів системи.

Розглянемо детальніше можливі події при проведенні атак. Зокрема, елемент множини $P_{i,1}$ може бути скопійований чи перенесений в інше місце пам'яті. Тоді, можливі два випадки: ця подія відбулась успішно, ця подія не відбулась (або через помилки, пов'язані з роботою операційних систем чи компонентів комп'ютерної системи; або в результаті проведеної атаки). Задамо можливі події за формулою (3.33):

$$p_{f,1} \overset{f}{\Rightarrow} p_{f,1}$$
$$p_{f,1} \overset{z}{\Rightarrow} p_{f,1,z} \tag{3.33}$$

Задамо матрицями категорії атак [366, 373, 377]. Введемо такі вершини матриці: джерело інформації, отримувач інформації, подія переривання, подія перехоплення, подія зміни, підробка. Зв'язки між цими вершинами зобразимо напрямленими дугами. Зображення графу, що відповідає таким подіям, на рис.3.8.

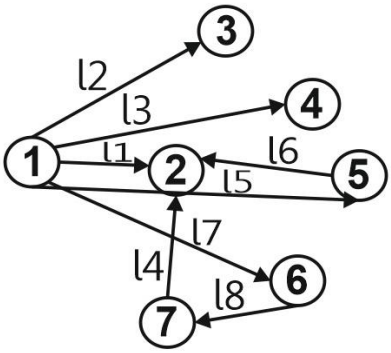


Рис. 3.8. Граф видів загроз

Позначення:

1-джерело інформації;

2-отримувач інформації;

3-подія переривання;

4-подія зміни інформації;

5-вершина, в якій відбувається перехоплення інформації;

6-отримання перехопленої інформації;

7-подія підробки інформації;

l_1 – передавання інформації здійснено вірно;

l_2 – відбулось переривання передавання інформації і вона не дійшла до отримувача;

l_3 – передавання інформації перервано і здійснюється її зміна;

l_4 – змінена інформація надсилається отримувачу;

l_5 – передавання інформації перехоплено, при цьому вона далі передається отримувачу без спотворень;

l_6 – передавання перехопленої інформації далі отримувачу;

l_7 – перехоплена інформація обробляється зловмисником;

l_8 - інформація від джерела не надсилалась, але зловмисник надсилає певну підроблену інформацію до отримувача.

Матриця інцидентності, що відповідає графу з рис. 3.8 зображена табл. 3.6.

Таблиця 3.6

Матриця інцидентності видів загроз

	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8
1	1	1	1	0	1	0	0	0
2	-1	0	0	-1	0	-1	0	-1
3	0	-1	0	0	0	0	0	0
4	0	0	-1	1	0	0	0	0
5	0	0	0	0	-1	1	1	0
6	0	0	0	0	0	0	-1	0
7	0	0	0	0	0	0	0	1

Види загроз залежать від особливостей комп’ютерних систем та їх компонентів. Виділимо компоненти, для яких встановимо їх взаємохв’язок з видами загроз: апаратне забезпечення, програмне забезпечення, данні, засоби організації зв’язку. Апаратне забезпечення через його доступність після збоїв викликаних втручанням може відмовляти в обслуговуванні. Програмне

забезпечення може мати такі види загроз: відмова користувачам в доступі, несанкціоноване копіювання, зміна функціоналу програми. Данні, які зберігаються, можуть бути видалені через видалення файлів, відмовити в доступі до них користувачам, несанкціоновано прочитані, змінено їх вміст. Засоби організації зв'язку можуть мати такі загрози, при здійсненні яких дозволять читання повідомлень, спостереження за трафіком, зміну вмісту, зміну часу доставки, порядку доставки повідомлень або їх дублювання, підробка повідомлень, видалення повідомлень.

Таким чином, здійснено виділення типових загроз у локальних мережах та запропоновано їх формалізоване представлення, використання якого є важливим при створення розподілених систем виявлення зловмисного програмного забезпечення.

Висновки до третього розділу

Розроблені алгебри поведінок ЗПЗ [387] є основою для системного розподілу інформації про характерні особливості ЗПЗ в процесі свого функціонування. Використання таких характеристик дозволить здійснювати виявлення ЗПЗ шляхом аналізу особливостей, які проявлятимуться при виконанні функцій. Тобто виконання кожної функції на множині ЗПЗ здійснюватиметься типовим способом, знання про який використовуватиметься при виявленні.

Властивості ЗПЗ представлені розробленими алгебрами і задано моделями, які дозволили створити удосконалену модель ЗПЗ в локальних мережах, яка на відміну від класичної моделі Коена, деталізована до рівнів властивостей ЗПЗ. Вона дозволяє представити ЗПЗ через механізми його поширення в плоскій моделі пам'яті, особливістю якої є розгляд паралельних середовищ поширення в пам'яті різних КС в локальній мережі. Це надасть змогу формалізовано представити ЗПЗ у локальних комп'ютерних мережах з метою його ідентифікації згідно характеристичних властивостей.

Розроблені алгебраїчні системи та алгебри з введеними операціями на множині ЗПЗ є основою для створення поведінкових сигнатур ЗПЗ з метою їх формалізованого представлення в системах виявлення. Особливістю розроблених алгебраїчних систем є структуризація ЗПЗ [388] за типами, яка дозволяє здійснювати їх розподіл і віднесення до підмножин на основі характеристичних властивостей ЗПЗ для проведення ідентифікації та класифікації.

Виділення типових загроз ЗПЗ в локальних мережах та їх формалізоване представлення дозволять створювати розподілені системи виявлення зловмисного програмного забезпечення.

Удосконалені моделі типів зловмисного програмного забезпечення поданням їх алгебрами поведінки є основою створення базису поведінкових сигнатур, і, на відміну від відомих представлень, враховують особливості їх функціонування в локальних мережах та дозволяють здійснити класифікацію за типами поведінки.

Основні результати розділу опубліковані у працях [387, 388, 377, 366, 373].

РОЗДІЛ 4
МЕТОДИ ВИЯВЛЕННЯ МЕРЕЖНОГО ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ

В якості мережного ЗПЗ розглядатимемо керовані бот-мережі та експлоїти, а також їх простіші варіації – worm-віруси. Функції, які закладені в їх алгоритми функціонування, виконують типові дії в комп'ютерних мережах. Основою цих дій є сканування портів, пошук вразливостей, розсилання запитів, встановлення в системі, поширення поштою, здійснення ретрансляції команд, переміщення спаму, пошук нових об'єктів для ураження і використання. Характерна особливість, яка чітко відрізняє цю підмножину ЗПЗ від інших, полягає в тому що переважно не відбуваються спроби копіювання зловмисного коду у всі файли, які розміщенні у вторинній пам'яті. Тому, необхідним є пошук та виокремлення характерних для цього типу ЗПЗ ознак, їх подальша формалізація та використання при виявленні. Враховуючи, що такий тип ЗПЗ функціонує та поширюється комп'ютерними мережами, то його пошук потрібно здійснювати мережними засобами [368, 369]. Оскільки ЗПЗ такого типу є складними програмними комплексами, які функціонують в глобальних комп'ютерних мережах, то для їх виявлення необхідним є використання аналогічних розподілених в комп'ютерних мережах засобів, зокрема і багатоагентних мережних систем [365, 367, 202]. З метою отримання переваги над таким ЗПЗ, його пошук можна локалізувати в локальних комп'ютерних мережах. Розроблена розподілена система виявлення ЗПЗ потребує наповнення знаннями про мережне ЗПЗ та методами [162 - 164, 200, 201, 203, 223, 350, 352, 381] його виявлення.

4.1. Типові компоненти еталонної моделі бот-мереж на основі функцій

Для проведення локалізації та ідентифікації мережного ЗПЗ здійснимо формалізацію закладених в них функцій. На основі множини функцій із розроблених алгебр типів ЗПЗ представимо їх за можливостями доступу до ресурсів комп'ютерних систем.

Розглянемо кероване мережне ЗПЗ бот-мережі. За своєю структурою в бот-мережах виділяють вузли, які відносяться до керування мережею і підтримки її цілісності, та вузли, які є кінцевими і з яких здійснюється виконання зловмисних дій. Керування бот-мережею здійснюється зловмисником через командно-контролюючий центр [390] безпосередньо або через інші проміжні віддалені контролюючі центри. Вся бот-мережа представляє собою розподілену програмну систему, в якій є рівень зв'язуючого програмного забезпечення. Виділимо дві основних складових бот-мережі: основний командно-керуючий центр, базові елементи мережі (боти). Крім того, для заплутування місцезнаходження основного командно-керуючого центру можуть використовуватись проміжні віддалені контролюючі центри, які можуть періодично змінювати свій стан з центрів на звичайні базові елементи бот-мережі і навпаки. Узагальнена структура бот-мережі зображена на рис. 4.1.

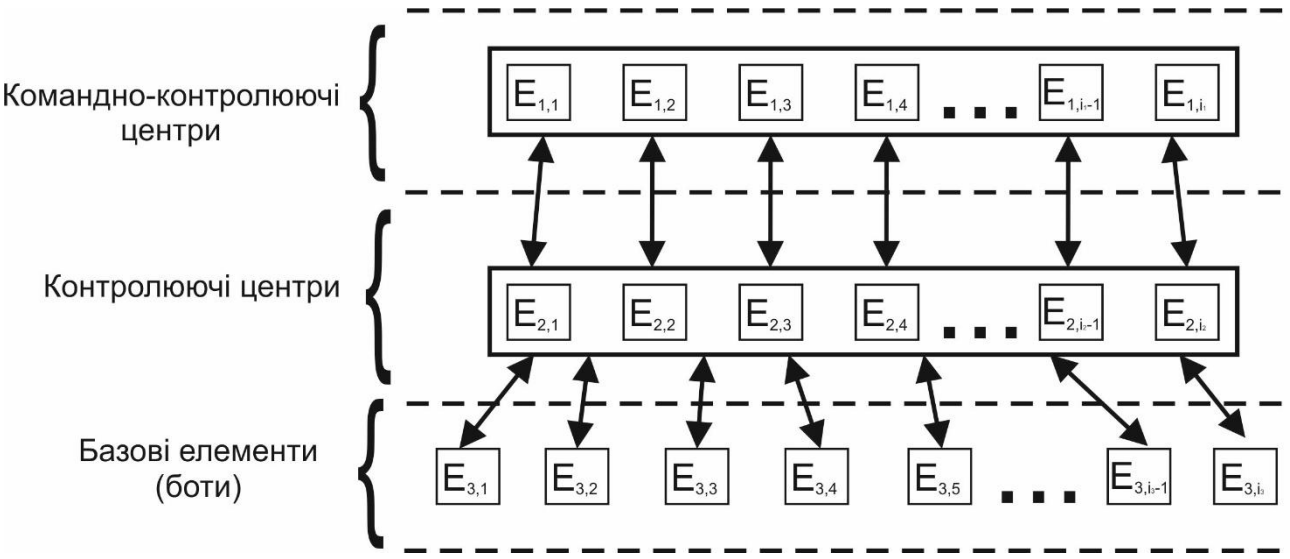


Рис. 4.1. Структура розподіленої керованої бот-мережі

Базові елементи бот-мережі позначимо підмножинами E_{3,i_3}, де $i_3 = 1, 2, \dots, n_3$, n_3 – кількість базових елементів мережі. Контролюючі центри бот-мережі позначимо підмножинами E_{2,i_2}, де $i_2 = 1, 2, \dots, n_2$, n_2 – кількість контролюючих центрів бот-мережі. При цьому рівень контролюючих центрів може бути і відсутнім, тобто базові елементи можуть комунікувати безпосередньо напряму з командно-керуючим центром. Але такий спосіб організації роботи бот-мережі при великій кількості базових елементів дуже неефективний. І крім того, така організація може призвести до швидкого виявлення командно-керуючого центру. Тому, наявність другого рівня, в якому перебуватимуть елементи бот-мережі відповідальні за підтримку цілісності сегментів, є обов'язковим. Функціонал елементів другого рівня може допускати можливість виконання команд і першого рівня, тобто виконання функцій елементів базового рівня. Елементи другого рівня можуть мати декілька ієрархічних рівнів. Командно-контролюючі центри бот-мережі займають третій рівень ієрархії і позначимо їх підмножинами E_{1,i_1}, де $i_1 = 1, 2, \dots, n_1$, n_1 – кількість командно-керуючих центрів бот-мережі. Кількість таких центрів може бути різною. Це необхідно для зміни доступу і адміністрування бот-мережею, щоб уникнути виявлення. Можливі також варіації, коли елементи трьох рівнів обмінюються місцями для заплутування і неможливості виявити опорні вузли. Але для командно-керуючого центру здійснити на практиці це важче, бо потрібен фізичний доступ до комп'ютерної системи. Крім того, зміна контролюючих центрів на базові елементи і навпаки займатиме багато часу на переналаштування сегменту бот-мережі, що може бути відслідковано. Тому, вважатимемо, що побудована бот-мережа може тільки розширюватись і при цьому змінювати конфігурування, а зміна її структури здійснюється виключно адміністратором за потреби, а не динамічно за заданими алгоритмами. Якщо б в функціонал базового елементу бот-мережі було закладено, також, можливість ставати і виконувати функції контролюючого центру, наприклад через певні інтервали часу, тоді б такий елемент бот-мережі володів би надлишковою

інформацією, що за певних умов його виявлення і дослідження, дозволило б виявити і базові елементи, приєднані до нього, і елементи з якими він комунікував. Тому, вважатимемо, що базовий елемент бот-мережі стає контролюючим центром тільки за умови приєднання до нього нових елементів в результаті успішно проведеної атаки. Таким чином, при достатньо великій кількості елементів бот-мережі базові елементи першого рівня за рахунок самостійного розширення можуть виконувати функції контролюючих центрів. Це важлива властивість, яка є основою для їх виявлення.

Бот-мережі можуть мати різну архітектуру [71] в залежності від топології і зв'язку елементів: мультисерверну, ієрархічну, випадкову (peer-to-peer), гібридну. Мультисерверна топологія передбачає наявність багатьох серверів, які двонаправлено зв'язані один з одним. Виведення з ладу одного з них визначається автоматично іншими елементами бот-мережі і в цьому випадку він вилучається з групи серверів. На рис. 4.1 відображена наявність елементів з підмножини E_{1,i_1} при $i_1 > 1$ дозволяє охарактеризувати бот-мережу як таку, що має мультисерверну топологію. Ієрархічна топологія утворюється (рис. 4.1) при наявності тільки одного елементу E_{1,i_1}, тобто при $i_1 = 1$, і при встановлені між кожними двома вузлами точно по одному зв'язку. Топологія «зірка» є частковим випадком ієрархічної топології при відсутності другого рівня і наявності елементів тільки першого і третього рівнів. Випадкова топологія на схемі рис. 4.1 може бути реалізована наявністю рівноцінного функційного навантаження в елементах всіх рівнів та встановленні зв'язку кожного з кожним, що вимагатиме фактичної наявності всього одного рівня бот-мережі. Гібридна топологія враховує всі перелічені топології. Таким чином, структура бот-мережі, яка зображена на рис. 4.1, покриває всі можливі відомі топології.

Представимо цілісну бот-мережу, як об'єднання її складових частин, за формулою (4.1):

$$E = \bigcup_{i_1=1}^{n_1} E_{1,i_1} \bigcup_{i_2=1}^{n_2} E_{2,i_2} \bigcup_{i_3=1}^{n_3} E_{3,i_3}, \qquad (4.1)$$

де E – множина складових частин бот-мережі.

Елементами підмножин $E_{w,i}$ є функції, з яких сформовано елемент $E_{w,i}$ бот-мережі. В різних елементах $E_{w,i}$ можуть бути однакові функції, тобто елементи $E_{w,i}$ можуть формуватись з однакових блоків (функцій). Тому, визначимо множину E_F, як таку що складається з різних функцій, які входять до множини E. Здійснимо представлення відповідних множин і підмножин через їх елементи функції за формулою (4.2):

$$\begin{aligned} E_{1,i} &= \{f_{1,i,1}, f_{1,i,2}, \dots, f_{1,i,q_i}\}, \\ E_{2,j} &= \{f_{2,j,1}, f_{2,j,2}, \dots, f_{2,j,p_j}\}, \\ E_{3,k} &= \{f_{3,k,1}, f_{3,k,2}, \dots, f_{3,k,r_k}\}, \end{aligned} \tag{4.2}$$

де q_i – кількість функцій в елементі бот-мережі $E_{1,i}$; p_j – кількість функцій в елементі бот-мережі $E_{2,j}$; r_k - кількість функцій в елементі бот-мережі $E_{3,k}$.

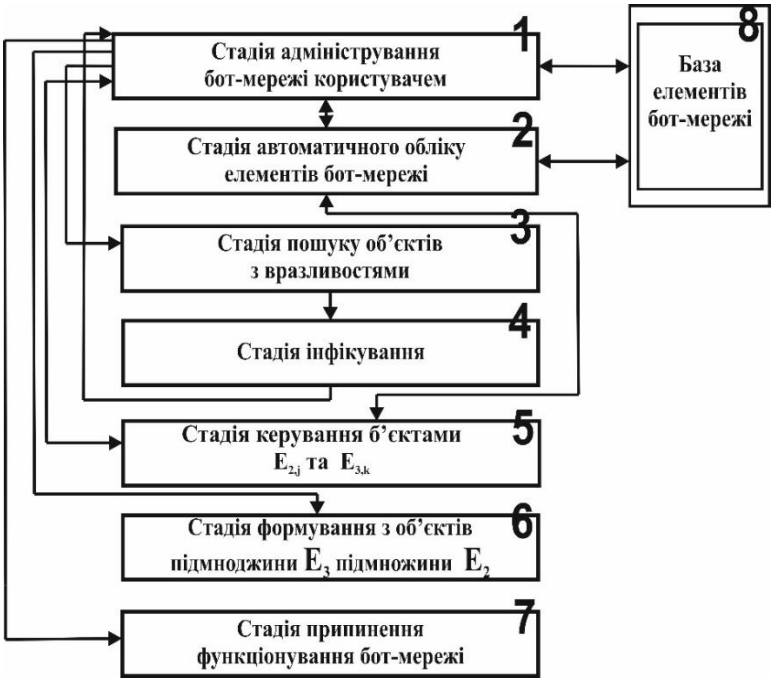
Тоді, множина E_F визначатиметься за формулою (4.3):

$$E_F = \{f_{F,1}, f_{F,2}, \dots, f_{F,p_F}\}, \tag{4.3}$$

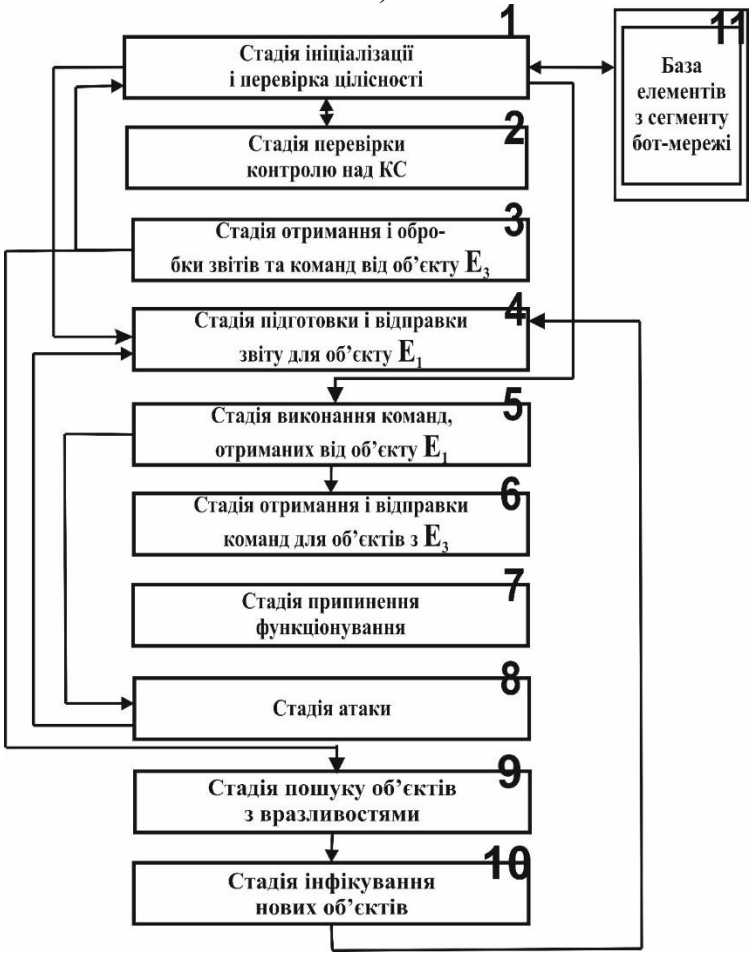
де p_F – загальна кількість різних функцій в бот-мережі; $f_{F,i}$ – i – та функція.

Множини функцій представлені в формулах 3.16 - 3.24 включають функції з множини E_F, тому $E_F \subseteq \Omega_F$. Для здійснення локалізації та ідентифікації бот-мереж в локальних комп'ютерних мережах розробимо представлення функцій множини E_F з врахуванням їх співвіднесення до різних підмножин множини Ω_F. Ці функції впливають на життєвий цикл бот-мережі та кожного з її елементів. Тому, здійснимо декомпозицію бот-мережі на такі складові, як основну зв'язуючу частину та її окремі елементи, і проаналізуємо життєвий цикл кожної із цих складових. Основна зв'язуюча частина містить командно-керуючий центр та засоби комунікації решти елементів мережі. Другою складовою бот-мережі в такому поділі є її елементи розглядувані окремо та в сукупності. Стадії

функціонування основних зв’язуючих частин бот-мережі зображено схемою на рис. 4.2.



а)



б)

в)

Рис. 4.2. Схеми функціонування трьох основних зв'язуючих частин бот-мережі

Вважатимемо, стадії функціонування цієї частини бот-мережі зображені на рис. 4.2 такими за умови, що в складі бот-мережі вже є елементи всіх трьох підмножин $E_{w,i},\ w = 1, 2, 3$. Тобто, функціонально бот-мережа має всі складові і може виконувати поставлені для неї зловмисником завдання. При наборі елементів тільки будь-яких двох підмножин, одна з яких обов'язково є E_{1,i_1}, життєвий цикл буде здійснюватись або в межах локальної мережі або частково в ній і частково за її межами. При наявності елементів тільки підмножини E_{1,i_1}, що відповідає фактично початку роботи по створенню бот-мережі, розгляд її життєвого циклу є актуальним за умови, що ці елементи перебувають в локальній комп'ютерній мережі, в якій може бути встановлена розподілена багаторівнева система.

Виконання закладених функцій в бот-мережах впливає на стадії їх життєвого циклу [202, 227, 390]. Встановимо залежність між ними та стадіями життєвого циклу бот-мереж.

Стадія адміністрування бот-мережі користувачем відбувається на всіх її рівнях, але основними елементами бот-мережі для її здійснення є елементи з множини E_{1,i_1}, які згруповані в елементі $e_{1,i,r} \in E_{1,i_1}$, r – кількість елементів в множині $E_{1,i}$. Кількість підмножин E_{1,i_1} визначається зловмисником, тобто може бути довільною. Це означає, що для уникнення свого виявлення зловмисник може здійснювати адміністрування бот-мережі з різних вузлів. Розглянемо можливе наповнення елементу $e_{1,i,r} \in E_{1,i_1}$. Цим наповненням буде визначатись множина бот-мереж для подальшого дослідження. В якості функцій приймемо такі, що виконуватимуть наступні дії:

$f_{1,i,1}$ – основна функція адміністрування бот-мережі (вибір меню адміністратора);

$f_{1,i,2}$ – функція обробки запитів, які надійшли від компонентів бот-мережі, з метою їх опрацювання адміністратором;

$f_{1,i,3}$ – функція доповнення бази даних новими елементами бот-мережі;

$f_{1,i,4}$ – функція корегування полів бази елементів на основі отриманої інформації від елементів бот-мережі;

$f_{1,i,5}$ – функція відправки команд на інші вузли бот-мережі;

$f_{1,i,6}$ – функція обробки відповідей на відправлені команди з інших вузлів бот-мережі;

$f_{1,i,7}$ – функція пошуку уразливих елементів в програмному забезпеченні комп'ютерних систем в мережі для створення нових елементів бот-мережі;

$f_{1,i,8}$ – функція формування та введення нових команд і задач для організації атак з елементів бот-мережі;

$f_{1,i,9}$ – функція проведення атаки на комп'ютерну систему для її ураження;

$f_{1,i,10}$ – функція створення нового вузла бот-мережі;

$f_{1,i,11}$ – функція перетворення елементів з множини E_{3,i_3} в елементи множини $E_{2,k}$;

$f_{1,i,12}$ – функція організації припинення функціонування бот-мережі.

Стадія автоматичного обліку елементів бот-мережі необхідна для

спрощення роботи адміністратора, враховуючи велику кількість елементів бот-мережі. Основними її функціями є такі:

$f_{1,i,13}$ – функція обробки повідомлень від елементів з множини E_{3,i_3} та елементів множини E_{2,i_2} бот-мережі;

$f_{1,i,14}$ – функція передачі адміністратору результатів автоматичного обліку бот-мережі;

$f_{1,i,15}$ – функція взаємодії з базою елементів бот-мережі;

$f_{1,i,16}$ – функція отримання і обробки команд від адміністратора бот-мережі.

Стадія пошуку об’єктів з вразливостями передбачає наявність інструментарію адміністратора для можливості на первинному етапі формування бот-мережі мати функції пошуку вразливостей. До основних функцій цієї стадії життєвого циклу командно-контролюючого центру бот-мережі віднесемо такі:

$f_{1,i,17}$ – функція з набором інструментів для пошуку вразливостей в об’єктах в мережі;

$f_{1,i,18}$ – функція отримання і обробки команд від адміністратора бот-мережі;

$f_{1,i,19}$ – функція відправки результатів пошуку об’єктів з вразливостями адміністратору бот-мережі;

$f_{1,i,20}$ – функція запуску стадії проведення інфікування об’єкту з вразливостями бот-мережі.

Стадія інфікування включає розміщення в уразливому об’єкті функцій для формування елементу множини E_{3,i_3} і може бути реалізована такими функціями:

$f_{1,i,21}$ – функція вибору місця записування зловмисного програмного забезпечення в уразливий об’єкт;

$f_{1,i,22}$ – функція надсилання зловмисного програмного забезпечення або його компонентів в уразливий об’єкт бот-мережі та його розміщення у попередньо вибраному місці;

$f_{1,i,23}$ – функція здійснення запуску та налаштування зловмисного

програмного забезпечення в уразливому об’єкті;

$f_{1,i,24}$ – функція отримання підтвердження та обробки першого повідомлення від зловмисного програмного забезпечення в новому об’єкті;

$f_{1,i,25}$ – функція передачі повідомлення адміністратору бот-мережі про створення нового об’єкту та інформації про нього.

Стадія керування об’єктами з підмножин E_{2,i_2} та E_{3,i_3} включає функції, які здійснюють обробку команд між об’єктами цих підмножин, а також адміністратором, і можуть бути такими:

$f_{1,i,26}$ – функція отримання і обробки команд від адміністратора;

$f_{1,i,27}$ – функція передачі повідомлення від об’єктів з підмножини E_{2,i_2} до об’єктів з підмножини E_{3,i_3};

$f_{1,i,28}$ – функція передачі повідомлення від об’єктів з підмножини E_{3,i_3} до об’єктів з підмножини E_{2,i_2};

$f_{1,i,29}$ – функція виконання команд адміністратора об’єктами з підмножини E_{2,i_2} та $E_{3,i}$ бот-мережі.

При наявності великої кількості об’єктів підмножини E_{3,i_3} і, як наслідок, ускладнення їх обліку та адміністрування адміністратор здійснює створення нового сегменту бот-мережі. При цьому виконуються такі функції:

$f_{1,i,30}$ – функція доповнення вибраного адміністратором елементу з підмножини E_{3,i_3} додатковими функціями для перетворення його на об’єкт з підмножини E_{2,i_2};

$f_{1,i,31}$ – функція передачі команд від адміністратора до вибраних ним елементів з підмножини E_{3,i_3} для організації їх зв’язку з вибраним сформованим об’єктом з підмножини E_{2,i_2};

$f_{1,i,32}$ – функція обробки повідомлень від елементів з підмножини E_{3,i_3} до віднесеного до них об’єкту з підмножини E_{2,i_2};

$f_{1,i,33}$ – функція надсилання повідомлень від об’єкту з підмножини E_{2,i_2} до віднесених до нього елементів з підмножини E_{3,i_3}.

В процесі завершення функціонування бот-мережі адміністратор переводить її до стадії припинення функціонування за допомогою таких функцій:

$f_{1,i,34}$ – функція знищення об’єкту з підмножини E_{3,i_3} бот-мережі;

$f_{1,i,35}$ – функція знищення об’єкту з підмножини E_{2,i_2} бот-мережі;

$f_{1,i,36}$ – функція знищення бази даних елементів бот-мережі;

$f_{1,i,37}$ – функція знищення командно-контролюючого центру бот-мережі.

База елементів бот-мережі може бути сформована як окремий компонент бот-мережі і перебувати в тій же КС, що і командно-керуючий центр, або бути його частиною.

Розглянемо функції, які формують стадії життєвого циклу елементів підмножини E_{2,i_2} бот-мережі. Основні функції етапів життєвого циклу бот-мережі для контролюючих центрів згруповані в елементах $e_{2,i,p} \in E_{2,i_2}$, де p – кількість елементів в множині E_{2,i_2}. Кількість підмножин E_{2,i_2} визначається зловмисником, тобто може бути довільною. Це означає, що для розбудови бот-мережі зловмисник може здійснювати створювати необхідну йому кількість контролюючих центрів з різним наповненням і структурою. Розглянемо можливе наповнення елементу $e_{2,i,p} \in E_{2,i_2}$.

Стадія ініціалізації і перевірки цілісності сегменту бот-мережі може при виконанні таких функцій:

$f_{2,i,38}$ – функція обробки повідомлень та команд отриманих від командно-контролюючого центру;

$f_{2,i,39}$ – функція передачі команд до віднесених елементів з підмножини E_{3,i_3} для організації їх зв’язку з об’єктом з підмножини E_{2,i_2};

$f_{2,i,40}$ – функція обробки повідомлень від елементів з підмножини E_{3,i_3} до віднесеного до них об’єкту з підмножини E_{2,i_2}.

Стадія перевірки контролю над КС функції для перевірки дійсної належності елементу бот-мережі відповідному її сегменту і представлена так:

$f_{2,i,41}$ – функція здійснення самоперевірки цілісності елемента підмножини E_{2,i_2}.

Стадія отримання і обробки звітів та команд від об'єктів підмножини E_{3,i_3} представлена функціями:

$f_{2,i,42}$ – функція отримання і обробки повідомлень від об'єктів підмножини E_{3,i_3};

$f_{2,i,43}$ – функція обробки команд від об'єктів підмножини E_{3,i_3}.

Стадія підготовки і відправки звіту для об'єкту з E_{1,i_1} визначається такими функціями:

$f_{2,i,44}$ – функція формування звіту у вигляді пакету;

$f_{2,i,45}$ – функція відправки звіту до командно-керуючого центру, яка враховує встановлений для цього час або подію (отримання відповідної команди).

Стадія виконання команд, отриманих від об'єкту з E_{1,i_1} визначається такими функціями:

$f_{2,i,46}$ – функція отримання та ідентифікації команд від об'єкту з E_{1,i_1};

$f_{2,i,47}$ – функція вибору відповідної команди/команд і їх виконання;

$f_{2,i,48}$ – функція вибору відповідної команди/команд для об'єктів з E_{3,i_3}, їх підготовка та відправка на вказані елементи бот-мережі;

$f_{2,i,49}$ – функція припинення функціонування даного елементу;

$f_{2,i,50}$ – функція виконання відповідної команди/команд проведення атаки на вказані об'єкти.

Стадія підготовки і відправки команд для об'єктів підмножини E_{3,i_3} визначається такими функціями:

$f_{2,i,51}$ – функція підготовки для відправки команд для елементів з E_{3,i_3};

$f_{2,i,52}$ – функція відправки команд для елементів з E_{3,i_3}, яка враховує їх активність на даний момент часу.

Стадія пошуку об'єктів з вразливостями формується за рахунок вибору

функцій, які враховують набори вразливостей в комп'ютерних системах мережі і можуть бути такими:

$f_{2,i,53}$ – функція пошуку комп'ютерних систем, які не є частиною цієї бот-мережі, на основі сканування;

$f_{2,i,54}$ – функція здійснення аналізу програмних і технічних засобів комп'ютерних систем;

$f_{2,i,55}$ – функція обробки даних проведеного аналізу.

Стадія інфікування нових об'єктів здійснюється після знаходження комп'ютерної системи, аналіз засобів якої показав, що її можливо інфікувати, та може здійснюватись такими функціями:

$f_{2,i,56}$ – функція вибору інструментів для інфікування на основі проведеного аналізу комп'ютерної системи (в т.ч. може пропонувати користувачу переглянути листи та сайти);

$f_{2,i,57}$ – функція здійснення інфікування комп'ютерних систем;

$f_{2,i,58}$ – функція закріплення зловмисного програмного забезпечення в комп'ютерній системі;

$f_{2,i,57}$ – функція здійснення інфікування комп'ютерної системи.

Стадія атаки формується набором команд, які отримуються з командно-керуючого центру і є функції для виконання цих команд. Стадію атаки можуть формувати такі функції:

$f_{2,i,58}$ – функція здійснення обробки команд від командно-керуючого центру для проведення атаки;

$f_{2,i,59}$ – функція формування команд для виконання стосовно вибраного об'єкту для атаки;

$f_{2,i,60}$ – функція здійснення атаки комп'ютерної системи.

Основні види атак [165], які можуть проводитись із застосуванням бот-мереж:

1) атаки спрямовані на виснаження ресурсів комп'ютерної мережі;

2) атаки спрямовані на виснаження ресурсів вузла мережі, що може

проявлятись в захопленні пам’яті, центрального процесора і інших компонентів комп’ютера;

3) атаки, які викликають збій в роботі вузла через помилки в роботі його програмного забезпечення;

4) атаки, які викликають зміну в конфігуруванні або стані системи, що призводить до неможливості передавання даних, скиданні з’єднання або суттєвому зниженню ефективності.

Ці перелічені атаки (DoS – атаки) відносяться до атак спрямованих на зниження продуктивності або блокування доступу до мережі чи конкретного комп’ютера і його ресурсів та мають на меті реалізацію загрози, суть якої полягає у відмові в обслуговуванні. Використовуючи бот-мережу зловмисник здійснює віддалені DoS – атаки, які, як правило, супроводжуються IP – spoofing тобто підробкою зворотньої адреси у відправлюваних пакетах. Це необхідно для приховування місця (вузла), з якого ведеться атака. Тому, функції, що відповідають за проведення атаки, містять механізми реалізації підробки зворотньої адреси.

DDoS – атаки є розподіленими DoS – атаками (Distributed Denial of Service). Вони здійснюються не з одного вузла, а з декількох одночасно. Тому, бот-мережі є важливим засобом для проведення таких атак. Враховуючи необхідність наявності властивості розподіленості, утиліти для реалізації таких атак містять дві компоненти: клієнтську і серверну. Серверна частина містить функції, які можуть виконуватись після переданих команд від клієнтської частини. При цьому всі утиліти мають такі команди: почати атаку, завершити атаку, вибрати атаку. Вибір типу атаки здійснюється в функції $f_{2,i,58}$ і може бути одним з типів DoS – атак.

Для встановлення серверної частини в різні комп’ютерні системи можуть бути використані троянські програми або здійснено інфікування комп’ютерних систем з віднесенням їх до елементів підмножин E_{2,i_2} та E_{3,i_3}. Зв’язок клієнтської частини, тобто елементів підмножини E_{1,i_1}, з елементами підмножин E_{2,i_2} та E_{3,i_3}

реалізовується різними моделями. Перша модель зв'язку базується на тому, що встановлений компонент системи виконує функцію $f_{2,i,56}$ шляхом відправки електронного листа з інформацією про IP-адресу на відкритий порт в комп'ютерній системі. Зловмисник заносить всі отримані данні про елементи підмножин E_{2,i_2} та E_{3,i_3} в базу і за потреби віддає команди для виконання. Ця модель зв'язку відома і може мати недолік, який полягає в тому що відкриваються нестандартні порти і вони можуть бути виявлені брандмауерами або маршрутизаторами. Хоча застосовуватись може теж, бо передбачити, що зловмисник буде комбінувати різні варіанти для відволікання уваги, є можливим. Також, недоліком цієї моделі є необхідність встановлення дуже великої кількості з'єднань з вузлами бот-мережі, що є неефективним.

Друга модель зв'язку між елементами підмножин E_{1,i_1}, E_{2,i_2} та E_{3,i_3} реалізовується на основі використання IRC – мереж. Елементи підмножин E_{2,i_2} та E_{3,i_3} заходять на певний канал, самостійно підключаються до бот-мережі і очікують команд. При цьому варіанті зв'язку IRC – оператор може відімкнути канал зловмисника при виявленні зловмисних активностей.

Третя модель зв'язку між елементами підмножин E_{1,i_1}, E_{2,i_2} та E_{3,i_3} базується на тому, що клієнтська частина розподілена таким чином, що інформація про елементи підмножин E_{2,i_2} та E_{3,i_3} надходить не безпосередньо до елементів підмножини E_{1,i_1}, а на який-небудь сервер в мережі Інтернет у вигляді файлу певного типу і вже звідти елементи підмножини E_{1,i_1} отримують данні про бот-мережу та передають команди туди. Крім того, елементи підмножин E_{2,i_2} та E_{3,i_3} теж можуть саме так обмінюватись інформацією. В цьому варіанті серверна частина стає фактично реверсивною троянською програмою чи компонентою, а клієнтська частина стає звичайним текстовим файлом і засобом доступу до нього.

В процесі завершення функціонування бот-мережі адміністратор переводить її до стадії припинення функціонування за допомогою такої функції:

$f_{2,i,61}$ – функція знищення об'єкту з підмножини $E_{2,i}$ бот-мережі.

Кількість функцій, які формують стадії життєвого циклу елементів підмножини $E_{3,i}$ бот-мережі, є меншою в порівнянні з кількістю функцій 1-го та 2-го рівнів. Розглянемо ці функції у взамозв'язку зі стадіями життєвого циклу елементів з підмножини $E_{3,i}$ бот-мережі. Елементи згруповані в $e_{1,i,q} \in E_{3,i}$, q – кількість елементів в множині $E_{3,i}$. Кількість підмножин множини $E_{3,i}$ визначається зловмисником та залежить від того скільки інфікованих комп'ютерних систем складають бот-мережу. Розглянемо можливе наповнення елементу $e_{3,i,q} \in E_{3,i}$.

Стадія ініціалізації кінцевого елементу бот-мережі може відбуватись при виконанні таких функцій:

$f_{3,i,62}$ – основна функція, після запуску якої відбувається запуск всіх додаткових функцій для забезпечення контролю над комп'ютерною системою;

$f_{3,i,63}$ – функція запуску додаткових функцій.

Стадія перевірки контролю над КС формується функцією для перевірки дійсної належності елементу бот-мережі відповідному її сегменту і може бути представлена так:

$f_{3,i,64}$ – функція здійснення самоперевірки цілісності елемента підмножини $E_{3,i}$ на основі запуску команд перевірки контролю над КС.

Стадія підготовки і відправки звіту для об'єкту підмножини $E_{2,i}$ представлена функціями:

$f_{3,i,65}$ – функція підготовки повідомлень для об'єкту з підмножини $E_{2,i}$;

$f_{3,i,66}$ – функція надсилання звіту для об'єкту з підмножини $E_{2,i}$.

Стадія виконання команд, отриманих від об'єкту підмножини $E_{2,i}$ визначається такими функціями:

$f_{3,i,67}$ – функція отримання та ідентифікації команд від об'єкту з $E_{2,i}$;

$f_{3,i,68}$ – функція вибору відповідної команди/команд і їх виконання;

$f_{3,i,69}$ – функція припинення функціонування даного елементу;

$f_{3,i,70}$ – функція виконання відповідної команди/команд проведення атаки на вказані об'єкти.

Стадія пошуку об’єктів з вразливостями формується за рахунок вибору функцій, які враховують набори вразливостей в комп’ютерних системах мережі і можуть бути такими:

$f_{3,i,71}$ – функція пошуку комп’ютерних систем, які не є частиною цієї бот-мережі, на основі сканування;

$f_{3,i,72}$ – функція здійснення аналізу програмних і технічних засобів комп’ютерних систем;

$f_{3,i,73}$ – функція обробки даних проведеного аналізу.

Функція $f_{3,i,71}$ містить засоби сканування портів, які є відкритими і перебувають в стані очікування запитів. Активні служби, які перебувають в стані очікування, можуть надати зловмиснику можливість отримати несанкціонований доступ. Це стає можливим коли в програмному забезпеченні є помилки або коли система безпеки невірно налаштована. Утиліта nmap, що міститься в функції $f_{3,i,71}$ підтримує багато різних методів сканування. Основною особливістю, яка є визначальною в цій функції і може бути використана та використовується для ідентифікації бот-мереж, є те що обов’язково виконуються дії по надсиланню певного пробного пакету портом-сканером на заданий номер порту вузла мережі, очікуванням відповіді від нього і отриманням відповіді, яка дозволяє визначати стан порту (відкритий/закритий). В результаті виконання цієї функції порт - сканер досліджує задані номери портів та за певних умов діапазон вузлів мережі. Виділимо цю характерну функцію в функції $f_{3,i,71}$ таким чином: $f_{3,i,71} = \{nmap, \dots\}$. Розглянемо методи сканування, які використовуються, і відобразимо їх відповідними функціоналами в функції $f_{3,i,71}$. Набір таких методів об’єднують в одній функції $f_{3,i,71}$ та отримують можливість залучення різних технік сканування і їх комбінацій.

Використання протоколу TCP для здійснення спроби підключення до кожного досліджуваного порту відбувається з проходженням повної перевірки встановлення з’єднання. Його суть полягає в трьохетапному підтвердженні за допомогою обміну повідомленнями SYN, SYN/ACK та ACK. Для здійснення

таких дій в функції $f_{3,i,71}$ запускається функція connect(). Якщо вона виконується успішно і повертає нуль, тоді запускається функція $f_{3,i,72}$, в якій міститься виклик функції getservbyport(), яка повертає інформацію про службу, що працює на цьому порту. Вона повертає інформацію про службу, яка працює на цьому порту, у вигляді структури типу servent. Одне з полів цієї структури містить офіційне ім'я служби. Якщо ця функція повертає нульове значення, тоді за номером порту не визначено ім'я служби і в базі відкритих портів робиться відповідна відмітка. Обов'язковим при виконанні функції $f_{3,i,72}$ є передача параметрів: номеру вузла мережі, номер початкового порту для сканування, номер кінцевого порту для сканування. Таким чином, при такому TCP-скануванні через спроби підключитись функції міститимуть такі складові: $f_{3,i,71} = \{nmap, connec\text{t}, \dots\}$, $f_{3,i,72} = \{getservbyport, \dots\}$.

Використовуючи TCP – сканування за допомогою повідомлень SYN, здійснюється сканування з незавершеним відкриттям сеансу, бо не встановлюється повне TCP з'єднання. На порт відправляється повідомлення і отримується підтвердження про те, що він перебуває в режимі очікування. Адміністратор бот-мережі надсилає досліджуваному вузлу повідомлення RST/ACK і повне з'єднання не встановлюється. Цей метод може бути віднесений до прихованих, бо частина системних журналів не фіксують такі спроби. Оскільки функція connect() встановлює повну процедуру встановлення з'єднання, то її при цьому скануванні не використовують. Адміністратор заповнює TCP заголовок, відправляє пакет. IP – заголовок може бути заповнений відповідною підсистемою або адміністратором.

TCP – сканування може бути реалізовано в бот-мережі ще такими методами окремо або їх об'єднанням з подальшим вибором:

1) за допомогою повідомлень FIN - вузлу надсилається пакет і у відповідь вузол згідно RFC793 повинен відправити пакет RST для всіх закритих портів; відсутність пакету означає, що всі порти закриті; для операційних систем Windows цей метод не працює;

2) за допомогою прапорів FIN/URG/PUSH досліджуваному вузлу надсилається пакет і у відповідь вузол згідно RFC793 повинен відправити пакет RST для всіх закритих портів;

3) за допомогою надсилання пакетів з відключеними прапорами і згідно RFC793 досліджуваний вузол повинен відправити повідомлення RST для всіх закритих портів;

4) за допомогою повідомлень ACK для отримання множини правил, які використовує брандмауер; на досліджуваний вузол надсилається ACK – пакет; якщо приходить RST – пакет, тоді порт класифікується як нефільтрований брандмауером, в іншому випадку - як фільтрований.

Крім використання протоколу TCP, на одному і тому ж номері порту можуть працювати служби за протоколами TCP та UDP. Методи TCP сканування не дозволяють виявити UDP – порти, які перебувають саме в стані очікування запиту. Для цього використовують порт-сканер, що працює за UDP – протоколом. Для цього на кожен порт досліджуваного вузла надсилається UDP – пакет та очікується ICMP – відповідь. Відповідь, яка надходить, означає, що порт закритий. Якщо ж відповіді немає, то це означає, що порт відкрито. Для очікування підтвердження того, що відповіді не буде, витрачається певний час. Також, при такому скануванні може бути великий відсоток хибних спрацювань через блокування стандартного повідомлення від досліджуваного порту маршрутизаторами.

Швидшим є метод такого UDP – сканування при якому задіюються віддалені UDP – служби. Але для цього адміністратор бот-мережі повинен закласти механізми правильного генерування запитів до них та правильної їх обробки.

З метою прискорення роботи бот-мережі в частині сканування портів використовують метод, що базується на багатопоточності. Для його реалізації в циклі запускають функцію, що створює багато потоків. Його організація може передбачати запуск потоків почергово, тобто після завершення попереднього запущеного потоку запуск на його місце наступного. Але може бути і така

організація, при якій всі потоки створюються одночасно та здійснюють сканування портів заданим в циклі порядком. При такій організації потрібна синхронізація потоків. Наприклад, з використанням м'ютекса. Оскільки система може надати доступ до процесора будь-якому потоку в будь-який час та в будь-якому місці коду, то це може викликати невірну роботу порт-сканеру. Зокрема, два потоки можуть інкрементувати свою глобальну змінну, яка відповідає за номер порту, а в цей час інший потік виконає з'єднання з віддаленим портом за цим значенням. Тому, фрагмент програми, в якому може відбуватись одночасний доступ потоків, тобто критична секція, обмежують функціями, які відображають критичну секцію, доступ до якої не може отримати жоден потік, поки він не виконається в поточному потоці. Критична секція передбачає встановлення номеру порту для сканування, виконання з'єднання запуском відповідної функції, виведення на екран результату, інкрементування глобальної змінної номеру порту і закриття дескриптору сокету. Організований таким чином порт - сканер працює на основі методу TCP-сканування підключенням з використанням багатопоточності. Для використання такого сканування важливим є досягнення більшої швидкодії. Якщо порівняти розглянути багатопоточний сканер з попередніми методами, то досягнення швидкодії забезпечується використання м'ютекса, в якому виконання функції встановлення зв'язку блокує роботу всіх решти потоків. Але це теж може сповільнювати роботу. Якщо не користуватись механізмом синхронізації, тоді потоки можуть спотворити виконання програми.

Для уникнення недоліків попереднього методу при реалізації функції встановлення зв'язку може бути використано метод неблокованих сокетів. Він полягає в створенні в межах одного процесу багато неблокованих сокетів і дослідженні їх станів. Після виклику функції встановлення зв'язку, для неблокованого сокету ініціюється встановлення з'єднання відправленням першого пакету трьохетапного підтвердження TCP і негайно повертається помилка EINPROGRESS, яка означає що встановлення з'єднання розпочалось, але не завершилось. В зв'язку з цим відслідковування з'єднання потрібно проводити постійно на предмет його успішного виконання. Це вимагає включати

додатково необхідні функції перевірки. Якщо є сервер і клієнт знаходяться на тому ж вузлі, що для випадку бот-мереж є малоймовірно або можливо тільки на первинній стадії її формування, тоді з'єднання буде швидким. Але ймовірність такого випадку в порівнянні з рештою дуже мала. Тому, розробники бот-мереж закладають відповідні функції перевірки і тим самим формують елементи поведінкової сигнатури за структурою функції сканування. Перевірка портів на готовність до запису чи читання здійснюється функцією вибору, аргументами якої є макроси для таких перевірок. Позитивний результат таких перевірок вказує, що порт відкрито для запису, тобто сканування відбулось успішно. Після цього порт-сканер може мати декілька станів: сокет не створено, сокет створено, читання банерів, запис рядка у відкритий порт, сокет в очікуванні з'єднання. В командному рядку обов'язково вказують адресу віддаленого вузла, час очікування готовності сокета. Якщо сокет не готовий до читання чи запису, тоді здійснюють перевірку як це довго триває і закривають його та виставляють стан в початкове положення. Проаналізовані методи організації сканування портів в мережі для їх реалізації вимагатимуть використання функції встановлення зв'язку, що можна досліджувати засобами РБС, які включатимуть аналіз прямих і зворотніх повідомлень.

Порти-сканери містять, також, функції для визначення типу та версії операційних систем віддалених вузлів за рахунок дослідження стеку. Для цього використовують технологію дослідження стеку, вважаючи що розробники різних ОС не завжди дотримуються вимог рекомендацій: надсилання пакету з різним переліком параметрів на досліджуваний відкритий та закритий порти, зокрема SYN|FIN|URG|PSH; надсилання NULL – пакету без встановлених прапорців з набором різних параметрів на відкритий порт; надсилання FIN – пакету на відкритий порт (переважно всі ОС не відповідають на цей пакет, але деякі ОС надсилають пакет FIN/ASK); початковий розмір вікна TCP для певних реалізацій стеку є унікальним; перевірка біту фрагментації в IP – заголовках, який деякі ОС встановлюють для підвищення продуктивності; значення ASC, яке в певних реалізаціях стеку IP задається по-різному, зокрема може повертати

отриманий номер послідовності або збільшене на одиницю значення номеру послідовності; надсилання UDP – пакету на закритий порт для визначення кількості повідомлень про помилки за певний час з метою визначення типу ОС, бо деякі ОС обмежують швидкість передачі повідомлень про помилки; вимірювання довжини повідомлень ICMP, бо різними ОС передаються повідомлення про помилки різної довжини. Можуть бути використані також інші можливості команди nmap.

Функція $f_{3,i,72}$ має засоби здійснення пошуку вразливостей в комп'ютерних системах. Відомості про вразливості зберігаються в базі вразливостей, яку наповнює та адмініструє зловмисник. Сканер вразливостей бере записи з бази вразливостей і робить запити до Web-серверу. Інформація про виявлені вразливості файлів і сценаріїв надсилається адміністратору або за наявності необхідних засобів обробляється автоматично, тобто використовується для здійснення атаки. Сканер послідовно проходить по всій базі вразливостей та діапазону адрес. Наприклад, база вразливостей містить такі данні:

/scripts/tools/newdsn.exe

/_vti_pvt/*.*

/catalog_type.asp.

Тоді, сканер передає в командний рядок наступний рядок:

<IP-адресу або ім'я досліджуваного Web-вузла>[:port] [IP-адреса або ім'я проксі-сервера][:port]. Обов'язковою є тільки IP-адреса або ім'я досліджуваного Web-вузла. Далі здійснюється розбір переданих аргументів відділяючи імена вузлів від номерів портів. При відсутності номера порту, за замовчуванням приймається порт 80 для схеми з http, а для схеми з https приймається порт 243. Зчитуючи почергово кожен рядок з бази вразливостей в циклі, сканер здійснює за допомогою функції connect() з'єднання з віддаленим вузлом, в якості якого виступає або Web-вузол або проксі-сервер. Цей спосіб дозволяє отримувати будь-які данні, які генеруються або зберігаються ресурсом. Після надсилання запиту сканер перевіряє отриману відповідь. Якщо в ній є код «200 ОК», то

запитуваний вразливий файл або сценарій присутній на Web – сервері. Хоча можуть бути і такі відповіді: «FOUND!!!», «Not Found», «404 Not Found», «403 Forbidden». Після отримання відповіді сканер розриває з'єднання функцією закриття та переходить до нового циклу повторення, тобто встановлює знову з'єднання для перевірки наступної вразливості або завершує роботу, якщо вибрані на перевірку всі вразливості з його бази.

Для захисту від сканерів адміністратори можуть змінювати сторінку помилки 404 на свою. Це дає можливість програмі кожного разу видавати результат «FOUND!!!» на кожен неіснуючий сценарій або файл, бо йому постійно повертатиметься код 200. Також, на Web – сервері адміністратори можуть розміщувати файли та сценарії такі, що за назвами є вразливими, але насправді такими не є. Тому, розробники бот-мереж закладають глибший пошук і аналіз для уникнення таких обхідних методів виявлення.

Для прискорення роботи сканерів додають багатопоточність або підтримку неблокованих сокетів. Для шифрування трафіка більшість Web – серверів використовують протокол https. Тому, сканер повинен мати підтримку протоколу захищених сокетів (SSL), роботи з списком проксі-серверів та можливість задання діапазону адрес для сканування. Все це відображається в структурі сканера і досліджується засобами РБС.

Обхід систем виявлення вторгнень (IDS) сканером здійснюється шляхом включення в нього таких алгоритмів, які здійснюють заміну / на /./ в запитах, використовують підряд декілька символів /, додають фіктивні шляхи за допомогою рядка ../ (каталог вказаний перед цим рядком ігнорується), додають фіктивні параметри, замінюють символи їх шістнадцятковими кодами, а також комбінують їх.

Сканер повинен обов'язково включати мережні протоколи SOCKS4 і SOCKS5, які дозволяють пересилати пакети від клієнта до сервера через проксі-сервер непомітно для них і, таким чином, задіюючи сервіси поза міжмережних екранів. З'єднання через SOCKS відбувається в два етапи: привітання з можливою аутентифікацією, повідомлення проксі-серверу про пункт

призначення. Привітання здійснюється в формі повідомлення, що надсилається клієнтом зразу після з'єднання з проксі-сервером і має такий вміст: 1 байт – номер версії SOCKS; 1 байт – кількість методів з'єднання/аутентифікації; N байт – перераховані підтримувані клієнтом методи. На привітання проксі-сервер повинен відповісти двома байтами: номером своєї версії, вибраним з надісланої послідовності методом. На наступному кроці клієнт повинен повідомити проксі-серверу з ким він має з'єднатись і яким чином. Для цього надсилаються такі данні в пакеті: один байт, що вказує номер версії; один байт, який відображає команду; один зарезервований байт, який завжди дорівнює 0x00; N байт, які вказують адресу віддаленого вузла; два байти, які вказують номер порту віддаленого вузла. Якщо з'єднання відбулось успішно, то проксі-сервер переходить в режим передачі будь-яких даних за адресою, яка вказана на другому кроці. Представимо з'єднання через SOCKS такими послідовностями, перша з яких така:

$$z_{1,p} = \left(n_{p,v,socks}, n_{p,v}, k_{p,metod}, s_{p,metod}\right), \quad (4.4)$$

де $n_{p,v,socks}$ – номер підтримки з'єднання через SOCKS, який приймає значення 4 або 5, $socks$ – означає індекс, що показує відношення відповідної змінної до номеру версії; $n_{p,v}$ – номер версії, ця компонента послідовності займає один байт і приймає значення 0x05 для п'ятої версії та 0x04 для четвертої версії; p – означає повідомлення; v – відповідає за означення версії; $k_{p,metod}$ – вказує кількість методів, значення 0x00 означає, що клієнт підтримує з'єднання без аутентифікації, значення 0x02 означає, що за вимогою можна видати username і password; $metod$ – означає індекс, що показує відношення змінної k до кількості методів; $s_{p,metod}$ – містить перераховані методи, $s_{p,metod} = \left(s_{p,metod,1}, s_{p,metod,2}, \dots, s_{p,metod,k_{p,metod}}\right)$ і займає певну кількість байт.

Друга послідовність містить такі параметри:

$$z_{2,p} = \left(n_{p,v,socks}, n_{p,v}, k_{p,komanda}, b, t_{p,metod}, n_{p,adresa}, p_{p,port}\right), \quad (4.5)$$

де $n_{p,v,socks}$ – номер підтримки з'єднання через SOCKS, приймає значення 4 або 5; $socks$ – означає індекс, що показує відношення відповідної змінної до номеру версії; $n_{p,v}$ – номер версії, ця компонента послідовності займає один байт і приймає значення 0x05 для п'ятої версії та 0x04 для четвертої версії; p – означає повідомлення; v – відповідає за означення версії; $k_{p,komanda}$ – вказує команду, яка може приймати значення 0x01 - означає вказівку організувати з'єднання, 0x02 – команда bind, 0x03 – для роботи по UDP – протоколу для версії 5; $komanda$ – означає відношення до команди; b – зарезервований байт, який завжди дорівнює 0x00, $t_{p,metod}$ – вказує на тип адреси, який буде розміщено після зарезервованого байту, і повідомляє SOCKS – серверу, в якому вигляді буде передаватись адреса віддаленого вузла, при цьому байт може приймати значення 0x01 - IPv4 заданий чотирма байтами в мережному порядку, 0x03 – ім'я вузла у вигляді звичайного рядка, тоді SOCKS – сервер повинен самостійно перетворювати його в IP-адресу, 0x04 - IPv6 в мережному порядку; $metod$ – означає індекс, що показує відношення змінної до методів; $n_{p,adresa}$ – адреса віддаленого вузла; $adresa$ – вказує на адресу для індекса; $p_{p,port}$ – порт віддаленого вузла; $port$ – вказує на порт для змінної.

Відповідь на такі пакети SOCKS – сервер надсилає пакетами такої ж структури, але з іншими значеннями. Аналіз цих значень дозволяє регламентувати подальші дії бот-мережі. Зокрема, якщо $n_{p,v} \neq 0$, то з'єднання необхідно розірвати, бо виникла помилка при його встановленні. Тип адреси і сама адреса можуть змінитись, якщо в запиті була надіслана не перетворена адреса. При цьому має повернутись IP-адреса. Якщо з'єднання встановилось, тоді проксі-сервер передає данні за адресою. Для організації цієї передачі визначають структуру пакету, який буде надсилатись на другому етапі.

Стадія інфікування нових об'єктів здійснюється після знаходження комп'ютерної системи, аналіз засобів якої показав, що її можливо інфікувати, та може здійснюватись такими функціями:

$f_{3,i,74}$ – функція вибору інструментів для інфікування на основі

проведеного аналізу комп'ютерної системи (в т.ч. може пропонувати користувачу переглянути листи та сайти);

$f_{3,i,75}$ – функція здійснення інфікування комп'ютерних систем;

$f_{3,i,76}$ – функція закріплення зловмисного програмного забезпечення в комп'ютерній системі;

$f_{3,i,77}$ – функція здійснення інфікування комп'ютерної системи.

Реалізація функції $f_{3,i,74}$ можлива на основі підбору паролів і інших механізмів проникнення в комп'ютерну систему.

Стадія атаки формується набором команд, які отримуються з командно-контролюючого центру і є функції для виконання цих команд. Стадію атаки можуть формувати такі функції:

$f_{3,i,78}$ – функція здійснення обробки команд від об'єкту з підмножини $E_{2,i}$ для проведення атаки;

$f_{3,i,79}$ – функція формування команд для виконання стосовно вибраного об'єкту для атаки;

$f_{3,i,80}$ – функція здійснення атаки комп'ютерної системи.

В процесі завершення функціонування бот-мережі адміністратор переводить її до стадії припинення функціонування за допомогою таких функцій:

$f_{3,i,81}$ – функція знищення об'єкту з підмножини $E_{3,i}$ бот-мережі.

Для аналізу базових елементів бот-мережі розглянемо їх життєвий цикл у взаємозв'язку між трьома рівнями. На рис. 4.3 зображено граф-схемою стадії функціонування бот-мережі з виділенням станів, в яких перебувають елементи різних рівнів.

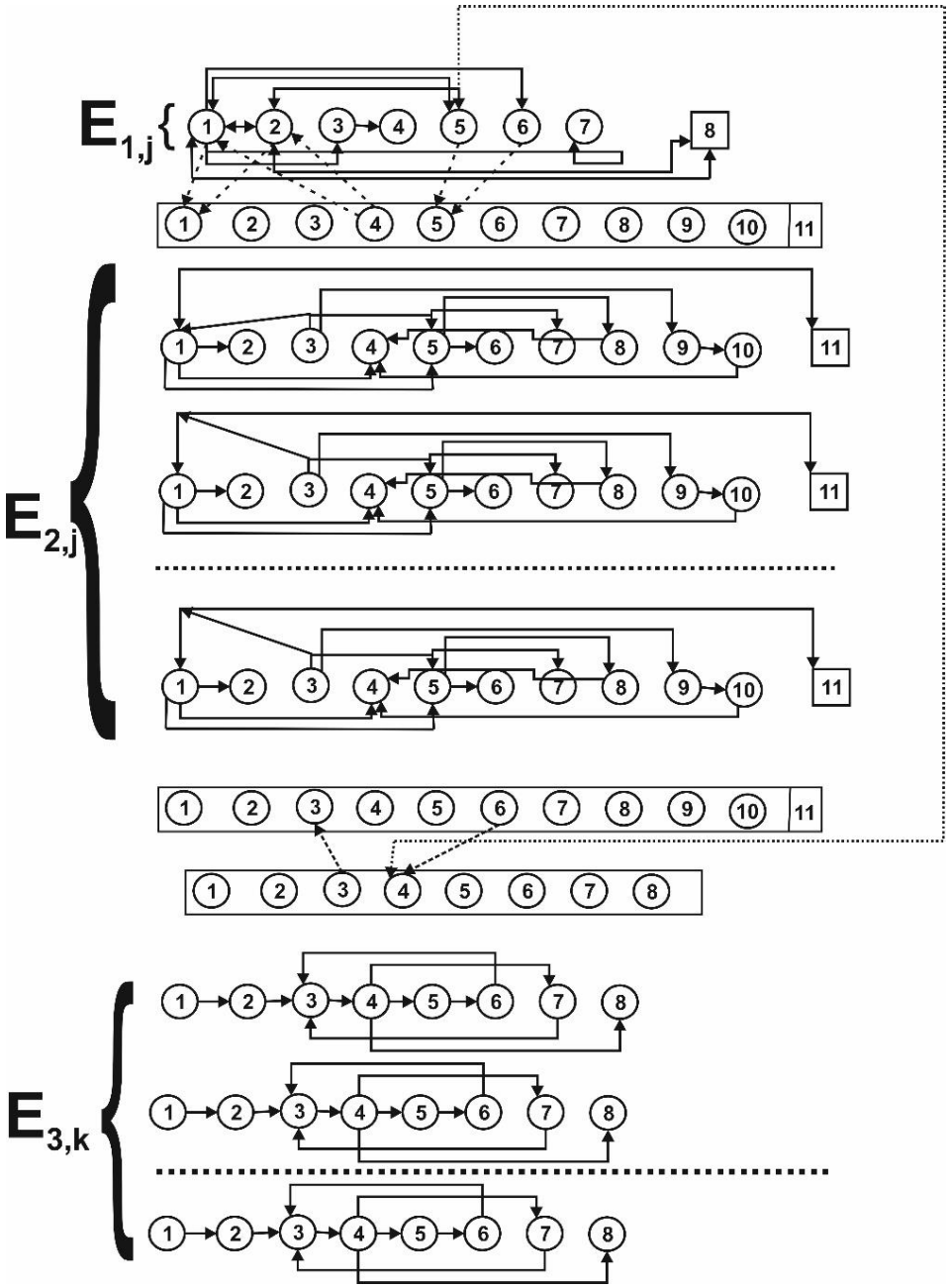


Рис. 4.3. Граф-схема стадій функціонування бот-мережі

Запропонована еталонна модель бот-мереж, в основі якої закладені типові компоненти, що виражаються відповідними функціями. Розроблені функції відображають типові дії мережного зловмисного програмного забезпечення в локальних мережах і комп'ютерних системах безпосередньо. Їх представлення дозволить виділити характеристичні ознаки бот-мережі і є основою для розробки методів їх виявлення та локалізації.

4.2. Характеристики компонентів моделей бот-мереж та їх представлення

Розроблена еталонна модель бот-мереж на основі типової архітектури (рис. 4.2) представлена функціями, в яких відображені поведінки бот-мереж [163, 164, 200]. Функції виступатимуть основою характерних ознак бот-мереж. Тому, здійснимо їх формалізацію для типового представлення і оптимального використання. З цією метою для розроблених функцій формуємо відображення їх дій в масиви, в яких буде відображатись інформація про наявність таких активностей, які можуть бути віднесені до бот-мереж. Формуємо послідовності з компонентами, які є функціями або групами функцій, що відносяться до одного з рівнів, а також групуємо дві-три послідовності згенерованих різними рівнями. Ці еталони закладені в базу функцій бот-мереж. Поява хоча б однієї послідовності вимагає віднесення її до тих, які потребуватимуть контролю на всіх КС мережі, що досягається за рахунок роботи розподіленої багаторівневої системи. Для деталізації зв'язків між різними рівнями бот-мереж, комп'ютерними системами та атакованими ресурсами представимо їх схемою переходів між рівнями бот-мережі на основі виконання заданих функцій $f_{1,i,1} - f_{3,i,81}$, зображеною на рис. 4.4.

На рис. 4.4 відображено по одному різному вузлу бот-мережі. Функціональне навантаження кожного етапу життєвого циклу бот-мережі через наявні функції в її стадіях впливає на життєвий цикл всієї бот-мережі. Відображення поточного стану бот-мережі здійснимо на основі встановлення відповідності кожного етапу функціонування кожного вузла бітовим матрицям стану бот-мережі і задамо формулою (4.6):

$$m_{q,i,c} = \begin{cases} 0, \text{ якщо } c - \text{та стадія функціонування } i \\ -\text{го вузла бот мережі неактивна} \\ 1, \text{ якщо } c - \text{та стадія функціонування } i \\ -\text{го вузла бот мережі активна} \end{cases}, \quad (4.6)$$

де q – рівень бот-мережі; $q = 1, 2, 3$; $m_{q,i,c} \in M_q$, M_q – матриця для q – го рівня бот-мережі.

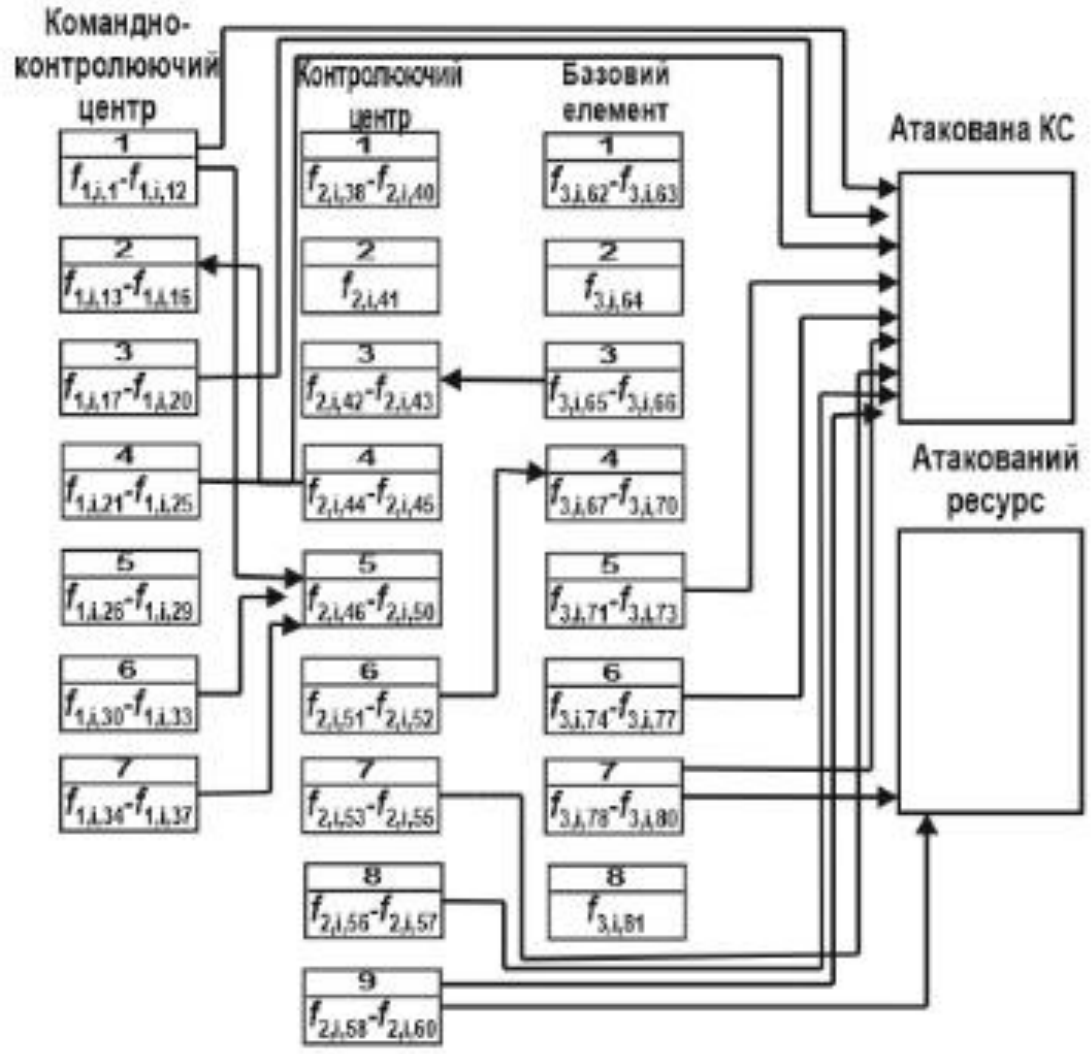


Рис. 4.4. Схема переходів між рівнями бот-мережі на основі виконання функцій

Числа 1-10 вказують на номер стадії функціонування командно-контролюючого та контролюючих центрів і базового елементу бот-мережі, які відображають відповідні елементи рис. 4.2 (а, б, в); функції в кожному блоці вказують на наповнення кожного етапу життєвого циклу.

Таку інформацію отримує зловмисник в процесі експлуатації бот-мережі і вона зберігається у відповідних базах бот-мережі. Крім того, з метою виявлення базових елементів бот-мережі в локальних комп'ютерних мережах на основі їх життєвого циклу така інформація може бути використаною для здійснення

порівняння їх виконання в різних КС в локальній мережі, тобто для виявлення подібної поведінки в процесі їх функціонування. Задамо відображення елементів множин, які формують стадії функціонування кожного рівня бот-мереж. Елементи $e_{q,i,r} \in E_{q,i}$, де q - рівень бот-мережі, $q = 1, 2, 3$, i – кількість елементів $E_{q,i}$ на q – тому рівні, r – кількість стадій функціонування вузла бот-мережі довільного рівня. Якщо елемент $e_{1,i,r} \in E_{1,i}$, то для структури бот-мережі з рис. 4.2 $r = 7$ і, тоді, справедливі такі відношення між елементами $e_{1,i,r}$ та функціями першого рівня, які виражаються формулами (4.7):

$$\begin{aligned}
e_{1,i,1} &= \{f_{1,i,1}, f_{1,i,2}, \dots, f_{1,i,12}\},\\
e_{1,i,2} &= \{f_{1,i,13}, f_{1,i,14}, \dots, f_{1,i,16}\},\\
e_{1,i,3} &= \{f_{1,i,17}, f_{1,i,18}, \dots\ f_{1,i,20}\},\\
e_{1,i,4} &= \{f_{1,i,21}, f_{1,i,22}, \dots, f_{1,i,25}\},\\
e_{1,i,5} &= \{f_{1,i,26}, f_{1,i,27}, \dots, f_{1,i,29}\},\\
e_{1,i,6} &= \{f_{1,i,30}, f_{1,i,31}, f_{1,i,33}\},\\
e_{1,i,7} &= \{f_{1,i,34}, f_{1,i,35}, \dots, f_{1,i,37}\}.
\end{aligned} \tag{4.7}$$

Аналогічно для елементів $e_{2,i,r} \in E_{2,i}$ значення $r = 10$ і для елементів $e_{3,i,r} \in E_{3,i}$ значення $r = 8$. Відображення $V_1: e_{q,i,c} \dashrightarrow m_{q,i,c}$ встановлює відповідність стадій функціонування кожного вузла кожного рівня бот-мереж бітовим векторам.

Функційне навантаження кожної з виокремлених за призначенням функцій залежить від типу операційних систем та їх API-функцій відповідно. Представимо функції, що формують функціонування бот-мереж, через API-виклики [389] бітовими послідовностями. Ці представлення залежатимуть від типів та конкретних операційних систем комп’ютерних систем локальної мережі, і відповідно будуть різними. Для цього виділимо особливі дії, які описуватимуться відповідними API-функціями, що можуть бути віднесені до зловмисних дій бот-мережі. Всі ці дії впливатимуть на атаковані КС та атаковані

ресурси. З рис. 4.4 видно, що на певних етапах функціонування бот-мереж викликаються функції, які здійснюють вплив, що полягає в наступному: сканування портів, розсилання спаму, завантаження файлів, перегляд директорій, переміщення файлів або директорій, створення перенаправлюваних портів, відкриття файлів, знищення процесу, виконання файлів, запис даних про натисненні клавіші, створення і знищення директорій, перегляд директорій, здійснення атак, створення хибного проксі-сервера та інші. Функції, які здійснюють такі впливи, присутні в усіх трьох рівнях бот-мереж. Але їх застосування залежить від етапів розбудови бот-мереж. Зокрема, функції першого рівня не використовуються після створення елементів другого рівня. А функції другого рівня можуть використовуватись, коли створено елементи третього рівня, а можуть вже і не використовуватись. Тоді, використовуватимуться функції третього рівня для здійснення атаки на нову КС з метою її включення в бот-мережу та для атаки на задані ресурси для вирішення зловмисних дій. Всі ці функції мають однакове функційне навантаження для вирішення поставлених задач атаки на КС і відрізняються лише часовими межами їх застосування, тому розглядатимемо їх як одну підмножину, яку можна задати так: $\{e_{1,i,1}, e_{1,i,3}, e_{1,i,4}\}$, $\{e_{2,i,7}, e_{2,i,8}, e_{2,i,9}\}$, $\{e_{3,i,5}, e_{3,i,6}, e_{3,i,7}\}$. Для атаки на задані ресурси використовують елементи другого та третього рівнів, які теж мають однакове функційне навантаження, але приймають участь в атаці одночасно, та їх можна задати так: $\{e_{2,i,9}\}$, $\{e_{3,i,7}\}$. Позначимо ці дії, які реалізовані у відповідних елементах $e_{q,i,r} \in E_{q,i}$, де q - рівень бот-мережі, $q = 1, 2, 3$, i – кількість елементів $E_{q,i}$ на q – тому рівні, r – кількість стадій функціонування вузла бот-мережі довільного рівня, через вектор зловмисних дій та атак. Позначимо його v_z і задамо так: $v_z = (v_{z,1}, v_{z,2}, \dots, v_{z,z_v})$, де z_v – кількість зловмисних дій та атак. Компонентами вектора є елементи множини $\{0; 1\}$. Відображення V_2: зловмисні дії та атаки $\dashrightarrow v_{z,i}$, де i – та зловмисна дія або атака, встановлює відповідність дій і компонентів вектору.

Для представлення команд API-функцій певної операційної системи, які

обов'язково повинні бути використані при виконанні функцій, що формують зловмисні дії або атаки, здійснимо їх задання через вектор API-функцій. Позначимо його v_{API} і задамо так: $v_{API,x} = (v_{API,x,1}, v_{API,x,2}, \dots, v_{API,x,z_{API}})$, де z_{API} – кількість API-функцій певної операційної системи, які можуть бути використанні для виконання зловмисних дій чи атак, x – кількість операційних систем, $v_{API,x,i}$ – компонента вектору, яка відповідає за позначення i – ої API-функції. Компонентами вектору є позначення API-функцій. Відображення V_3: API − функції ⇢ $v_{API,x,i}$, де i – та API-функція x – ої операційної системи, встановлює відповідність функцій і компонентів вектору.

Для формування бази зловмисних дій та атак на основі функцій сформуємо матрицю відношень зловмисних дій і атак до API функцій через які вони можуть бути реалізовані з врахуванням типів операційних систем. Задамо відображення V_4: $v_{z,i} \times v_{API,x,i}$ ⇢ $\{0; 1\}$, яке встановлює зв'язок між компонентами векторів зловмисних дій та функціями, якими вони можуть бути реалізовані. Кількість таких матриць, утворених відображенням V_4 залежить від кількості актуальних операційних систем, які використовуються в комп'ютерних системах. Розмірність матриць залежить від кількості підтримуваних API-функцій для реалізації зловмисних дій. Тобто, отримані таким чином матриці мають різну розмірність і їх кількість може змінюватись. Наявність таких матриць є основою для формалізації і побудови еталонів зловмисних дій та атак.

В процесі виконання зловмисних дій та атак на КС та ресурси бот-мережа задіює функції, які обов'язково вимагатимуть зворотньої реакції від об'єктів атаки. Дослідження таких відповідей у формі пакетів від них дозволяє бот-мережам уточнювати варіанти ведення атак. Зворотні реакції надсилаються тим елементам відповідних рівнів бот-мереж з яких надсилались певні пакети, тому їх обробка здійснюється ними. Введемо вектор зворотніх реакцій $v_{API,x,i,j}$, який формуватиметься на основі відповідей на виконані функції, які представлені вектором $v_{API,x,i}$, де j – кількість можливих відповідей на виконану i – ту API-функцію x – ої операційної системи. Задамо відображення V_5: $v_{API,x,i} \times j$ ⇢

$v_{API,x,i,j}$, яке встановлює зв'язок між компонентами векторів API-функція x – ої операційної системи та кількості відповідей до результату відповіді на зловмисні дії та функції, які використані для впливу на атаковані КС та ресурси. Кількість таких матриць, утворених відображенням V_5 залежить від кількості векторів API функцій. Подальші дії бот-мережі будуть залежати від відповідей, які вони отримають від атакованих КС чи ресурсів, але їх кількість обмежена тими функціями, які використовуються, тому можливе повторне використання тих же функцій або інших з наявних чи зупинка атаки на досліджувані КС чи ресурс.

Для кожної функції, які формують наповнення елементів $e_{q,i,r} = \{f_{q,i,1}, f_{q,i,2}, \dots, f_{q,i,r}\} \in E_{q,i}$, де q – рівень бот-мережі, $q = 1, 2, 3$, i – кількість елементів $E_{q,i}$ на q – тому рівні, r – кількість стадій функціонування вузла бот-мережі довільного рівня, через вектор зловмисних дій та атак, різних рівнів бот-мережі, визначаються зловмисні дії і/або атаки, будуються їх вектори та їх представлення через API-функції і їх подання бінарними векторами. А також, встановлюється зв'язок між ними, який буде характеризувати елементи бот-мережі як компоненти для здійснення їх ідентифікації.

Здійснимо формалізацію зловмисних дій та атак для представлення підмножин $E_{q,i}$ де q – рівень бот-мережі, $q = 1, 2, 3$, i – кількість елементів $E_{q,i}$ на q – тому рівні з врахуванням декомпозиції бот-мережі на функції $f_{q,i,1} - f_{q,i,81}$ і їх призначення.

Для цього узагальнимо введені вектори для кожної функції. Зокрема, вектор зловмисних дій та атак для відповідної функції з функцій $f_{q,i,1} - f_{q,i,81}$ позначимо $v_{z,f_{q,i,r}}$ і задамо так: $v_{z,f_{q,i,r}} = \left(v_{z,1,f_{q,i,r}}, v_{z,2,f_{q,i,r}}, \dots, v_{z,z_v,f_{q,i,r}}\right)$, $v_{z,i,f_{q,i,r}}$ – компонента вектору $v_{z,f_{q,i,r}}$, вектор для представлення команд API-функцій певної операційної системи, що формують зловмисні дії або атаки, позначимо $v_{API,f_{q,i,r}}$ і задамо так: $v_{API,x,f_{q,i,r}} = \left(v_{API,x,1,f_{q,i,r}}, v_{API,x,2,f_{q,i,r}}, \dots, v_{API,x,z_{API},f_{q,i,r}}\right)$, $v_{API,x,i,f_{q,i,r}}$– компонента вектору $v_{API,x,f_{q,i,r}}$, бінарну матрицю відношень зловмисних дій і атак до API-функцій через які вони можуть бути реалізовані з

врахуванням типів операційних систем задамо відношенням так: $v_{z,i,f_{q,i,r}} \times v_{API,x,i,f_{q,i,r}} \dashrightarrow \{0;1\}$, вектор зворотніх реакцій $v_{API,x,i,j,f_{q,i,r}}$, де q - рівень бот-мережі, $q = 1,2,3$, i – кількість елементів $E_{q,i}$ на q – тому рівні, r – кількість стадій функціонування вузла бот-мережі довільного рівня, через вектор зловмисних дій та атак, z_v – кількість зловмисних дій та атак, z_v – кількість зловмисних дій та атак, $z_{API,x,f_{q,i,r}}$ – кількість API-функцій певної операційної системи, які можуть бути використанні для виконання зловмисних дій чи атак, x – кількість операційних систем, $v_{API,x,i,f_{q,i,r}}$ – компонента вектора, яка відповідає за позначення i – тої API-функції, i – та API-функція x – ої операційної системи, встановлює відповідність функцій і компонентів вектора, j – кількість можливих відповідей на виконану i – ту API-функцію x – ої операційної системи.

Для вектору представлення команд API-функцій певної операційної системи $v_{API,f_{q,i,r}}$, що формують зловмисні дії або атаки, задамо відношення V_6: $v_{API,f_{q,i,r}} \dashrightarrow v_{API,f_{q,i,r},K}$, яке встановлює відповідність кожної команди API-функції кількості її входження у вектор $v_{API,f_{q,i,r}}$, тобто компонентами вектору $v_{API,f_{q,i,r},K}$ є числові величини.

Дії бот-мережі, які не направлені на здійснення зловмисних атак, та дії з її адміністрування також відносяться до зловмисних, бо відбуваються поза контролем користувача і ним не здійснюються. Тому, важливим є наповнення всіх функції $f_{q,i,1} - f_{q,i,81}$. Кожна з них має свої особливості, врахування яких в сукупності дозволить будувати шаблони для виявлення бот-мереж.

Виділимо детальніше функції, які задіюються безпосередньо для виконання зловмисних дій та атак.

Функція $f_{1,i,7}$ включає інструментарій для пошуку вразливостей інших КС. Представимо ці механізми компонентами вектору зловмисних дій та атак: $v_{z,1,f_{1,i,7}}$ – здійснення сканування ввімкнених КС в мережі, $v_{z,2,f_{1,i,7}}$ – здійснення визначення операційної системи певної визначеної КС в мережі, $v_{z,3,f_{1,i,7}}$ – здійснення сканування портів КС в мережі.

Розглянемо наповнення функції $f_{1,i,9}$. Вона може включати такі зловмисні дії та атаки, які представимо як компоненти відповідного вектору: $v_{z,1,f_{1,i,9}}$ – здійснення спам-атак, $v_{z,2,f_{1,i,9}}$ – здійснення фішинг-атак (для отримання логіна і пароля КС), $v_{z,3,f_{1,i,9}}$ – здійснення завантаження зловмисних файлів в КС.

Функція $f_{1,i,10}$ здійснює створення нового вузла бот-мережі і її вектор має такі компоненти: $v_{z,1,f_{1,i,10}}$ – здійснення завантаження у визначену КС мережі файлів, які формують програмне забезпечення нового вузла, $v_{z,2,f_{1,i,10}}$ – здійснення активації завантаженого програмного забезпечення у визначеній КС, $v_{z,3,f_{1,i,10}}$ – здійснення сканування портів КС в мережі.

Функція $f_{1,i,17}$ містить набір інструментів для пошуку вразливостей в КС з метою подальшого її використання в якості нового вузла бот-мережі і її вектор має такі компоненти: $v_{z,1,f_{1,i,17}}$ – здійснення спонукання користувача для переходу на підготовлений зловмисником веб-сайт, з якого в КС встановлюється зловмисний код, $v_{z,2,f_{1,i,17}}$ – здійснення спонукання користувача КС надати його облікову інформацію на основі методів соціальної інженерії, $v_{z,3,f_{1,i,17}}$ – електронні листи з вкладеними візуально корисними посиланнями для виконання зловмисного коду, $v_{z,4,f_{1,i,17}}$ – посилання на зловмисний веб-сайт із наявним троянським кодом, $v_{z,5,f_{1,i,17}}$ – посилання на зловмисний веб-сайт, який є копією потрібного користувачу веб-сайту із наявним троянським кодом, $v_{z,6,f_{1,i,17}}$ – електронні листи з повідомленнями про необхідність термінового перегляду і відповіді на них з вкладеним в них зловмисним кодом, який завантажується в КС, $v_{z,7,f_{1,i,17}}$ – пошук в КС залишених раніше іншими зловмисниками стандартних зловмисних програм для проникнення в КС (наприклад, бекдорів).

Функція $f_{1,i,28}$ здійснює передачу повідомлень вузлам другого рівня і її вектор має такі компоненти, які залежать від вибраного протоколу прихованої передачі повідомлень: $v_{z,1,f_{1,i,28}}$ – здійснення вибору протоколу передачі

прихованого повідомлення, де кількість та використання протоколів регламентується зловмисником, $v_{z,2,f_{1,i,28}}$ – здійснення передачі прихованого повідомлення вузлу другого рівня у визначеній КС, $v_{z,3,f_{1,i,28}}$ – здійснення повідомлення від вузла КС в мережі згідно протоколу. Відобразимо тип протоколу в компонентах цього ж вектору, починаючи з четвертої. Він теж задаватиметься певними функціями при реалізації по-різному, тому матиме і свій перелік API-функцій.

Наповнення функції $f_{1,i,37}$ залежатиме від типу подій, які призвели до необхідності знищення командно-контролюючого центру бот-мережі, і можуть бути викликаними виявленням центру сторонніми особами з аварійним видаленням або прийнятим рішенням власника мережі і здійснення цих дій коректно. Компоненти вектору, який відображає ці події, задамо так: $v_{z,1,f_{1,i,37}}$ – аварійне видалення, $v_{z,2,f_{1,i,37}}$– коректне видалення, яке передбачає перехід до нового центру або попереднє видалення всіх вузлів бот-мережі.

Наповнення функції $f_{3,i,63}$ формується з метою забезпечення контролю над КС та впливу на її програмне забезпечення, в якій міститься сам вузол бот-мережі. Компоненти вектору, який відображає ці події, задамо так: $v_{z,1,f_{3,i,63}}$ – перевірка наявності блокування антивірусних засобів, захист яких було подолано при інфікуванні КС, $v_{z,2,f_{3,i,63}}$- здійснення блокування або закриття портів, $v_{z,3,f_{3,i,63}}$ – здійснення відслідковування натиснень клавіш клавіатури, $v_{z,4,f_{3,i,63}}$ – збереження зображення екрану, $v_{z,5,f_{3,i,63}}$ – здійснення збереження даних з веб-камери, $v_{z,6,f_{3,i,63}}$ – здійснення визначення електронної пошти, яку використовує користувач КС, $v_{z,7,f_{3,i,63}}$ – здійснення пошуку електронних адрес, $v_{z,8,f_{3,i,63}}$ – пошук паролів та інших конфіденційних ключів і їх збереження, $v_{z,9,f_{3,i,63}}$ – здійснення збереження списку користувачів КС, $v_{z,10,f_{3,i,63}}$ – здійснення збереження користувачів певних електронних ресурсів, $v_{z,11,f_{3,i,63}}$ – деактивація антивірусних засобів, $v_{z,11,f_{3,i,63}}$ – здійснення призупинення служб і

процесів, $v_{z,12,f_{3,i,63}}$ – здійснення запуску необхідних служб і процесів, $v_{z,13,f_{3,i,63}}$ – здійснення зміни конфігураційних файлів, $v_{z,14,f_{3,i,63}}$ – здійснення відкриття системних портів, $v_{z,14,f_{3,i,63}}$ – здійснення закриття системних портів.

Наповнення функції $f_{3,i,64}$ формується з метою забезпечення перевірки контролю вузла 3 - го рівня бот-мережі і відповідно компоненти вектору, який відображає ці події, задамо так: $v_{z,1,f_{3,i,64}}$ – здійснення перевірки файлів КС з останнім списком оновлень, $v_{z,2,f_{3,i,64}}$ - здійснення перевірки програмних компонент вузла бот-мережі в КС, $v_{z,3,f_{3,i,64}}$ - здійснення сканування оперативної пам'яті КС на наявність запущених процесів.

Функція $f_{3,i,65}$ формується з метою підготовки повідомлень вузлу 2 - го рівня бот-мережі, зокрема і для підтримки цілісності бот-мережі, і відповідно компоненти вектору, який відображає ці події, задамо так: $v_{z,1,f_{3,i,65}}$ – перевірка автентичності пароля, $v_{z,2,f_{3,i,65}}$ – здійснення запиту на оновлення програмних компонент вузла бот-мережі в КС, $v_{z,3,f_{3,i,65}}$ – здійснення запиту на оновлення всього програмного забезпечення вузла 3 – го рівня, $v_{z,4,f_{3,i,65}}$ – здійснення запиту на оновлення списку підключень в мережі, $v_{z,5,f_{3,i,65}}$ - здійснення запиту на конфігурацію вузла, $v_{z,6,f_{3,i,65}}$ – здійснення запиту на перезапуск програмного забезпечення вузла, $v_{z,7,f_{3,i,65}}$ – здійснення запиту на оновлення певних функцій атаки для вузла, $v_{z,8,f_{3,i,65}}$ – здійснення запиту про статус вузла, $v_{z,9,f_{3,i,65}}$ – здійснення запиту про виведення інформації про користувача КС, $v_{z,10,f_{3,i,65}}$ – здійснення запиту на відображення поточних підключень до вузла інших компонент бот-мережі, $v_{z,11,f_{3,i,65}}$ – здійснення запиту на зміну пароля користувача, $v_{z,12,f_{3,i,65}}$ – здійснення запиту на зміну власного пароля вузла, $v_{z,13,f_{3,i,65}}$ – здійснення запиту на автентифікацію пароля, $v_{z,14,f_{3,i,65}}$ – здійснення запиту на підтвердження атаки.

Функція $f_{3,i,80}$ забезпечує виконання атаки. Компоненти вектору, який відображає ці події, задамо так: $v_{z,1,f_{3,i,80}}$ – здійснення генерування запитів до

визначеного ресурсу на основі підготовлених шаблонів, $v_{z,2,f_{3,i,80}}$ – здійснення надсилання визначених зловмисником адресатам електронних листів, $v_{z,3,f_{3,i,80}}$ – здійснення надсилання визначеним зловмисником адресатам повідомлень через соціальні мережі, $v_{z,4,f_{3,i,80}}$ – здійснення надсилання визначеним зловмисником адресатам повідомлень через менеджери миттєвих повідомлень, $v_{z,5,f_{3,i,80}}$ – здійснення надсилання визначеним зловмисником адресатам повідомлень через програми IP-телефонії, $v_{z,6,f_{3,i,80}}$ – здійснення сканування визначених зловмисником ввімкнених активних в мережі КС, $v_{z,7,f_{3,i,80}}$ – здійснення визначення операційної системи атакованої КС, $v_{z,8,f_{3,i,80}}$ – здійснення сканування портів атакованої КС, $v_{z,9,f_{3,i,80}}$ – здійснення надсилання визначеним зловмисником адресатам повідомлень через менеджери миттєвих повідомлень, $v_{z,10,f_{3,i,80}}$ – здійснення атаки на відмову ресурсу, $v_{z,11,f_{3,i,80}}$ – здійснення виконання завантаження файлів на визначену КС, $v_{z,12,f_{3,i,80}}$ – здійснення виконання під'єднання до новоствореного вузла, $v_{z,13,f_{3,i,80}}$ – здійснення підбору пароля для КС.

Функція $f_{3,i,81}$ формується з метою знищення вузла 3-го рівня бот-мережі і відповідно компоненти вектору, який відображає ці події, задамо так: $v_{z,1,f_{3,i,81}}$ – здійснення запуску функції, яка знімає блокування з антивірусних засобів, $v_{z,2,f_{3,i,81}}$ – знищення файлів, які формували вузол бот-мережі, $v_{z,3,f_{3,i,81}}$ – очистка місця в пам'яті, в якому розміщувалась резидентна функція вузла бот-мережі.

Зловмисні дії можуть визначатись декількома API-функціями, а також різними їх варіаціями. Для врахування такого можливого представлення зловмисних дій задамо вектори $v_{API,x,f_{q,i,r}}$, $v_{API,x,i,f_{q,i,r}}$, $v_{API,f_{q,i,r},K}$ так, щоб вони відображали варіації: $v_{API,x,f_{q,i,r},l,m}$, $v_{API,x,i,f_{q,i,r},l,m}$, $v_{API,f_{q,i,r},K,l,m}$, де l – номер компоненти, m – номер варіації можливого представлення зловмисної дії.

Вибір API-функцій для формування відповідних векторів здійснимо тільки з тих, які можуть бути використані при реалізації однієї з функцій $f_{q,i,1} - f_{q,i,81}$,

оскільки функція може бути представлена декількома векторами, а тому і включатиме тільки ті, які використовуватимуться. Це дозволяє оптимізувати представлення, на відміну від використання повної множини API-функцій певної операційної системи. При доповненні функції новими можливими реалізаціями здійснюють оновлення векторів, розширюючи чи зменшуючи кількість їх компонентів.

Важливими для представлення векторів зловмисних дій API-функціями є функції, які виконують відкриття файлу, закриття файлу, виконання програми, завершення процесу, зв'язок з IP-адресами, зв'язок з потенційно підозрілими портами, здійснення підключення до IP-адрес, здійснення підключення до портів, передача вмісту буфера, передача до потенційно підозрілих портів, встановлення ключів регістру, видалення ключів регістру, створення служб, відкриття служб. Всі ці функції реалізовані відповідними API-функціями конкретних операційних систем. При їх дослідженні ПМ РБС необхідним є встановлення можливих аргументів. При побудові сигнатури зловмисного коду замість використання всіх API-викликів, що здійснює вузол бот-мережі, з метою оптимізації виділимо для розгляду лише критичні API-функції. Критичні API-виклики містять усі виклики API, які можуть призвести до порушення безпеки, зміни усталеної поведінки роботи системи або виклики, що використовуються для комунікації (модифікація значення системного реєстру, файлового вводу-виводу, API-доступу до мережних ресурсів (WinSock), тощо). В процесі створення сигнатури не розглядатимемо API-виклики, які можна додати або вилучати з програми без зміни її зловмисної поведінки (наприклад MessageBox, printf, malloc тощо).

Таким чином, представлення бот-мережі через можливі її структурні компоненти [390] та їх реалізації дозволяє використовувати його як еталонні ознаки з метою виявлення та локалізації вузлів бот-мереж в комп'ютерних системах локальних мереж. Представлення бот-мережі задано множиною векторів, інформація про компоненти яких розподілена між векторами і подана в табл. 4.2.

Таблиця 4.2

Характеристики компонентів бот-мережі

Складові бот-мережі	Опис, поняття	Формули
1	2	3
E	бот-мережа	$E = \bigcup_{i_1=1}^{n_1} E_{1,i_1} \bigcup_{i_2=1}^{n_2} E_{2,i_2} \bigcup_{i_3=1}^{n_3} E_{3,i_3}$
$E_{q,i}$	вузол бот-мережі	$e_{q,i,r} \in E_{q,i}$
$e_{q,i,r}$	елемент вузла	$e_{q,i,r} = \{f_{q,i,1}, f_{q,i,2}, \dots, f_{q,i,n_r}\}$
$m_{q,i,c}$	бітова матриця відображає числову характеристику вузла: стан	$V_1: e_{q,i,c} \dashrightarrow m_{q,i,c}$
$f_{1,i,1} - f_{3,i,n_f}$	функції, які формують елементи вузлів	$v_{z,f_{q,i,r}}$
$v_{z,f_{q,i,r}}$	вектор зловмисних дій і атак	V_2: зловмисні дії та атаки $\dashrightarrow v_{z,f_{q,i,r}}$
$v_{z,1,f_{q,i,r}}$	компонента вектора $v_{z,f_{q,i,r}}$	$v_{z,f_{q,i,r}} = \left(v_{z,1,f_{q,i,r}}, v_{z,2,f_{q,i,r}}, \dots, v_{z,z_v,f_{q,i,r}}\right)$
$v_{API,x,f_{q,i,r},m_u,l}$	вектор API-функцій	V_3: API − функції $\dashrightarrow v_{API,x,f_{q,i,r},m_u,l}$
$v_{API,x,1,f_{q,i,r},m_u,l}$	компонента вектора API-функцій $v_{API,x,f_{q,i,r},m_u,l}$	$v_{API,x,f_{q,i,r},m_u,l} = \left(v_{API,x,1,f_{q,i,r},m_u,l}, v_{API,x,2,f_{q,i,r},m_u,l}, \dots, v_{API,x,n_{m_u},f_{q,i,r}m_u,,l}\right)$
$v_{API,x,f_{q,i,r},m_u,l}$	матриця	$V_4: v_{z,f_{q,i,r}} \times v_{API,x,f_{q,i,r},m_u,l} \dashrightarrow \{0;1\}$
$v_{API,x,i,j}$	вектор зворотніх реакцій	$V_5: v_{API,x,i} \times j \dashrightarrow v_{API,x,i,j}$

Продовження таблиці 4.2

1	2	3
$v_{API,f_{q,i,r},K}$	вектор кількості входження API-функцій	V_6: $v_{API,f_{q,i,r}} \dashrightarrow v_{API,f_{q,i,r},K}$
$v_{z,m_u,f_{q,i,r},K}$	компонента вектора $v_{API,f_{q,i,r},K}$	$v_{API,f_{q,i,r},K} = \left(v_{z,1,f_{q,i,r},K}, v_{z,2,f_{q,i,r},K}, \dots, v_{z,m_u,f_{q,i,r},K}, \dots, v_{z,n_{m_u},f_{q,i,r},K}\right)$

В результаті в двох матрицях представленими на рис. 4.5 та рис. 4.6 міститиметься інформація про можливі ключові API-функції, які використовуються для виконання зловмисних дій та кількості їх входжень.

		$v_{API,x,f_{q,i,r},m_u,l}$			
		$v_{API,x,1,f_{q,i,r},m_u,l}$	$v_{API,x,2,f_{q,i,r},m_u,l}$	…	$v_{API,x,n_{m_u},f_{q,i,r}m_u,,l}$
$v_{z,f_{q,i,r}}$	l	1	2	…	n_{m_u}
$v_{z,m_u,f_{q,i,r}}$	l_j			…	

Рис. 4.5. Матриця зв'язку однієї компоненти вектору зловмисних дій та l_j – го вектору API-функцій

		$v_{API,x,f_{q,i,r},m_u,l}$			
		$v_{API,x,1,f_{q,i,r},m_u,l}$	$v_{API,x,2,f_{q,i,r},m_u,l}$	…	$v_{API,x,n_{m_u},f_{q,i,r}m_u,,l}$
$v_{z,f_{q,i,r},K}$	l	1	2	…	n_{m_u}
$v_{z,m_u,f_{q,i,r},K}$	l_j			…	

Рис. 4.6. Матриця зв'язку однієї компоненти l_j – го вектору зловмисних дій та кількості входжень API-функцій

Позначення в табл. 4.2 і таблицях рисунків 4.6 та 4.7: m_u – номер

компоненти вектору зловмисних дій та атак $v_{z,f_{q,i,r}};\ u = 1, 2, \dots, N_{z,q,i,r};\ N_{z,q,i,r}$ – кількість компонент вектору $v_{z,f_{q,i,r}}$ для функції $f_{q,i,r},\ l_j$ – номер представлення m_u – ї компоненти вектору зловмисних дій для функції $f_{q,i,r};\ j$ – число, що вказує на номер варіанту представлення; $l_j = 1, 2, \dots, l_{m_{u,q,i,r}}; l_{m_{u,q,i,r}}$ – кількість варіантів представлення API функціями m_u – ї компоненти вектору зловмисних дій для функції $f_{q,i,r};\ x$ – номер операційної системи; $x = 1, 2, \dots, N_x; N_x$ – кількість різних операційних систем та їх версій; n_{m_u} – кількість API-функцій m_u – ї компоненти вектору зловмисних дій для функції $f_{q,i,r}$.

Визначимо кількість K_{v_z} таких векторів для зберігання відомостей про можливі зловмисні дії за формулою (4.8):

$$K_{v_z} = N_x \times \sum_{q=1}^{3} \sum_{i=1}^{i_q} \sum_{r=1}^{r_i} (N_{z,q,i,r} \times l_{m_{u,q,i,r}}), \tag{4.8}$$

Де i_1 – кількість елементів на 1 – у рівні бот-мережі; аналогічно i_2 – на другому; i_3 – на третьому; r_i – кількість функцій в i – у елементі; $N_{z,q,i,r}$ – кількість компонент вектору $v_{z,f_{q,i,r}}$ для функції $f_{q,i,r};\ l_{m_{u,q,i,r}}$ – кількість варіантів представлення API-функціями m_u – ї компоненти вектору зловмисних дій для функції $f_{q,i,r}; x$ – номер операційної системи; $x = 1, 2, \dots, N_x; N_x$ – кількість різних операційних систем та їх версій.

За формулою 4.6 визначається, також, кількість векторів $v_{z,f_{q,i,r},K}$ входжень API-функцій. Для розглядуваної архітектури бот-мережі представленої на рис. 4.1 і рис. 4.3 припустимо, що елементи вузлів бот-мережі, які відносяться до одного рівня, є однаковими і можуть відрізнятись тільки врахуванням версії операційної системи, тоді для формули 4.6 приймемо такі значення змінних $i_1 = 7,\ i_2 = 10,\ i_3 = 8$, а також вважатимемо, що кожна функція представлена вектором з однією компонентою і кількість представлень API-функціями для

кожної компоненти вектору єдине $l_{m_{u,q,i,r}} = 1$. В результаті отримаємо такі обчислення (формула (4.9)):

$$K_{v_z} = N_x \times \sum_{q=1}^{3} \sum_{i=1}^{i_q} \sum_{r=1}^{r_i} (N_{z,q,i,r} \times l_{m_{u,q,i,r}}) = N_x \times (\sum_{i=1}^{7} \sum_{r=1}^{r_i} N_{z,q,i,r} + \sum_{i=1}^{10} \sum_{r=1}^{r_i} N_{z,q,i,r} + \sum_{i=1}^{8} \sum_{r=1}^{r_i} N_{z,q,i,r}) = N_x \times (37 + 24 + 20) = N_x \times 81. \quad (4.9)$$

Таким чином, при початкових даних з спроектованого прототипу бот-мережі розрахунки за формулою (4.9) підтверджують кількість функцій. Формула (4.9) дозволяє визначати кількість функцій, якщо вони різні для елементів одного рівня та для різних елементів вузлів бот-мережі. Такі розрахунки необхідні для оцінки розмірів бази зловмисних дій та атак на етапі проектування розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення.

Розроблена еталонна модель бот-мережі на основі компонентів трьох рівнів, структурними компонентами яких є функції, є основою для представлення відомих бот-мереж, які були виявлені і певним чином класифіковані за характерними ознаками. Такі відомі виявлені бот-мережі використовують не всі закладені в еталонну модель бот-мережі функції і елементи. А також, їх структура відноситься до певних топологій і не є повністю такою як в еталонній моделі бот-мережі. Відомі бот-мережі побудовані з використанням тільки певних елементів і тому вся така бот-мережа на основі її елементів є підмережею еталонної. Еталонна модель бот-мережі та розподіл відомих бот-мереж за класами, які відрізняються між собою, є основою створення бази для виявлення нових бот-мереж за їх характерними ознаками і віднесення до одного з класів, щоб в подальшому їх можна було використовувати. Використання тільки еталонної моделі бот-мережі є недостатнім, бо частина характерних ознак бот-мереж може бути знівельована менш характерними. Наявність в базі зловмисного програмного забезпечення для порівняння інформації про відомі бот-мережі, в яких є менше компонентів,

ніж в еталонній моделі, підсилює важливість характеристичних ознак завдяки меншій кількості розглядуваних елементів.

Розроблені для компонентів еталонної моделі та моделей типових бот-мереж числові характеристики є важливими для розробки методу виявлення бот-мереж у локальних комп'ютерних мережах.

4.3. Метод виявлення бот-мереж у локальних комп'ютерних мережах

Особливістю використання розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення в локальних мережах є те, що з метою здійснення виявлення розроблені відомі методи орієнтовані саме на виявлення ЗПЗ в окремих КС чи корпоративних мережах [164, 223] за певним його типом чи типами, але при цьому не враховують при виявленні особливості антивірусних засобів, в яких вони реалізовані. РБС розроблена для здійснення тривалого спостереження за протіканням процесів в окремих КС та локальній мережі. Це надає можливість накопичувати інформацію про процеси, які протікають в конкретній КС, порівнювати їх між різними КС локальної мережі. Також, перевагою РБС над ЗПЗ є те, що РБС містить в своїх програмних модулях інформацію про кожну КС локальної мережі і використовує її для порівняння в процесі свого функціонування. Оскільки місцем дослідження виступають КС локальної мережі, де адміністратор мережі встановив визначене загальне і спеціалізоване програмне забезпечення, тоді РБС отримує перевагу над ЗПЗ, якому необхідно буде пробувати отримувати інформацію про нього. Користувачі в локальній мережі обмежені у користуванні КС, але вони теж можуть не санкціоновано встановлювати потрібне їм програмне забезпечення, відвідувати веб-сайти із зловмисним програмним забезпеченням. Враховуючи, що вузли бот-мережі в КС можуть не проявлятись швидко, перебуваючи в стані очікування і збору потрібної інформації або тримаючи під контролем КС не здійснювати інших дій, то їх виявлення на основі відслідковування можливих зловмисних дій потребує не тільки тривалого спостереження, що є важливим, але і відповідних

методів виявлення з метою локалізації та структури засобів виявлення, які б надавали суттєві переваги при виявленні. Методи виявлення ЗПЗ повинні враховувати особливості засобів, в яких вони будуть реалізовані. Їх реалізація без отримання переваги за рахунок структури засобів не зможе надати їм переваги особливо тоді, коли такі засоби будуть встановлюватись в КС, в якій вже присутнє ЗПЗ.

При розгляді сукупності КС в локальній мережі припустимо, що частина з них є вузлами бот-мережі, а решта не є частиною бот-мережі. При цьому можуть бути варіанти, при яких КС можуть містити файлове ЗПЗ, яке не відноситься до бот-мереж. А також, з КС може вестись атака на іншу КС або здійснюватись зловмисні дії в КС. Якщо КС містить вузол бот-мережі чи файлове ЗПЗ, то на КС може здійснюватись атака ззовні з вузлів іншої бот-мережі чи проводитись іншим мережним ЗПЗ. Врахуємо також, що при відсутності мережного ЗПЗ в КС атака з цієї КС не відбувається. Крім того, вважатимемо, що усі розглядувані КС містять ПМ РБС, тобто в них здійснюється моніторинг подій і приймається рішення про виявлення і локалізацію мережного ЗПЗ, в тому числі вузлів бот-мереж. Виділимо ці 12 варіантів в табл. 4.3 і згруповано в залежності від наявності бот-мереж в КС:

1) вузол бот-мережі в КС здійснює зловмисні дії тільки в межах КС і атака з КС на інші КС не проводиться, але здійснюється або не здійснюється атака на цю ж розглядувану КС іншим мережним ЗПЗ, при цьому інше ЗПЗ в КС може бути присутнім або не бути;

2) те ж, що і у варіанті 1, тільки з КС здійснюється атака вузлом бот-мережі;

3) вузла бот-мережі в КС немає, наявне файлове ЗПЗ, атака на КС ззовні здійснюється або не здійснюється;

4) ЗПЗ немає, атака ззовні на КС здійснюється або не здійснюється.

Події, які не представлені в табл. 4.3 і їх при повній групі подій є ще чотири, не можуть відбуватись. Зокрема, якщо немає ЗПЗ в КС, тоді з неї не може вестись атака на інші КС.

Таблиця 4.3

Варіанти подій в КС

№ варіанта подій	Вузол бот-мережі в КС	Файлове ЗПЗ в КС	Атака з КС	Атака на КС
1	+	-	+	+
2	+	-	+	-
3	+	-	-	+
4	+	-	-	-
5	+	+	+	+
6	+	+	+	-
7	+	+	-	+
8	+	+	-	-
9	-	+	-	+
10	-	+	-	-
11	-	-	-	+
12	-	-	-	-

Варіанти подій з табл. 4.3 є основою для розробки методу виявлення бот-мереж у локальних комп'ютерних мережах в частині відображення в ньому особливостей різного стану КС, в якому вона може перебувати.

Для здійснення обробки подій в КС програмним модулем РБС необхідно здійснити формалізоване представлення розроблених функцій бот-мереж через поведінкові сигнатури на основі API-функцій. Підготовка таких еталонних моделей полягає у формуванні знань на основі функцій, які вказують на наявність бот-мереж. З цією метою потрібно виконати такі основні кроки:

1. Здійснити для кожної функції $f_{q,i,r}$ еталонної моделі бот-мережі її реалізацію в різних операційних системах.

2. Для кожної функції $f_{q,i,r}$ задати вектор зловмисних дій та атак $v_{z,f_{q,i,r}}$, виділивши його компоненти. Узагальнити компоненти різних векторів, які відповідають однаковим функціоналам.

3. Здійснити трасування складових функцій $f_{q,i,r}$ для отримання їх

представлення через API-функції для кожної операційної системи чи її актуальної версії.

4. Пронумерувати всі отримані задіяні в представленні API-функції або всі API функції згідно переліку для кожної операційної системи.

5. Здійснити нумерування API-функцій для кожної компоненти вектору зловмисних дій та атак еталонної моделі бот-мережі для підготовки бітової матриці $v_{z,f_{q,i,r}} \times v_{API,x,f_{q,i,r},m_u,l}$ з врахуванням різних представлень компонент вектору зловмисних дій та атак.

6. Сформувати вектори для функцій ознак f за номерами API-функцій, які їх формують, для трьох типів елементів бот-мережі.

7. Промаркувати критичні API-функції у векторах $v_{API,x,f_{q,i,r},m_u,l}$.

8. Побудувати вектори кількості входжень API функцій $v_{z,f_{q,i,r},K}$ у векторах зловмисних дій та атак $v_{z,f_{q,i,r}}$.

9. Здійснити побудову бітової матриці $v_{z,f_{q,i,r}} \times v_{z,f_{q,i,r},K}$ з врахуванням різних представлень компонент векторів зловмисних дій та атак. Кожна компонента кожного вектору кількості входжень API функцій є числом, що визначає кількість разів входження відповідної API-функції, яка входить в певний вектор.

10. Побудувати вектор зворотніх дій $v_{API,x,i,j}$ для кожної функції $f_{q,i,r}$.

11. Задати бітову матрицю зворотніх дій на основі відношення V_4: $v_{z,f_{q,i,r}} \times v_{API,x,f_{q,i,r},m_u,l} \dashrightarrow \{0; 1\}$.

12. Визначити кількість компонент векторів для кожної з функцій $f_{q,i,r}$ і занести в кожен вектор додатковою останньою компонентою.

Отримані результати за цими кроками заносимо до бази зловмисних дій та атак і визначаємо цю інформацію, як таку що характеризує клас «0». Для відомих типів бот-мереж, які сформовано на основі їх виявлення, проводимо виконання кроків 1-12 для кожного типу і відносимо отриману інформацію в базу зловмисних дій та атак відповідно за класами «1»-«6». Для тих функцій типів

бот-мереж «1»-«6», які не є у відкритому доступі, використовуємо функції-аналоги з еталонної моделі бот-мережі. Збільшення кількості класів є необхідним для розширення можливості самонавчання класифікаторів ПМ РБС.

Отримавши вектори, в яких представлені числові величини в якості їх компонент, що дозволяє на основі них перейти до здійснення класифікації нових виконуваних об’єктів в КС, потрібно організувати збір інформації про виконувані процеси в КС на основі API-функцій та розробити метод їх обробки для перевірки віднесення до одного з класів бот-мереж.

В КС локальної мережі засобами РБС досліджуватимуться такі події:

1) виконання команд в КС, зокрема які вважаються підозрілими і можуть бути віднесенні до команд ЗПЗ;

2) виконання команд, з яких є такі, що здійснюють надсилання пакетів з КС;

3) обробка отриманих пакетів і команд, які надійшли ззовні.

Важливим, також, є дослідження відправлених пакетів або виконання мережних команд. Пакети або команди можуть бути надіслані КС, які перебувають в досліджуваному сегменті мережі або за його межами. Це впливає на встановлення достовірності виявлення. Якщо КС перебувають в локальній мережі і в них встановлено програмні модулі РБС, тоді отримані пакети і команди можуть бути досліджені і з тієї КС, з якої вони надіслані, і в тій КС куди вони надійшли.

Таким чином, стадія моніторингу КС передбачає:

1) збір інформації про активні процеси в КС;

2) збір інформації про мережну активність та трафік (вхідний, вихідний).

Для отримання API-функцій ПМ РБС використовують стандартні утиліти і налаштовують їх відповідно до структури векторів API-функцій. При цьому використовується їх довжина, яка представляється відповідною компонентою векторів, а також можуть використовуватись тільки марковані критичні API-функції.

Відомі монітори викликів API-функцій надають інформацію про

ідентифікатори ниток, назви DLL, які створили нитки. А також синтаксис API-виклику зі всіма параметрами і повернутими значеннями. Якщо виклик відбувся невдало, то теж надають інформацію про нього. Такі монітори містять більше 10000 API функцій з 166 DLL бібліотек, а також більше 700 методів з більше, ніж 600 COM-інтерфейсів, включаючи Shell, Browser, DirectShow, DirectSound, DirectX і інші. Монітори API-функцій здійснюють доступ до вхідного і вихідного буферів, а також будують дерево викликів ієрархії API-викликів. Також, підтримуються різні операційні системи та їх версії. Таким чином, вважатимемо практичну реалізацію монітору API-викликів наявною і такою, що може бути використаною в програмних модулях РБС. Це важлива компонента, яка дозволяє отримувати результат моніторингу API-викликів в динамічному режимі, тобто проводити необхідний для реалізації РБС збір інформації про виконувані процеси. Інформація про ймовірно зловмисні події заноситься до вектору ймовірно зловмисних подій, який в подальшому використовується для здійснення переведення його компонент в чисельні ознаки.

Здійснення моніторингу API-викликів [229, 389] в динамічному режимі проводиться програмним модулем РБС в кожній КС його компонентою-утилітою і виконується так:

1) здійснити моніторинг API-викликів, які динамічно виконуються в КС;

2) початок послідовного збору розпочинати з моніторингу першої API-функції, що відноситься до групи критичних, які можуть бути виконані зловмисним програмним забезпеченням, і занесенням їх до вектору ймовірно підозрілих дій для зафіксованого процесу;

3) завершити збір API-функції, віднесених до певного процесу, після його повного завершення або до його зупинки;

4) збір API-функцій для певного пробудженого процесу продовжувати до його завершення чи наступної призупинки;

5) для прописаного в ПМ програмного забезпечення, завантаження якого здійснюється після запуску КС і такий порядок визначено адміністратором, виконувати з першої команди;

6) надіслати на перевірку тільки після того, коли кількість зібраних API-функцій є не меншою, ніж мінімальна кількість компонентів у векторах зловмисних дій та атак бази зловмисних дій та атак;

7) після отримання вектору з допустимою кількістю зібраних API-функцій надсилати на перевірку зібрану частину і продовжувати збір далі до завершення процесу, який породжує API-виклики.

Для отримання чисельної ознаки компонент вектору ймовірно зловмисних подій здійснимо визначення відсоткової відповідності вектору ймовірно зловмисних подій до кожного вектору кожного класу з бази векторів зловмисних дій та атак такими способами:

1) встановлення відсотку максимальної подібності з врахуванням кількості їх компонент з кожним вектором зловмисних дій та атак і занесення цієї інформації в характеристичну матрицю 1 (рис. 4.5);

2) встановлення відсотку входження API-функцій до кількості у відповідних векторах і занесення цієї інформації в характеристичну матрицю 1 (рис. 4.5);

3) встановлення відсотку максимальної подібності з врахуванням кількості їх компонент з кожним вектором зворотніх зловмисних дій та атак і занесення цієї інформації в характеристичну матрицю 2 (рис. 4.6).

Для віднесення отриманого вектору ймовірно зловмисних дій до ЗПЗ або неінфікованого здійснимо класифікацію за сімома введеними класами типів бот-мереж та восьмим класом, який відповідатиме неінфікованому програмному забезпеченню. Для здійснення класифікації відомо багато методів. Враховуючи показники швидкості, які необхідно забезпечити для отримання результатів ПМ РБС виберемо наївний баєсівський класифікатор. Як і інші методи класифікації, він має ряд переваг та недоліків. Але для проведення класифікації вектору ймовірно зловмисних подій, частини визнаних недоліків у вибраному класифікаторі не має. Зокрема, збір та зберігання статистичної інформації для застосування теореми ускладнено, але в розглядуваній задачі така інформація вже підготовлена. Іншим важливим недоліком при використанні цього методу

вважають припущення, що множини ознак обов'язково не повинні перетинатись. Але для формалізованої бот-мережі через функції, які представлені векторами зловмисних дій та атак, такої проблеми не має, оскільки всі вектори є незалежними між собою і буде здійснюватись пошук на відповідність до одного з векторів. Кожна функція бот-мережі має свій набір параметрів, якими вона представляється. Декілька функцій можуть мати подібні набори API-функцій, але перевірка буде здійснюватись в два етапи: спочатку перевірка стосовно до кожного вектору, потім перевірка стосовно до кожного класу в цілому. І отримання подібних результатів стосовно векторів було уточнено за рахунок відповідних розрахунків для кожного класу в цілому. Крім того, використання наївного баєсівського класифікатору дозволятиме отримати ймовірності, які будуть використані для оцінювання ймовірності бути ураженим всього ПМ, в форматі тих же числових величин, які отримуватимуться на інших етапах розрахунків.

На основі отриманої чисельної інформації з двох типів характеристичних матриць здійснюємо визначення віднесення отриманих даних моніторингу до певного класу на основі наївного баєсівського класифікатору.

Введемо позначення для вектору ймовірно зловмисних дій так: $v_{p,e_s} = \left(v_{p,e_s,1}, v_{p,e_s,2}, \dots, v_{p,e_s,n_{v_p}}\right)$, де p – позначення ймовірно зловмисної дії, e_s – номер вектору, e – число, що відповідає номеру вектору, який сформовано з певної кількості або всіх API-функцій досліджуваного процесу, s – номер варіанту послідовності API-функцій досліджуваного процесу вектору для e процесу, n_{v_p} – кількість компонент вектору v_{p,e_s}. Компонентами вектору v_{p,e_s} є номери API-функцій, які беруться із загальної нумерації для всіх таких функцій і функцій для бот-мереж. Введемо, також, вектор $v_{p,K,e_s} = \left(v_{p,K,e_s,1}, v_{p,K,e_s,2}, \dots, v_{p,K,e_s,n_{v_p}}\right)$, який відображатиме кількість входжень кожної API-функції у векторі v_{p,e_s}. Згідно класифікатору необхідно встановити належність вектору v_{p,e_s} до одного з класів бот-мереж. Нехай $V_p = \left\{v_{p,1}, v_{p,2}, \dots, v_{p,n_{V_p}}\right\}$ – вибірка сформована на

основі значень API-функцій для векторів типу v_{p,e_s}, A – гіпотеза про належність значень V_p до одного з класів K_l бот-мереж, де $l = 0, 1, \dots, 6$. Для вирішення задачі класифікації необхідно визначити ймовірність того, що вибірка V_p належить класу K_l, враховуючи знання про опис атрибутів V_p, тобто потрібно визначити $P(\mathrm{A} \mid V_p)$ ймовірності того, що гіпотеза A містить данні з вибірки V_p. Апостеріорна ймовірність $P(\mathrm{A} \mid V_p)$ – це ймовірність того, значення A залежить від певних атрибутів вибірки V_p і визначимо її за теоремою Баєса згідно формули (4.10):

$$P(\mathrm{A} \mid V_p) = \frac{P(V_p \mid \mathrm{A})\, P(\mathrm{A})}{P(V_p)}, \tag{4.10}$$

де $P(V_p \mid \mathrm{A})$ - ймовірність присутності V_p при істинності гіпотези A, тобто що є належність до класу бот-мереж і яка ймовірність присутності при цьому значень з V_p; $P(\mathrm{A})$ – апріорна ймовірність того що значення V_p належатимуть до одного з класів K_l бот-мереж і не залежить від значень з V_p; $P(V_p)$ - ймовірність належності класу, тобто настання причини.

Здійснимо побудову наївного баєсівського класифікатора для визначення належності певному класу бот-мереж. Класи бот-мереж K_l визначені і представлені сукупністю пар двох векторів $v_{z,f_{q,i,r}}$, $v_{API,x,f_{q,i,r},m_u,l}$. Вибірка V_p належить до класу K_l з найвищою апостеріорною ймовірністю тоді і тільки тоді, коли виконується умова (формула (4.11)):

$$P(K_{l_1} \mid V_p) > P(K_{l_2} \mid V_p) \tag{4.11}$$

для всіх l_1 та l_2 таких, що $0 \leq l_1 \leq 6,\ 0 \leq l_2 \leq 6,\ l_1 \neq l_2$.

Таким чином, здійснюємо пошук класу, який максимізує ймовірність $P(K_l \mid V_p)$, тоді такий клас $P(K_l \mid V_p)$ буде класом з максимальною апостеріорною гіпотезою. Ймовірність $P(V_p)$ буде однаковою в кожному з підкласів для всіх

класів. Максимізації підлягає тільки чисельник в формулі Баєса: $P(V_p \mid K_l)P(K_l)$. В ньому ймовірність $P(K_l)$ оцінюємо за кількістю входження кожної з API функцій до підкласів, визначених функціями бот-мереж. Ця кількість відображена у векторах $v_{API,x,f_{q,i,r},m_u,l}$. Ймовірності $P(V_p \mid K_l)$ представлені для кожного входження API-функцій до кожного підкласу оцінюємо за базовою вибіркою для атрибуту v_{p,e_s} API-функцій з вибірки K_l. Оскільки, значення атрибутів для API-функцій є дискретною величиною, яка показує кількість зразків класу K_l в множині значень кожного класу, що мають значення для атрибуту v_{p,e_s} API-функцій розділене на частоту кількості K_l – тих зразків класу в сукупному представленні типів бот-мереж. Для уникнення проблеми «нульової частоти», пов'язаною з можливою наявністю API-функції, якої немає серед API-функцій, використовуваних в базових класах бот-мереж, здійснимо лапласову корекцію, оскільки кількість компонентів векторів зловмисних дій та атак є достатньо великою.

Розрахунок проміжних величин для обчислення за формулою Баєса представлено в табл. 4.5.

Таблиця 4.5

Розрахунок проміжних величин

Клас, y	Підклас	Вектори API функцій			Разом:
		1	…	n_{m_u}	
1	2	3	4	5	6
0	1	$v_{API,x,1,f_{q,i,r},1,l}$	…	$v_{API,x,n_{m_{u,0}},f_{q,i,r},1,l}$	$l_{1,0}$
	…	…	…	…	…
	$m_{u,0}$	$v_{API,x,1,f_{q,i,r},m_{u,0},l}$	…	$v_{API,x,n_{m_{u,0}},f_{q,i,r},m_{u,0},l}$	$l_{m_{u,0}}$
	Разом:	$\sum_{g=1}^{m_{u,0}} v_{API,x,1,f_{q,i,r},g,l}$	…	$\sum_{g=1}^{m_{u,0}} v_{API,x,n_{m_{u,0}},f_{q,i,r},g,l}$	$\sum_{g=1}^{m_{u,0}} l_{g,0}$
…	…	…	…	…	

Продовження таблиці 4.5

1	2	3	4	5	6
6	1	$v_{API,x,1,f_{q,i,r},1,l}$	…	$v_{API,x,n_{m_{u,6}},f_{q,i,r},1,l}$	$l_{1,6}$
	…	…	…	…	…
	$m_{u,6}$	$v_{API,x,1,f_{q,i,r},m_{u,6},l}$	…	$v_{API,x,n_{m_{u,6}},f_{q,i,r}m_{u,6},l}$	$l_{m_{u,6}}$
	Разом:	$\sum_{g=1}^{m_{u,6}} v_{API,x,1,f_{q,i,r},g,l}$	…	$\sum_{g=1}^{m_{u,6}} v_{API,x,n_{m_{u,6}},f_{q,i,r},g,l}$	$\sum_{g=1}^{m_{u,6}} l_{g,6}$
Разом:		$\sum_{y=0}^{6}\sum_{g=1}^{m_{u,i}} v_{API,x,1,f_{q,i,r},g,l}$		$\sum_{y=0}^{6}\sum_{g=1}^{m_{u,i}} v_{API,x,1,f_{q,i,r},g,l}$	$\sum_{y=0}^{6}\sum_{g=1}^{m_{u,i}} l_{g,i}$

Числові величини $l_{j,y}$ – кількість варіантів представлення API-функціями для функції $f_{q,i,r}$ j-того підкласу y – класу. Ними представлено кількість елементів в кожному з підкласів.

Суми показників по рядках і стовпцях співвідносяться за формулою (4.12):

$$\sum_{y=0}^{6}\sum_{g=1}^{m_{u,i}} l_{g,i} \leq \sum_{y=0}^{6}\sum_{g=1}^{m_{u,i}} v_{API,x,1,f_{q,i,r},g,l} + \cdots + \sum_{y=0}^{6}\sum_{g=1}^{m_{u,i}} v_{API,x,1,f_{q,i,r},g,l}. \quad (4.12)$$

Для $y = 0$, тобто для класу 0, і підкласу $m_{u,0} = 1$ різні представлення через API функції компоненти $v_{z,1,f_{q,i,r}}$ вектору $v_{z,f_{q,i,r}}$, отримуємо ймовірності $P(V_p \mid \mathrm{A})$ за формулами (4.13):

$$P\left(v_{z,1,f_{q,i,r}} \mid K_0\right) = \frac{v_{API,x,1,f_{q,i,r},1,l}}{l_{1,0}},$$

$$P\left(v_{z,1,f_{q,i,r}} \mid K_0\right) = \frac{v_{API,x,2,f_{q,i,r},1,l}}{l_{1,0}}, \quad (4.13)$$

…

$$P\left(v_{z,1,f_{q,i,r}} \mid K_0\right) = \frac{v_{API,x,n_{m_{u,0}},f_{q,i,r},1,l}}{l_{1,0}}.$$

Узагальнимо їх для всіх класів та їх підкласів і векторів зловмисних дій та атак для здійснення обчислення ймовірності належності класу при проведенні

класифікації за формулою (4.14):

$$P\left(v_{z,m_{u,y},f_{q,i,r}} \mid K_y\right) = \frac{v_{API,x,n_{m_{u,y}},f_{q,i,r},m_{u,y},l}}{l_{m_{u,y}}}. \quad (4.14)$$

Таким чином, отримуємо за формулою (4.15):

$$\prod_{j=1}^{m_{u,y}} P\left(v_{z,j,f_{q,i,r}} \mid K_y\right) = \prod_{j=1}^{m_{u,y}} \frac{v_{API,x,n_{m_{u,y}},f_{q,i,r},m_{u,y},l}}{l_{m_{u,y}}}. \quad (4.15)$$

Далі обчислюємо ймовірність для підкласу класу за узагальненою формулою (4.16):

$$P(K_y) = \frac{l_{m_{u,y}}}{\sum_{g=1}^{m_{u,y}} l_{g,y}}. \quad (4.16)$$

Ймовірність для компоненти вектору ймовірно зловмисних подій обчислюємо за узагальненою формулою (4.17):

$$P(v_{p,e_s,g}) = \frac{v_{p,K,e_s,g}+1}{n_{v_p}+1}. \quad (4.17)$$

Загальна формула (4.18) для здійснення пошуку в підкласах класу y така:

$$P(K_y \mid v_{p,e_s}) = \frac{\prod_{j=1}^{m_{u,y}} \frac{v_{API,x,n_{m_{u,y}},f_{q,i,r},m_{u,y},l}}{l_{m_{u,y}}} * \frac{l_{m_{u,y}}}{\sum_{g=1}^{m_{u,y}} l_{g,y}}}{\prod_{g=1}^{n_{v_p}} \frac{v_{p,K,e_s,g}+1}{n_{v_p}+1}}. \quad (4.18)$$

З обчислених значень $P(K_y \mid v_{p,e_s})$ для вектору v_{p,e_s} вибираємо максимальне і здійснюємо віднесення цього вектору до підкласу певного класу, якщо отримане значення більше порогового значення встановленого для цього класу. Якщо порогові значення отриманих ймовірностей не відповідають заданим для підкласів, тоді вектор v_{p,e_s} не відноситься до жодного із заданих класів. В цьому випадку розглядаємо варіант продовження збору і обробки даних

про API-функції або припиняємо процес його дослідження. Враховуючи побудову підкласів і класів, яка допускала можливість подібності деяких підкласів різних класів між собою, здійснимо уточнення класифікації для вектору v_{p,e_s}, особливо якщо відбувся збіг відповідних значень в різних підкласах. Тому, проводимо розрахунки за цією ж методикою тільки по відношенню до класів в цілому віднесення цього вектору v_{p,e_s} за формулою (4.19):

$$P\left(K_y \middle| v_{p,e_s}\right) = \frac{\prod_{y=0}^{6} \frac{\sum_{g=1}^{m_{u,y}} v_{API,x,y,f_{q,i,r},g,l}}{\sum_{g=1}^{m_{u,y}} l_{g,y}} \cdot \frac{\sum_{g=1}^{m_{u,y}} l_{g,y}}{\sum_{y=0}^{6}\sum_{g=1}^{m_{u,y}} l_{g,y}}}{\prod_{g=1}^{n_{v_p}} \frac{v_{p,K,e_s,g}+1}{n_{v_p}+1}}. \quad (4.19)$$

В результаті отримуємо другу групу ймовірностей, що відноситься до класів, та на основі неї здійснюємо віднесення до певного підкласу визначеного при обчисленні класу.

Оскільки, отримані набори векторів базуються на кількості входжень API-функцій, які можуть повторюватись в них. Тому, було здійснено перехід від початкових векторів зловмисних дій та атак до векторів, що відображають у своїх компонентах кількості входжень API-функцій. Така модель підготовки даних відповідає мультиноміальній моделі побудови характеристики послідовності подій, які полягають у випадковому виборі однієї API-функції з скінченої множини функцій. Для віднесення документу до певного класу здійснюють знаходження добутку ймовірностей тих API-функцій, які увійшли у вектор ймовірно підозрілих дій. Таким чином, використана мультиноміальна генеративна модель враховує кількість повторень API-функцій і не враховує відсутність певних API-функцій. Але для великої кількості компонент вектору інформація про відсутні функції, які є в класі і яких немає у векторі, враховується неявним чином, бо для API-функції, які мають більше число входжень, матимуть більшу ймовірність для певного класу. Відповідно, якщо деякі з API-функцій не зустрілись у досліджуваному векторі, але вони наявні в певному класі в

невеликій кількості, то мають невелику ймовірність входження для певного класу і відповідно не впливатимуть суттєво на визначення приналежності класу.

Визначення належності вектору v_{p,e_s} до класу K_y чи його підкласу здійснюємо на основі обчислень ймовірностей для кожного класу чи підкласу за формулою (4.20):

$$P\left(v_{p,e_s} \mid K_{y,g}\right) = P\left(|n_{v_p}|\right) n_{v_p}! \prod_{w=1}^{n_{m_u}} \left(\frac{1}{v_{p,K,e_S,w}!} \left(\frac{v_{API,x,w,f_{q,i,r},g,l}}{\sum_{g=1}^{m_{u,0}} v_{API,x,w,f_{q,i,r},g,l}} \right)^{v_{p,K,e_S,w}} \right). \quad (4.20)$$

Для проведення навчання необхідно навчити ймовірності $P\left(v_{z,z_v,f_{q,i,r}} \mid K_{y,g}\right)$. Для цього здійснюють оптимальні оцінки ймовірностей того, що певна API-функція зустрінеться в кожному класі чи підкласі згладивши результат за схемою Лапласа за формулою (4.21):

$$P\left(v_{z,d,f_{q,i,r}} \mid K_{y,g}\right) = \frac{1 + \sum_{b=1}^{\sum_{g=1}^{m_{u,y}} l_{g,y}} v_{API,x,b,f_{q,i,r},g,l} \, P\left(K_{y,g} \mid v_{z,b,f_{q,i,r}}\right)}{n_{m_u} + \sum_{d=1}^{n_{m_u}} \sum_{b=1}^{\sum_{g=1}^{m_{u,y}} l_{g,y}} v_{API,x,d,f_{q,i,r},g,l} \, P\left(K_{y,g} \mid v_{z,b,f_{q,i,r}}\right)}. \quad (4.21)$$

Здійснення навчання наївного баєсівського класифікатора проводимо наступним чином:

1) задати підкласи по одному представленню API-функціями для кожного з них та розрахувати ймовірності для кожної API-функції, підкласів та класів і зафіксувати їх як первинні;

2) для кожної відомої наступної варіації компоненти функції її представлення API-функціями здійснити класифікацію за баєсівським класифікатором для класів і підкласів; якщо отримане представлення віднесено вірно до вказаного підкласу, тоді додати його марковані елементи до тих, що вже є у підкласі, як окремий зразок; якщо отриманий результат не відносить його до потрібного підкласу, тоді віднести його до його підкласу, але при цьому зробити порівняння із іншими значеннями класів; результатом порівняння буде

відхилення, тобто різниця, ймовірностей від початкових; якщо результуюча ймовірність суттєво (більше 10 %) відрізняється від первинної ймовірності підкласу чи класу, тоді створити окремий підклас цього підкласу класу; зафіксувати для кожного розрахунку кроку навчання отримані ймовірності для кожної API-функції, підкласів та класів; для тих підкласів і класів, де розбіжність з первинними становить більше 10 %, здійснити створення нового підкласу в його підкласі; отримані за декілька ітерацій всі ймовірності усереднити та зафіксувати як відповідні ймовірності для використання в подальших розрахунках;

3) після завершення основного етапу навчання та доповнення баєсівського класифікатора новими даними здійснити перевірку відхилень для додатково сформованих підкласів певних підкласів і встановити розбіжність між їх середніми значеннями ймовірностей; якщо розбіжність становить менше 10 %, тоді доповнити підклас даними додаткового підкласу, його вилучити та перерахувати всі ймовірності і їх середні значення;

4) визначити різниці середніх ймовірностей за кроками навчання та ймовірностей отриманих розрахунком класифікатора; якщо для певних підкласів різниці більше 10 %, тоді продовжувати для них навчання додаванням додаткових даних і повторенням кроків 1-3.

Для здійснення глибшої перевірки на віднесення до певного підкласу вектору зловмисних дій та атак $v_{z,f_{q,i,r}}$ здійснимо на основі його значень та значень вектора ймовірно зловмисних дій підготовчі кроки з кодування послідовностей API-функцій в них: для першого аналізу беремо кожні дві, потім для другого аналізу - кожні три. Аналогічно будуємо для таких кодувань класифікатор на основі мультиноміальної генеративної моделі та здійснюємо його навчання.

Вектор v_{p,e_s} може бути не віднесений до жодного класу і підкласу із заданих, тобто дослідження його компонент встановило, що серед них немає ймовірно зловмисних дій. Але встановлення такого факту базується на

застосуванні не тільки пошуку максимального значення ймовірності, обчисленої за теоремою Баєса, але і відповідність цієї ймовірності пороговим значенням класів і підкласів, визначення яких здійснюють в процесі навчання класифікатора. Це пов'язано з тим, що виконуваний процес, який відображено вектором v_{p,e_s}, може не відноситись до зловмисного програмного забезпечення, тобто належати класу потрібного неінфікованого ПЗ. В такому випадку виконуваний процес ПМ РБС далі не розглядається.

Враховуючи велику кількість API-функцій в системах, що впливає на оперативність отримання результату, можна здійснити їх групування за різним призначенням і відповідно покращити їх розрізнення за функціональним спрямуванням. Також, невеликий програмний код кожного підкласу, завдяки декомпозиції бот-мережі на невеликі функціональні компоненти, та оптимальна кількість викликів API-функцій дозволяє здійснити класифікацію на початкових етапах, коли в класах перебуває небагато елементів, без залучення перевірки на встановлену послідовність функцій. Із збільшенням кількості елементів в класах, що досягається за рахунок їх доповнення еталонними зразками в процесі навчання та за рахунок здійснення класифікації безпосередньо, необхідність здійснення перевірки на другому рівні стає обов'язковою.

Отримані результати фіксуємо для використання в загальній формулі ймовірнісної оцінки стану ПМ, який передається всім решті ПМ з метою визначення стану РБС в цілому.

Здійснення самонавчання проводимо за схемою навчання (кроки 1-4). При виконанні аналізу вектору ймовірно зловмисних дій заносимо почергово його данні до кожного класу та здійснюємо розрахунки. Якщо відхилення отриманих ймовірностей перебувають в межах порогових значень для одного з підкласів, тоді після завершення класифікації включаємо його нові данні до його підкласу та робимо новий розрахунок для всього класифікатора і його середніх значень відхилень ймовірностей.

Здійснимо графічне відображення отриманих результатів на двовимірну площину. Для цього необхідно виконати такі кроки:

1) побудуємо на додатній осі абсцис відрізки, що мають унормовані довжини за кількістю API-функцій, які задано в кожному класі;

2) кожен відрізок розглядатимемо як сторону квадрату і побудуємо в першій чверті відповідні сім квадратів; на стороні кожного квадрату, яка лежить на осі абсцис, побудуємо квадрати для підкласів;

3) центром кожного класу чи підкласу вважатимемо середину відповідного йому відрізка на осі абсцис;

4) координати нового об’єкту визначаємо за усередненою формулою для абсцис і ординат та будуємо нові точки для кожного класу навпроти кожної точки класу на прямих, що проходять через ці точки і є паралельними до осі ординат; значення ординати (в кожному підкласі та класі) дорівнює усередненому значенню, отриманому класифікатором;

5) віддаленість від точок, якими визначатиметься клас, вказуватиме на відношення до певного класу;

6) якщо РБС встановить, що з’явився новий тип бот-мереж, і точки для семи класів будуть віддалені від центрів на відстань, що характеризуватиме дуже слабку ймовірність входження до класу, то буде задано новий клас і побудовано наступний квадрат з центром на його стороні.

При додаванні нового елементу до класифікатора [385] ПМ в одній з КС після цього здійснюється розсилання всім іншим ПМ РБМ інформація про це і сам контейнер з новим елементом для доповнення класифікатора. Таким чином, знання отримані одним ПМ передаються іншим компонентам системи для використання.

Рішення про місце здійснення обробки сформованого вектора ймовірно зловмисних дій визначає ПМ, в якому було зібрано ці данні. Якщо аналіз завантаження основних ресурсів показує велику завантаженість, тоді він направляє запит іншому ПМ на можливість обробки. Після отримання підтвердження, ПМ КС надсилає відповідний контейнер з вектором. Результати обробки в цьому випадку надходять тому ПМ, який ставив задачу обробки, в тому випадку, якщо ймовірність наявності зловмисного коду не встановлена.

Якщо ж класифікатор програмного модуля обробника встановив наявність зловмисного коду, тоді він додає до свого класифікатора нові данні, розсилає відповідний контейнер з новими знаннями (інформація про новий елемент для класифікатору, данні про виконуваний процес, на основі якого сформовано такий вектор) решті ПМ та повідомлення тому ПМ про наявність в КС процесу, який виконує зловмисні команди.

Таким чином, завдяки використанню РБС між її ПМ здійснюється обмін інформацією про зловмисні процеси в КС для додаткової перевірки та динамічно залучаються обчислювальні ресурси інших КС для виконання певних завдань інших компонентів системи. Це дозволяє оперативно виконувати задачі виявлення зловмисного програмного забезпечення. РБС надає можливість аналізувати додатково відмічені процеси та здійснювати виявлення ЗПЗ в тих КС локальної мережі, в яких воно не було виявлено в процесі дослідження або було пропущено, за рахунок інформації про зловмисний процес з іншої КС.

Крім того, розширення можливостей програмних модулів РБС по виявленню ЗПЗ на основі використання технологій обробки знань для здійснення самонавчання та організація їх взаємодії дозволяє реалізувати дві мети, які виконуються ними самостійно: поповнюють свої підкласи і класи новими елементами; створюють нові класи розширюючи класифікатор.

Проаналізуємо можливі варіанти подій в КС за присутності в ньому ПМ РБС та ЗПЗ. На основі їх дослідження визначимо стратегії поведінки ПМ в процесі появи таких подій.

Розглянемо випадок, коли в КС вже створено вузол бот-мережі, тобто завантажено зловмисне програмне забезпечення вузла бот-мережі, отримано контроль над певним програмним забезпеченням КС. Якщо вузол бот-мережі міститься в КС, тоді при її запуску стартове зловмисне навантаження повинно проявити себе в одному з процесів, які створюються прописаними виконуваними програмами в файлах автозапуску. Інакше, воно не зможе активуватись в КС при кожному її ввімкнені і, як наслідок, буде вилучено з часом з бот-мережі. Або воно очікуватиме запуску користувачем певних програм чи програми, що теж

може відбутись або не відбутись. Тому, висуваємо гіпотезу, що запуск програмного забезпечення вузла бот-мережі здійснюється після ввімкнення КС. В процесі отримання контролю над КС бот-мережа чи на перших етапах свого функціонування вузол бот-мережі здійснить призупинення роботи відомих антивірусних засобів та перейде в режим імітації їх роботи.

Варіантів встановлення ПМ РБС може бути два: 1) під час нового встановлення програмного забезпечення КС встановлюється програмне забезпечення ПМ; 2) програмне забезпечення ПМ встановлюється у вже функціонуючу КС. При першому варіанті КС може бути новою і тому потребує встановлення ПЗ або вже використовуваною раніше і потребує повного переустановлення ПЗ. Але навіть, якщо вона є новою, то як правило містить операційну систему і до включення її до складу локальної мережі вже міститиме певне визначене ПЗ. Таким чином, вірогідність отримати зловмисне програмне забезпечення при першому варіанті дуже мала, але теоретично можлива. При другому варіанті ймовірність наявності ЗПЗ у КС більша, ніж в першому.

Якщо припустити, що ЗПЗ у першому варіанті не було, тоді ПМ РБС встановлюється першою і отримує доступ для контролю над всією КС. Цей контроль передбачає основні такі дії: запуск першою, звірку виконуваних програм з файлу автозапуску, розміщення резидентної програми в пам’яті, моніторинг API-викликів і збір їх у вектори. Програмне забезпечення вузла бот-мережі може потрапити в КС двома способами: користувач, мережа. Тоді, таке ЗПЗ буде обов’язково пробувати прописати себе для можливості активації при наступному ввімкнені КС, створити можливості для розміщення своєї резидентної частини в пам’яті. При спробі прописатись в ПМ РБС його буде виявлено через здійснення самоконтролю при запуску, який закладено в функціонал ПМ, якщо ЗПЗ допустить запуск наступних після нього команд. Якщо ж ЗПЗ не допустить запуск команд ПМ, тоді він не відзвітується перед рештою ПМ РБС і буде ними вилучений з РБС, що дозволить блокувати таку КС. При другому варіанті якщо ЗПЗ не було, а з’явилось, тоді та ж стратегія, що і для першого розглянутого варіанту. А якщо ЗПЗ вже було, тоді в процесі

функціонування КС виникне конфлікт між зловмисним програмним забезпеченням вузла бот-мережі і ПМ, суть якого полягатиме в змаганні за перший запуск та доступ до оперативної пам’яті, облік для ПМ і блокування відомих антивірусних засобів. Певний час вони можуть функціонувати разом, виконуючи свої задачі незалежно один від одного. В будь-якому з варіантів ПМ виступить як об’єкт для атаки в якості приманки. Але не все ЗПЗ, а особливо бот-мережі, розроблено на основі стратегії змагання за повний контроль над КС чи перший запуск. Для приховування своєї присутності стратегія перебування в КС може передбачати, наприклад, тільки прописування в одному з файлів, які містяться у файлі автозапуску. Тоді, виявити таке ЗПЗ можна лише за його проявами, важливим з яких для вузла бот-мережі є необхідність підтримки зв’язку з своїм контролюючим центром. Тобто, для цього випадку вузол бот-мережі не проявляє активність, а очікує команди і потім запускає повний пакет програмного забезпечення необхідний для її виконання. Така стратегія дозволяє перебувати в КС тривалий час без виявлення. Але ця та інші стратегії, що закладені в механізм функціонування бот-мережі, можуть порушити інші події, які викликані сторонніми проявами. Наприклад, розглянемо перший варіант подій з табл. 4.3, тобто коли отримано команду здійснити атаку на іншу КС чи ресурс, а в цей час подібна атака відбувається на цю ж КС. Така подія може відбуватись тільки лише коли атака на КС ведеться не з цієї ж бот-мережі, вузол якої міститься в КС, а іншим зловмисним програмним забезпеченням. В цьому випадку ПМ РБС переходить до стану 7, який активується через надмірне звернення до портів та викликом функцій орієнтованих на встановлення мережних з’єднань. ПМ визначає IP адресу, куди націлена інтенсивна відправка пакетів і здійснює збільшення інтервалу надсилання пакетів через блокування відповідних пакетів. Проведення атаки на цю ж КС, яка є в локальній мережі, тому не може містити ресурсів для атаки, може здійснюватись тільки іншим зловмисником для встановлення контролю над нею. Ця атака може відбуватись з іншої КС цієї ж мережі, тоді ПМ встановлює КС, про яку повідомляє решті ПМ РБС для здійснення її дослідження, або з-за меж локальної мережі. В будь-якому

з випадків відбивання атаки здійснюватиметься відповідними засобами. Але частина атакуючих дій може бути успішною, тоді в КС будуть одночасно присутні ПМ, ПЗ вузла бот-мережі та нове ЗПЗ. При їх функціонуванні виникне конфлікт, який проявиться в спробі здійснення контролю над КС, що буде впливати на її нормальний порядок функціонування. В цьому випадку ПМ при звірці інформації про розміщені в КС файли виявить поточні зміни і почне дослідження підозрілих файлів.

У другому варіанті подій, якщо в КС міститься вузол бот-мережі і з неї здійснюється атака, ПМ порівнює зростаючу інтенсивність викликів мережних з'єднань та здійснює їх затримки методом призупинення та виявляє процесу, який їх генерує. Цей варіант можливий при повному контролі ПМ у КС і другорядній ролі вузла бот-мережі, якщо ж інакше, то тобі вузол бот-мережі змагався б за ресурси КС з ПМ і цим виявив би себе.

В третьому варіанті подій розглядаємо можливість проникнення в КС з малою ймовірністю. При цьому ПМ аналізуватиме інтенсивність звернення до портів та надходження пакетів, а також подальшого розміщенням файлів, що надійшли і фіксування місця їх розміщення. В подальшому ПМ розміщує такі файли в свій реєстр для спостереження за ними протягом певного часу.

В четвертому варіанті подій ПМ здійснює змагання за ресурси з ПЗ вузла бот-мережі. І аналогічно, як у першому варіанті, може бути дві варіації: ПЗ вузла бот-мережі прагнутиме встановити повний контроль над КС або згідно своєї стратегії функціонування приховуватиме свою присутність до отримання команди на проведення атаки.

П'ятий варіант подій включає результати першого з врахуванням того, що додатково наявне в КС файлове ЗПЗ. Його присутність ускладнюватиме функціонування КС, бо воно перебуватиме в оперативній пам'яті, здійснюватиме своє поширення шукаючи об'єкти для втілення. Характерною особливістю в цьому варіанті буде завантаженість ресурсів і сповільнення роботи КС. ПМ відмічатиме зміни в файлах, які прописані в її реєстрі для цієї КС. Основним місцем, де зіткнуться ПМ, ПЗ вузла бот-мережі та файлове ЗПЗ

буде оперативна пам’ять. Атака на цю КС підсилюватиме ускладнення роботи КС, що аналогічно до першого варіанту призведе до тривалої обробки результатів моніторингу ПМ в КС і повідомлення про події іншим ПМ РБС.

Аналогічно до першого і п’ятого варіантів будуємо стратегії розвитку подій для шостого, сьомого та восьмого варіантів подій. Варіанти 9-12 розглянемо в розділі 5, бо вони не передбачають наявність ПЗ вузла бот-мережі.

Стратегії ПМ у різних варіантах подій:

1) КС контролюється бот-мережею, ПМ у КС заблоковано вузлом бот-мережі, КС виконує поставлені користувачем задачі;

2) КС контролюється бот-мережею, ПМ в КС заблоковано вузлом бот-мережі, КС функціонує з тривалими перебоями та затримками у виконанні запитів користувача;

3) КС контролюється бот-мережею, ПМ в КС заблоковано вузлом бот-мережі, КС функціонує з тривалими перебоями та затримками у виконанні запитів користувача, файлове ЗПЗ у КС здійснює своє поширення;

4) КС контролюється ПМ, вузол бот-мережі в КС досліджується ПМ, КС виконує поставлені користувачем задачі;

5) КС контролюється ПМ, вузол бот-мережі та файлове ЗПЗ у КС досліджуються ПМ, КС виконує поставлені користувачем задачі;

6) КС контролюється ПМ, який здійснює моніторинг і обробку подій.

Проаналізуємо логіку взаємодії програмного модуля РБС та ПЗ вузла бот-мережі в КС для отримання стратегій. Задамо стадії функціонування варіантів подій та можливих стратегій їх розвитку часовою діаграмою та виділимо в ній повторювані фрагменти для здійснення оптимізації при прийнятті рішення ПМ РБС. Шаблони можливих варіантів подій у КС та їх варіації формують одну з шести стратегій. Стратегії для подій в КС локальної мережі представлено у вигляді фрагменту в табл. 4.6 та деталізовано в табл. А.2 додатку А.

Таблиця 4.6

Стратегії для подій в КС локальної мережі

№ з/п	Встановлення в КС до (1)/після(0) встановлення ПМ або ПЗ вузла бот-мережі		Стартовий контроль в КС: так(1) / ні(0)		Атака з цієї КС: так(1) / ні(0)	Атака з цієї КС на КС цієї ж мережі: так(1) / ні(0)	Атака на КС: так(1) / ні(0)	Атака з КС цієї ж мережі: так(1) / ні(0)	Наявність файлового ЗПЗ, встановленого до(1) / після(0) встановлення ПМ	Стратегії
	ПМ	Б	ПМ	Б						
1	1	0	0	1	1	1	1	1	1	1
2	1	0	0	1	1	1	1	1	0	1
…	…	…	…	…	…	…	…	…	…	…
512	0	0	0	1	0	0	0	0	0	6

Позначення:

«ПМ» – програмний модуль, «Б» – ПЗ вузла бот-мережі.

Ймовірності в станах залежать від варіантів подій, які відбуватимуться в КС, пов’язаних з присутністю ЗПЗ, та визначаються за формулами (4.22):

$$P_{j,\text{ПМ}} = \sum_{s=1,}^{8} p_{s,j}$$

$$p_{1,j} = \begin{cases} 0, \text{ПМ знаходиться тільки в стані 1 або стані 1 і 8 одночасно} \\ \frac{s}{100}, \text{ПМ перейшов до стану } s \text{ і продовжує бути в стані 1} \end{cases} \text{(зросте якщо перейде в один зі станів),}$$

$$p_{2,j} = \frac{\sum_{c=1}^{7} p_{s,j,c}}{7*6},$$

$$p_{3,j} = \frac{\sum_{c=1}^{7} p_{s,j,c}}{7*5},$$

$$p_{4,j} = \frac{\sum_{c=1}^{7} p_{s,j,c}}{7*4}, \qquad (4.22)$$

$$p_{5,j} = \frac{\sum_{c=1}^{7} p_{s,j,c}}{7*3},$$

$$p_{6,j} = \frac{\sum_{c=1}^{7} p_{s,j,c}}{7*2},$$

$$p_{7,j} = \frac{\sum_{c=1}^{7} p_{s,j,c}}{7}.$$

$p_{8,j} = \begin{cases} 0\text{, ПМ знаходиться тільки в стані 1 і 8 одночасно} \\ \frac{s}{100}\text{, ПМ перейшов до стану } s \text{ і продовжує бути в стані 1 та 8} \end{cases}$, якщо переривається ця робота, або при її виконанні активуються інші стани, а роботу по оптимізації потрібно завершити.

Узагальнена формула (4.23) для станів 2-7 така:

$$p_{s,j} = \frac{\sum_{c=1}^{7} p_{s,j,c}}{7*s}, \tag{4.23}$$

де j – тий програмний модуль РБС, s – номер стану.

Для подій, коли декілька станів одночасно працюють, розрахунок проводиться з врахуванням виконання умов.

Метод виявлення бот-мереж в локальних комп’ютерних мережах, суть якого в здійсненні активного моніторингу системних подій та узгоджених взаємодії компонентів розподіленої системи при прийнятті рішення, дає можливість створення на його основі засобів, здатних інтегруватись в розподілену систему та класифікувати бот-мережі за їх поведінковими сигнатурами, що формуються закладеними в їх компоненти функціями, і складається з таких основних кроків:

1. Отримати данні про активні процеси та мережні пакети на основі активного моніторингу виконання команд в КС, починаючи з першої API-функції кожного процесу, що буде виконуватись після запуску КС.

2. Здійснити збір даних моніторингу після виявлення певних ймовірно зловмисних проявів в КС у вектор.

3. Сформувати вектор ознак ймовірно підозрілих дій для зібраних даних, компонентами якого є API-функції.

4. Прийняти рішення про місце обробки вектору ймовірно зловмисних дій.

5. Якщо аналіз завантаженості ресурсів КС показав невеликий відсоток завантаженості, тоді здійснити обробку в цій КС, інакше надіслати в іншу визначену ПМ КС.

6. Здійснити класифікацію вектору ймовірно зловмисних дій.

7. Аналіз результатів кроку 6.

7.1. Якщо встановлено віднесення такого вектору до певного підкласу класу бот-мереж, тоді додати цю інформації до класифікаторів всіх ПМ.

7.2. Якщо встановлено віднесення такого вектору до декількох підкласів класів бот-мереж, тоді здійснити аналіз із залученням решти ПМ РБС на основі обробки варіантів подій з табл. 4.3.

7.3. Якщо близькість для включення до певного підкласу є нечіткою, але додатково із залученням решти ПМ визначено, що вектор містить зловмисні дії, тоді здійснити створення нового класу для бот-мереж, занести данні, оновити налаштування класифікатору, передати результат решті КС.

7.4. Якщо перевірка встановила, що досліджуваний вектор не містить зловмисного навантаження, тоді здійснити зупинку дослідження процесу, на основі якого він був сформований.

7.5. Якщо встановлено що досліджуваний вектор містить зловмисне навантаження, тоді здійснити зупинку відповідного процесу.

7.6. Здійснити пошук і дослідження аналогічних процесів в інших КС мережі на основі отриманих відомостей, де встановлена РБС її програмними модулями.

8. Обробити варіанти з табл. 4.3 та табл. А.2 додатку А із залученням решти ПМ РБС. На основі варіантів подій табл. 4.3 та табл. А.2 додатку А здійснити виокремлення варіантів, в яких можливе відключення зловмисним програмним забезпеченням ПМ РБС, і встановити таку подію для оцінки іншими компонентами РБС. В цьому випадку здійснити вилучення ПМ з РБС.

8.1. Для варіантів 1-256 задіяти стратегію «1» на основі прийняття рішення рештою ПМ та здійснити вилучення ПМ з РБС.

8.2. Для варіантів 257-320 задіяти стратегію «2» на основі прийняття рішення іншими ПМ РБС.

8.3. Для варіантів 321-340 задіяти стратегію «3» на основі прийняття рішення іншими ПМ.

8.4. Для варіантів 341-484 задіяти стратегію «4» на основі прийняття рішення всіма ПМ з РБС та здійснити обмін інформацією між ПМ.

8.5. Для варіантів 485-502 задіяти стратегію «5» на основі прийняття рішення всіма ПМ з РБС та здійснити обмін інформацією між ПМ.

8.6. Для варіантів 503-512 задіяти стратегію «6» на основі прийняття рішення ПМ та здійснити обмін інформацією з іншими ПМ РБС.

9. Обчислити значення ймовірностей в станах ПМ і надіслати вимогу для інших ПМ здійснити обчислення ймовірності бути ураженою для всієї РБС. Цей крок здійснюється позапланово через дослідження наявного зловмисного прояву в одній з КС.

10. Здійснити оптимізацію вектору, що додається в базу зловмисних дій та атак, за генетичним алгоритмом.

11. Сформувати значення ймовірностей перебування в станах для надсилання іншим ПМ для визначення стану РБС (за формулами (4.22)).

12. Залучити засоби для здійснення самоконтролю та забезпечення стійкості ПМ в КС при групі подій з кроку 8 (табл. 4.3), які відносяться до зовнішніх впливів.

Таким чином, згідно розробленого методу програмні модулі РБС навчені і мають змогу досягати таких цілей: вилучення ймовірно уражених ПМ з РБС, встановлення відношення до ЗПЗ типу бот-мереж на основі обміну і обробки знань, створення нового класу бот-мереж на основі фрагменту програмного коду. Розбудовані ПМ згідно методу такої обробки ймовірно зловмисних подій міститимуть засоби для прийняття рішення про зміну структури РБС та визначають уражену КС завдяки побудованій архітектурі РБС і організації

взаємозв’язку її компонентів.

Розроблений метод [379] дозволяє здійснювати виявлення бот-мереж у локальних комп’ютерних мережах. Він базується на здійсненні активного моніторингу системних подій та узгодженій взаємодії компонентів розподіленої системи при прийнятті рішення та дає можливість створювати на його основі засоби, які здатні інтегруватись в розподілену систему та класифікувати бот-мережі за їх поведінковими сигнатурами, що формуються закладеними в їх компоненти функціями.

Висновки до четвертого розділу

Запропонована еталонна модель бот-мереж, в основі якої її типові компоненти задані відповідними функціями відображають типові дії [164, 165] мережного зловмисного програмного забезпечення в локальних комп’ютерних мережах. Їх представлення дозволить виділити характеристичні ознаки бот-мережі і є основою для розробки методів їх виявлення та подальшої локалізації. Розроблена еталонна модель бот-мереж на основі компонентів трьох рівнів є основою для представлення відомих бот-мереж, які були виявлені і певним чином класифіковані за характерними ознаками.

Еталонна модель бот-мереж та розподіл відомих бот-мереж за класами [202], які відрізняються між собою, є основою створення бази для виявлення нових бот-мереж за їх характерними ознаками і віднесення до одного з класів, щоб в подальшому їх можна було використовувати при класифікації.

Розроблені для компонентів [390] еталонної моделі та моделей типових бот-мереж числові характеристики є необхідними для застосування методу виявлення бот-мереж в локальних комп’ютерних мережах.

Розроблений метод [374, 379] виявлення бот-мереж в локальних комп’ютерних мережах, суть якого полягає в здійсненні активного моніторингу системних подій та узгодженій взаємодії компонентів розподіленої системи при прийнятті рішення, дає можливість створювати засоби, які здатні інтегруватись

в розподілену систему та класифікувати бот-мережі за їх поведінковими сигнатурами, що формуються закладеними в їх компоненти функціями.

Основні результати розділу опубліковані у працях [163, 164, 201–203, 223, 227, 368, 387, 388, 390, 374, 379].

РОЗДІЛ 5
МЕТОДИ ВИЯВЛЕННЯ ФАЙЛОВОГО ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ

Файлове зловмисне програмне забезпечення характеризується тим, що його поширення першочергово має відбуватись в окремій комп'ютерній системі. Файлове ЗПЗ, що міститься у файлах виконуваних програм, після запуску виконуваних програм отримує можливість для пошуку програмних файлів для подальшого свого поширення, для виконання деструктивних дій та отримання контролю над КС з метою приховування своєї присутності. Переміщення файлового ЗПЗ в інші КС мережі може бути здійснено переважно користувачами через порушення політик безпеки. Наявність однакового за функціоналом файлового ЗПЗ в різних КС локальної мережі організації чи підприємства потребує для його виявлення наявності таких методів його виявлення, які б дозволяли враховувати результати моніторингу і сканування різних КС локальної мережі і приймати рішення про наявність ЗПЗ. Такі методи [369] можуть бути реалізовані в розподілених системах [173, 174, 224 – 226, 305]. Спроектована РБС [172] дозволяє здійснювати наповнення її рівнів різними методами виявлення ЗПЗ.

5.1. Моделі файлового зловмисного програмного забезпечення

До файлового ЗПЗ віднесемо класичні файлові віруси [373], поліморфні віруси [200, 378, 383], метаморфні віруси [224], троянські програми [228, 229, 362] і завантажувальні віруси. Середовищем їх перебування [373] в КС є оперативний запам'ятовуючий пристрій, зовнішня пам'ять та завантажувальний сектор. Відомі класичні методи виявлення такого файлового ЗПЗ, які базуються на основі сигнатурного аналізатора, не враховують появу нового ЗПЗ. Це впливає на оперативність виявлення. Тому, розглядатимемо поведінкові

сигнатури [366, 377, 389], які базуються на моделях і алгебрах [387], представлених формулами (3.1) – (3.15).

Для побудови поведінкової сигнатури [366] припустимо, що програмний об’єкт інфіковано ЗПЗ. Оскільки головною особливістю роботи файлового ЗПЗ є поширення (копіювання) повністю або частково у інший об’єкт, то об’єкт у який можливе поширення вірусного коду та його подальша коректна робота в ньому будемо називати середовищем існування вірусного коду. При побудові поведінкової сигнатури можливі середовища існування представимо у вигляді вершин графа. Також, у вигляді вершини графа зобразимо програмний об’єкт, як середовище у якому вже присутній вірусний код. Таким чином, враховуючи сучасні технології розробки файлового ЗПЗ, можна виділити три потенційних середовища його перебування [377]: програмний файл, завантажувальний сектор, ОЗП. Отже, граф поведінкової сигнатури програмного об’єкту буде мати чотири вершини: V_0 – програмний об’єкт, V_1 – програмний файл, V_2 – завантажувальний сектор, V_3 – ОЗП. Сукупність способів та засобів, які використовує вірусний код для свого поширення (та маскування) між середовищами перебування називатимемо шляхом поширення. Покажемо можливі шляхи поширення у вигляді зв’язків між вершинами графа. Отримаємо граф, зображений на рис. 5.1.

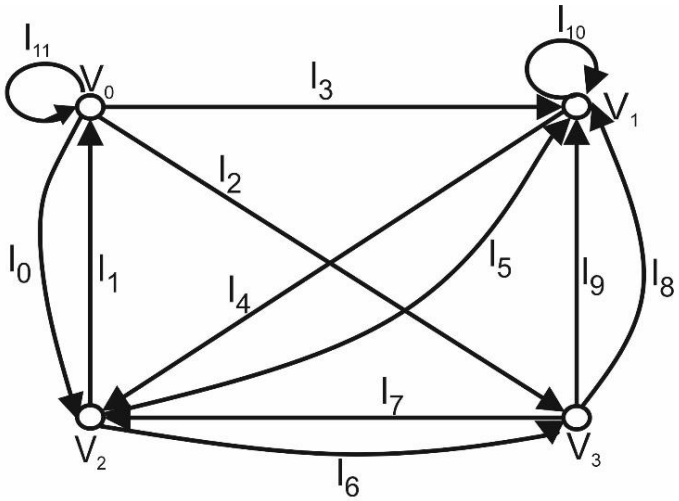


Рис. 5.1. Граф середовищ перебування файлового ЗПЗ та шляхів його поширення

Із отриманого графа [366] видно, що ребра графа: l_0 – відображає спосіб взаємодії програмного об'єкту з ОЗП, є властивим для всіх резидентних та маскованих вірусів; l_1 – відображає взаємодію програмного об'єкту з ОЗП і є властивим для всіх маскованих вірусів; l_2 – відображає спосіб взаємодії програмного об'єкту з завантажувальним сектором, даний шлях поширення властивий для завантажувальних вірусних програм; l_3 – відображає спосіб взаємодії з файлом (файлами), цей шлях поширення властивий для всіх існуючих вірусних програм; l_4, l_5 – відображають способи взаємодії вірусного коду, що знаходиться у файлі з ОЗП, даний шлях поширення властивий для резидентних та маскованих вірусів; l_6 – відображає спосіб взаємодії вірусного коду в ОЗП з кодом у завантажувальному секторі і є властивим для маскованих та завантажувальних вірусів; l_7 – відображає спосіб взаємодії вірусного коду у завантажувальному секторі з ОЗП і є властивим для завантажувальних вірусів; l_8 - відображає спосіб взаємодії вірусного коду в завантажувальному секторі з файлом (файлами) зовнішньої пам'яті і є шлях властивий для завантажувальних вірусів; l_9 – відображає шлях поширення завантажувальних вірусів; l_{10}, l_{11} – відображають шляхи приховання своєї присутності у програмного об'єкту та файлах вірусним кодом і даний шлях поширення є властивим для маскованих вірусів. Граф, який зображено на рис. 5.1, відповідає моделі програмного об'єкту для поліморфно-завантажувально-маскованого вірусу.

З розвитком інформаційних та комп'ютерних технологій для вірусних програм стануть доступними нові середовища існування, тому побудовану модель можна легко розширити шляхом введення нових вершин та ребер. Граф моделі програмного об'єкту для резидентного завантажувального віруса зображено на рис. 5.2.

Граф моделі програмного об'єкту представимо у вигляді матриці інцидентності. Вершини графа відобразимо сповпцями матриці, а ребра рядками матриці. Перед початком генерації моделі програмного об'єкту кожному елементу матриці присвоїмо значення нуль. Під час побудови матриці моделі програмного об'єкту змінюємо елементи матриці у залежності від дій

програмного об’єкту. Взаємодію середовищ перебування і шляхів поширення відображаємо у матриці таким чином: елементу матриці, який відповідає вершині графа і дузі, для якої ця вершина є початковою, присвоюємо значення «+1», а елементу матриці, який відповідає вершині графа і дузі, для якої ця вершина є кінцевою, присвоюємо значення «-1».

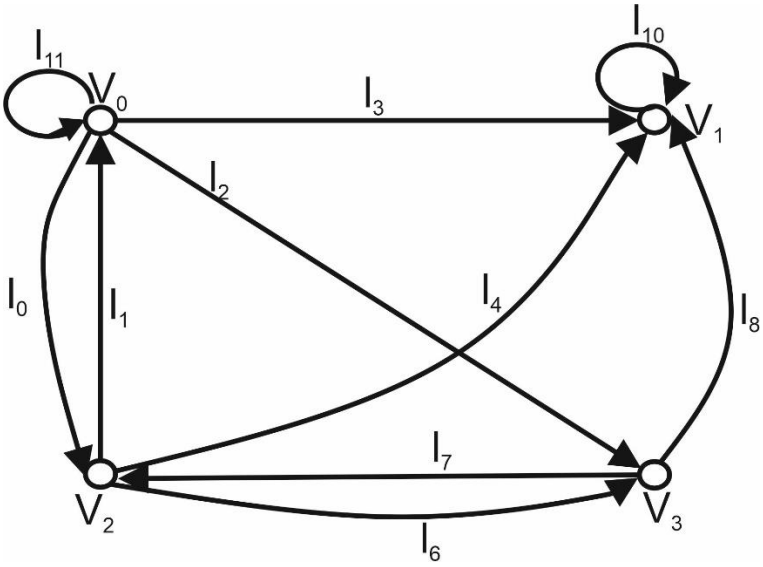


Рис. 5.2. Граф моделі програмного об’єкту для резидентного завантажувального вірусу

Таким чином, в матриці інцидентності відображено узагальнений алгоритм роботи програмного об’єкту. Крім того, нульовий стовпець відповідатиме ізольованій вершині, а нульовий рядок - петлі. Матриця інцидентності графу моделі програмного об’єкту для поліморфно-завантажувально-маскованого вірусу представлена в табл. 5.1.

Таблиця 5.1

Матриця інцидентності графу моделі програмного об’єкту

l	V_0	V_1	V_2	V_3
1	2	3	4	5
0	+1	0	-1	0
1	-1	0	+1	0
2	+1	0	0	-1

Продовження таблиці 5.1

1	2	3	4	5
3	+1	-1	0	0
4	0	-1	+1	0
5	0	+1	-1	0
6	0	0	-1	+1
7	0	0	+1	-1
8	0	-1	0	+1
9	0	+1	0	-1
10	0	0	0	0
11	0	0	0	0

Матриця інцидентності графу моделі програмного об’єкту для резидентного завантажувального вірусу представлена в табл. 5.2.

Таблиця 5.2

Матриця інцидентності графу моделі програмного об’єкту для резидентного завантажувального вірусу

l	V_0	V_1	V_2	V_3
1	2	3	4	5
0	+1	0	-1	0
1	-1	0	+1	0
2	+1	0	0	-1
3	+1	-1	0	0
4	0	-1	+1	0
5	0	0	0	0

Продовження таблиці 5.2

1	2	3	4	5
6	0	0	-1	+1
7	0	0	+1	-1
8	0	-1	0	+1
9	0	+1	0	-1
10	0	0	0	0
11	0	0	0	0

Програмний об’єкт моделюється в емуляторі процесора [377]. Для отримання еталону програмного об’єкту, в якому є файлове ЗПЗ, необхідно здійснити його лексичний [376] та синтаксичний аналіз. Якщо програмний об’єкт вимагає надати йому середовище перебування, то емулятор процесора програмного модуля РБС надає йому еталон середовища. Для створення еталонів середовищ використовується інформація з бази даних структур програмного об’єкту та бази даних типів оперційних систем, а також спеціально підготовлені еталони деяких середовищ. Матриця інцидентності створюється і будується у пам’яті у вигляді двомірного масиву. Рядки масиву відображають усі можливі шляхи поширення. Опис шляхів поширення представлено у табл. 5.3 (детальніше представлення в табл. А.3 Додатку А), у якій стовпець «Шлях l» вказує на шлях поширення згідно з графом моделі програмного об’єкту та шлях поширення згідно з графом моделі ЗПЗ у ПМ РБС. Стовпець «Port» вказує порти КС, використання яких можливе для поширення вірусного коду. Стовпець «Int» вказує вектор переривання, що відповідає даному шляху поширення. Стовпці таблиці «Ah», «Al» містять значення, яке необхідно помістити у регістри для виклику переривання. Стовпець «Дія» вказує чому сприяє даний шлях – деструктивній дії чи поширенню вірусного коду. Якщо виконується деструктивна дія, то комірка таблиці містить букву «D», якщо відбувається поширення коду, то комірка таблиці містить «R». У стовпці «Опис» подано

короткий опис для шляхів поширення.

Таблиця 5.3

Зв’язок шляхів поширення файлового ЗПЗ та команд

l	Шлях l	Port	Int	Ah	Al	Дія	Опис
1	3,10,11		13h	00h		D	Скидання пристрою
2	3,10,11		13h	01h		D	Запитати статус помилки диску
3	3,8-11		13h	02h		R/D	Читати сектор
…	…	…	…	…	…	…	…
67	0,1,4,5	080-083				R/D	Управління пам’яттю
68	3,8-11	320-32f				R/D	Доступ до жорсткого диску
69	3,8-11	070-177				R/D	Доступ до жорсткого диску
70		Інші				D	Інші порти КС

Деталізація матриці інцидентності надає можливість розділити шляхи поширення вірусного коду та шляхи поширення деструктивних дій. У випадку, якщо шлях поширення придатний для поширення, як вірусного коду, так і деструктивних дій, призначення шляху поширення визначаємо аналізуючи еталон файлового ЗПЗ, до якого було звернення цим шляхом. Отримана модель програмного об’єкту містить достатньо інформації для побудови моделі вірусної програми та моделі деструктивних дій. Матриця інцидентності графу моделі програмного об’єкту побудованої ПМ РБС для типового поліморфно-завантажувально-маскованого віруса представлена в табл. 5.4.

Таблиця 5.4

Матриця інцидентності графу моделі для типового поліморфно-завантажувально-маскованого вірусу

l	V_0	V_1	V_2	V_3
1	2	3	4	5
10	-1	0	+1	0
41	+1	0	-1	0
45	+1	0	-1	0
56	-1	0	0	+1

Продовження таблиці 5.4

1	2	3	4	5
22	-1	0	0	+1
57	-1	+1	0	0
58	-1	+1	0	0
45	-1	+1	0	0
42	-1	+1	0	0
50	-1	+1	0	0
40	1	0	0	0
39	1	0	0	0
44	0	+1	-1	0
63	0	+1	-1	0
36	1	0	0	0

Матриця інцидентності графу моделі програмного об’єкту побудованої ПМ РБС для типового резидентного завантажувального віруса представлена в табл. 5.5.

Таблиця 5.5

Матриця інцидентності графу моделі файлового ЗПЗ для типового резидентного завантажувального віруса

l	V_0	V_1	V_2	V_3
10	-1	0	+1	0
45	+1	0	-1	0
56	-1	0	0	+1
58	-1	+1	0	0
45	-1	+1	0	0
42	-1	+1	0	0
39	0	0	0	0
44	0	+1	-1	0
63	1	+1	-1	0

Модель ЗПЗ будуємо у вигляді матриці інцидентності, яка має таку саму структуру як і матриця моделі програмного об’єкту. Модель файлового ЗПЗ будуємо шляхом дослідження еталону програмного об’єкту та моделі ЗПЗ. Для побудови матриці моделі вірусної програми проводимо деталізацію деяких середовищ перебування (файл) способом поділу відпрацьованого еталону на три частини. Це необхідно для подальшого коректного знешкодження вірусного

коду. Отже, матриця інцидентності вірусної програми буде мати три сповпця для еталону файлу. Вміст кожного з стовпців буде вказувати куди записався вірусний код: на початок, в середину чи кінець файлу. Будемо вважати, що вірусний код записався на початок файлу, якшо перший байт файлу після заголовка належить потенційному вірусному коду. Будемо вважати, що вірусний код записався у кінець файлу, якщо останній байт файлу належить потенційному вірусному коду. Якщо попередні умови не виконуються, то вірусний код записався у середину файлу.

У матрицю моделі вірусної програми розміщуємо рядки з матриці моделі програмного об’єкту, які вказують лише шляхи поширення вірусного коду. Рядки матриці розташовуються у порядку використання шляхів поширення. У матрицю моделі вірусної програми не переносимо не задіяні в моделі програмного об’єкту рядки.

Матриця інцидентності графу моделі вірусної програми побудованої ПМ РБС для типового поліморфно-завантажувально-маскованого віруса, що записався у кінець програмного об’єкту представлена в табл. 5.6.

Таблиця 5.6

Матриця інцидентності графу моделі для типового поліморфно-завантажувально-маскованого віруса

l	V_0	V_{11}	V_{12}	V_{13}	V_2	V_3
1	2	3	4	5	6	7
10	-1	0	0	0	+1	0
41	+1	0	0	0	-1	0
45	+1	0	0	0	-1	0
56	-1	0	0	0	0	+1
22	-1	0	0	0	0	+1
57	-1	0	0	+1	0	0
58	-1	0	0	+1	0	0

Продовження таблиці 5.6

1	2	3	4	5	6	7
45	-1	0	0	+1	0	0
42	-1	0	0	+1	0	0
50	-1	0	0	+1	0	0
40	1	0	0	0	0	0
39	1	0	0	0	0	0
44	0	0	0	+1	-1	0
63	0	0	0	+1	-1	0

Матриця інцидентності графу моделі вірусної програми побудованої ПМ РБС для типового резидентного завантажувального віруса, який записався в початок програмного об’єкту представлена в табл. 5.7.

Таблиця 5.7

Матриця інцидентності графу моделі для типового резидентного завантажувального віруса

l	V_0	V_{11}	V_{12}	V_{13}	V_2	V_3
10	-1	0	0	0	+1	0
45	+1	0	0	0	-1	0
56	-1	0	0	0	0	+1
58	-1	+1	0	0	0	0
45	-1	+1	0	0	0	0
42	-1	+1	0	0	0	0
39	1	0	0	0	0	0
44	0	0	0	0	-1	0
63	0	+1	0	0	-1	0

Модель деструктивних дій будуємо у вигляді матриці інцидентності, яка має таку саму структуру, як матриці моделі вірусної програми. Приклад моделі

деструктивних дій вірусної програми побудованої ПМ РБС для типового поліморфно-завантажувально-маскованого віруса, що записався у кінець програмного об’єкту і не містить жодних деструктивних функцій, представлено в табл.5.8.

Таблиця 5.8

Матриця інцидентності графу моделі деструктивних дій

l	V_0	V_{11}	V_{12}	V_{13}	V_2	V_3
39	1	0	0	0	0	0
40	1	0	0	0	0	0

Приклад моделі деструктивних дій вірусної програми побудованої ПМ РБС для типового резидентного завантажувального віруса, який записався в початок програмного об’єкту, деструктивною дією якого є виведення повідомлення на екран, представлено в табл. 5.9.

Таблиця 5.9

Матриця інцидентності графу моделі деструктивних дій (виведення повідомлення на екран)

l	V_0	V_{11}	V_{12}	V_{13}	V_2	V_3
65	-1	0	0	0	+1	0

Розглянемо особливості побудови та аналізу отриманих автоматично моделей вірусних програм та деструктивних дій [366]. Якщо можливе поширення вірусного коду або поширення деструктивних дій, то необхідно проаналізувати вміст еталону програмного об’єкту у місці зміненому ланцюжком лексем, що реалізують цей шлях поширення. Якщо матриці моделей вірусних програм та деструктивних дій містять однакові рядки (один і той же шлях поширення), то це означає, що вірусний код руйнує програмний об’єкт і його відновлення неможливе. Якщо матриця деструктивних дій містить не менше двох ненульових елементів і програмний об’єкт ушкоджений вірусною програмою, то це вказує на наявність поліморфного вірусу. Якщо неможливо

побудувати матрицю вірусних дій, то програмний об’єкт вірусом не інфікований.

Розроблене представлення файлового ЗПЗ дозволяє врахувати поведінку виконуваних програм і файлового ЗПЗ [369, 366, 377, 387]. Представлення різних типів ЗПЗ [388] є основою для формування бази його еталонних поведінкових сигнатур і використати їх для порівняння з отриманими в результаті сканування сигнатурами. Використання представлення через матриці інцидентності потребує деталізації кожного її ненульового елементу матриць послідовністю викликів API-функцій, тобто розробки її поведінкової сигнатури.

5.2. Методи виявлення файлового зловмисного програмного забезпечення

Представлення еталонних зразків поведінкових сигнатур в базах поведінкових сигнатур файлового ЗПЗ через використання матриць інцидентності дозволяє розробляти методи виявлення файлового ЗПЗ [174, 226, 229, 305, 373, 389] на основі динамічно отриманих в процесі функціонування КС поведінкових сигнатур. Для реалізації таких методів в РБС використовується емулятор процесора або система дизасемблювання, які є складовими програмних модулів. При здійсненні аналізу програмного об’єкту на наявність файлового ЗПЗ в межах КС локальних мереж характерні ознаки для програмного об’єкту отримані в одній КС порівнюються з отриманими ознаками цього ж програмного об’єкту в іншій КС. Для отримання таких характерних ознак та виявлення за ними файлового ЗПЗ у КС потрібно розробити відповідні методи.

5.2.1. Метод виявлення файлового зловмисного програмного забезпечення на основі динамічного формування поведінкової сигнатури шляхом відстеження викликів API

Відоме файлове ЗПЗ є дуже різноманітним. Зокрема, для приховування його присутності в програмних об’єктах та пам’яті КС зловмисники

використовують різні методи. На основі цих методів і їх модифікацій сформувались різнотипне і змішане файлове ЗПЗ. Тому, створювані нові методи повинні охоплювати якомога більше типів файлового ЗПЗ. Оскільки поліморфний та метаморфний віруси приховують від АПЗ свій програмний код, то для їх дослідження потрібне отримання послідовності викликів їх API-функцій [389]. Тому, і для звичайного файлового ЗПЗ можуть бути застосовні методи, які враховують поліморфний [348, 378] або метаморфний функціонал, якщо їх об’єктами виявлення є можливі комбінації різних технологій. Тоді такі методи виявлення повинні використовувати дослідження поведінкових сигнатур на основі аналізу викликів API-функцій.

Використання у вірусах технології [174] заплутування програмного коду унеможливлює виокремлення сталої частини коду, аналіз якої дозволив би прийняти рішення про можливе інфікування. Проте, це стає можливим при використанні в якості основи сигнатури API-викликів функцій, тобто набору готових класів, процедур, функцій, структур і констант, що надаються застосунком або операційною системою для використання у зовнішніх програмних продуктах.

Процес виконання програм, зокрема вірусів, супроводжується використанням ними API-викликів. Наприклад, вірусна програма для пошуку виконуваних файлів, які повинні бути інфіковані, як правило, використовує наступну послідовність API викликів: FindFirstFileA, FindNextFileA і FindClose, що розміщені в бібліотеці KERNEL32.DLL. Таким чином, зазначена послідовність API-викликів може бути використана для побудови сигнатури вірусної програми. В загальному, можна відзначити, що використання API-функцій в якості сигнатури, дозволяє виокремити сталу семантичну (поведінкову) складову, в той час як синтаксична складова буде різною.

Сформуємо вимоги [389], яким має відповідати сигнатура вірусної програми на основі трасування API-викликів наступним чином:

– аналіз сигнатури має дозволяти визначати не тільки клас вірусних програм (сімейство), а й модифікацію вірусу (наприклад, у вірусу virut є

модифікації ce, a, b, тощо);

– буди компактною (мати не великий розмір), бо зі збільшенням розміру сигнатури збільшується розмір бази сигнатур та час аналізу сигнатури, і відповідно, час виявлення;

– алгоритми виявлення, що засновані на аналізі вірусної сигнатури повинні мати прийнятну часову та обчислювальну складність;

– алгоритми виявлення, що використовують запропоновану сигнатуру мають володіти високою достовірністю виявлення та низьким рівнем хибних спрацювань.

Відомі підходи до формування сигнатури на основі трасування API-викликів дозволяють здійснити процес виявлення вірусних програм, проте, основним їх недоліком є відсутність протидії застосуванню фейкових API-викликів (API-виклики, що навмисно вводяться до вірусної програми з метою заплутування поведінки). Це може ускладнити процес пошуку найбільшої спільної послідовності викликів, і відповідно, призвести до зменшення рівня виявлення.

Розглянемо формування [389] сигнатури поведінки програми на основі трасування API-викликів. При побудові сигнатури вірусу замість використання всіх API-викликів, що здійснює вірусна програма, будуть розглядатися лише критичні API-функції. Критичні API-виклики містять усі виклики API, які можуть призвести до порушення безпеки, зміни усталеної поведінки роботи системи або виклики, що використовуються для комунікації (модифікація значення системного реєстру, файлового вводу-виводу, API-доступу до мережних ресурсів (WinSock), тощо). В процесі створення сигнатури вірусної програми не розглядаються API-виклики, які можна додати або вилучати з вірусної програми без зміни її зловмисної поведінки (наприклад MessageBox, printf, malloc тощо).

Сигнатура поведінки програми на основі трасування API-викликів може бути представлена у вигляді сукупності двох складових (рис. 5.3): частота виклику та характер взаємодії критичних API-викликів. Аналіз першої складової

дозволяє визначити розподіл критичних API-викликів за групами шкідливої активності та відображає кількісну складову сигнатури. Друга складова сигнатури передбачає відображення у векторний простір характеру взаємодії критичних API-функцій вірусної програми та описує їх взаємозв'язок. Аналіз другої складової сигнатури надає можливість розмежувати вірусні програми від корисних застосунків не тільки за наявністю критичних API-викликів, але й за їх взаємодією між собою.

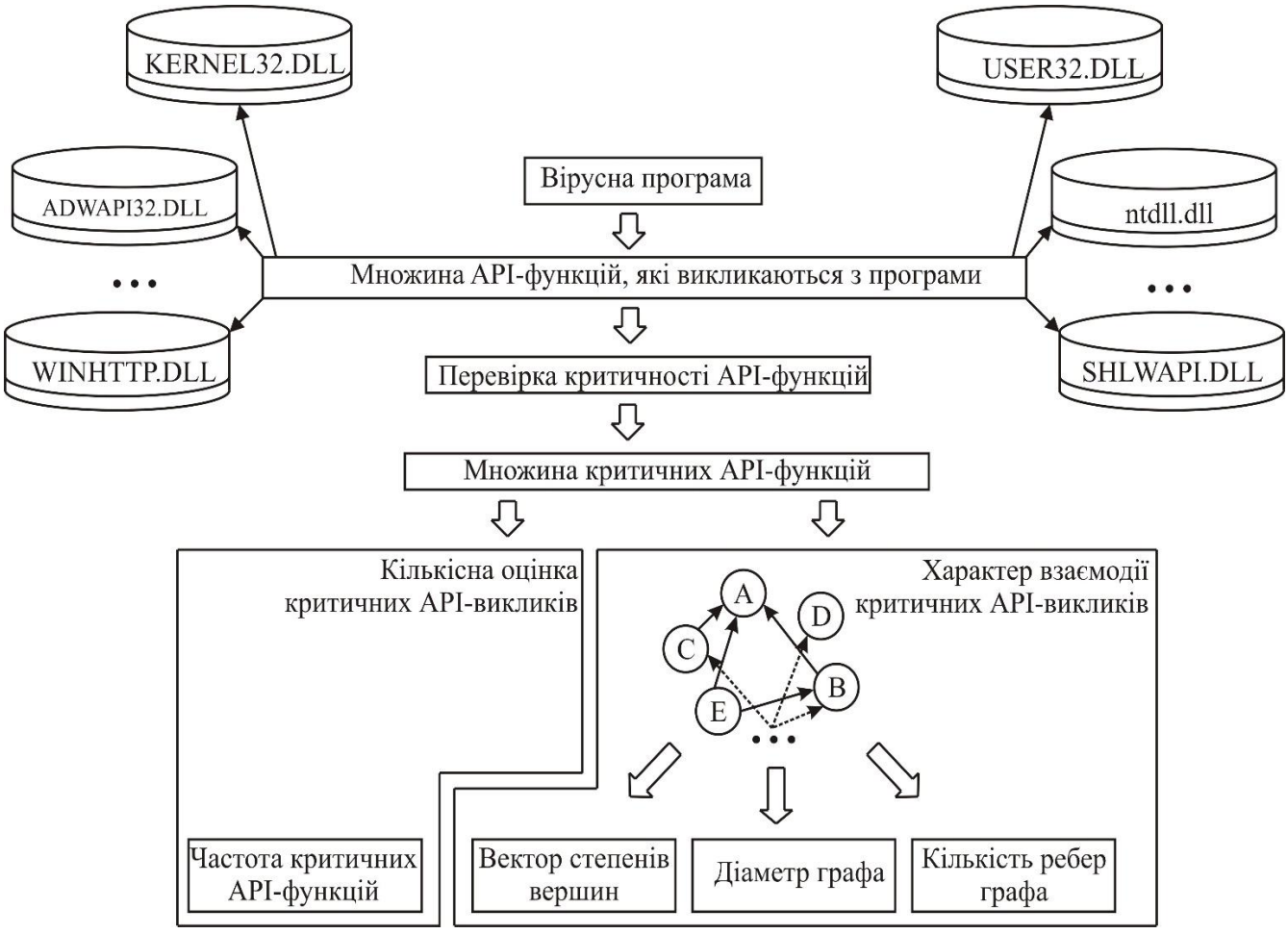


Рис. 5.3. Процес формування сигнатури поведінки вірусної програми на основі трасування API викликів

Для опису характеру взаємодії критичних API-викликів представимо вірусну програму у вигляді орієнтованого графа за формулою (5.1):

$$G_V = \langle V, E \rangle, \tag{5.1}$$

де V – множина вершин, що представляють групу критичних API-функцій; E – множина переходів між групами критичних API-функцій, послідовність яких описує поведінку вірусної програми.

Для формального подання сигнатури вірусної програми представимо її у вигляді кортежу за формулою (5.2):

$$S = \left\langle A, F, \left\langle D, d_G, n_E \right\rangle \right\rangle, \tag{5.2}$$

де A – множина API-викликів, що здійснює вірусна програма класу C_i, в процесі власного функціонування; F – множина частот виклику критичних API-функцій; D – вектор степенів вершин графу G_V; d_G – діаметр графа G_V; n_E – кількість ребер графа G_V.

При формуванні вірусної сигнатури на основі трасування API-викликів спільним для обох етапів є категоризація API-викликів за класами. В процесі аналізу та дослідження множини вірусних програм було отримано 1624 API-функції, які можна віднести до класу критичних API-функцій. Для створення сигнатури вірусних програм зазначені API-функції було розділено на 26 класів [389]. У табл. 5.10 представлені приклади класів API-функцій та їх опис. Наприклад, функції DeleteFiles та CreateDirectory визначаються як клас B, тобто функції для роботи з файлами та директоріями. У випадку, якщо послідовність API-викликів представляється функціями CallNextHookEx, isDebuggerPresent та CreateProcess, то в якості складової сигнатури отримаємо наступну послідовність “AFH”. Зазначене представлення API-викликів дозволяє зберігати поведінку програми компактно, яка представлена API-викликами. Окрім того, групування API-функцій за класами критичних дій надає можливість представити у вигляді одного позначення множину функцій, які є подібні за функціональними властивостями (наприклад, CreateProcessAsUser та CreateProcess є подібними за виконуваними функціями) та можуть використовуватись різними екземплярами,

що належать одному вірусному сімейству. В процесі формування сигнатури та категоризації критичних API-викликів не враховуються вхідні параметри та результат виконання відповідної API-функції.

Таблиця 5.10

Категоризація API-функцій за класами, їх опис та приклади

Клас API	Опис	Приклад	Кількість API-функцій
Class A	Функції перехоплення	CallNextHookEx, SetWindowsHookEx	12
Class B	Робота з файлами та директоріями	DeleteFiles, CreateDirectory, CopyFile	242
Class C	Модифікація системного реєстру	RegCreateKey, RegDeleteValue	48
Class D	Синхронізація	CreateMutex, CreateMutexEx	213
…	…	…	…
Class Z	Функції керування пристроями	DeviceControl, DvdLauncher	24

Аналіз множини частот виклику критичних API-викликів дозволяє сформувати параметр приналежності до вірусного класу, що визначає зв’язок між вірусною програмою та одним із класів вірусних програм за кількістю критичних API-викликів. Це є достатньою умовою для віднесення підозрілої програми до одного із класів вірусних програм чи корисних застосунків. Проте, аналіз даного параметру не несе інформацію про характер взаємодії між критичними API-викликами, і відповідно, не можливо віднести підозрілу програму до конкретної модифікації вірусу, а лише до цілого класу.

Тому, друга складова сигнатури вірусної програми покликана відображати характер взаємодії критичних API-функцій вірусної програми та описувати взаємозв’язок між ними, що дозволить здійснити розмежування вірусних

програм в середині класу.

З цією метою вірусну програму можна представити у вигляді орієнтованого графа. Для відображення даного графу у сигнатурі вірусної програми представимо його у вигляді множини степенів вершин D, тобто кожен елемент цієї множини визначає число інцидентних ребер для відповідної вершини. Наприклад, за множиною ступенів вершин графу, що складається з послідовності {3,3,2,2,1,1} можна побудувати такі графи, як зображено на рис. 5.4.

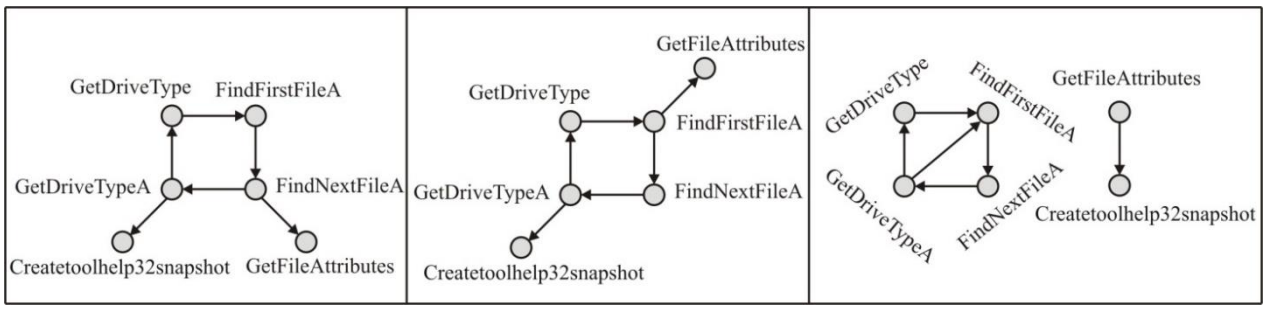


Рис. 5.4. Приклади графів з однаковим степенем вершин

Приклади графів з однаковим степенем вершин характеризуються наявністю постійної складової (з'єднання у вигляді квадрата). Подібні закономірності притаманні для вірусних програм та для об'єктів з інших галузей [30] при їх представленні графами. Розрахунок степенів вершин графу можна здійснити на основі представлення графів їх матрицями інцидентності [377] або на основі аналітичних виразів, зокрема при визначенні [30] First Zagreb Index.

Окрім, множини степенів вершин для розмежування вірусних програм в середині класу виокремимо ще дві ознаки: діаметр графа та кількість ребер. Діаметр графа визначатиме максимальну послідовність виклику критичних API-функцій, в той час як кількість ребер – загальну кількість виконуваних вірусною програмою дій.

Оскільки, однією із складових запропонованої сигнатури є множина частот виклику критичних API-викликів, то на основі елементів цієї множини здійснимо

визначення параметру приналежності до вірусного класу. Він визначає ступінь належності екземпляра вірусу до одного із класів вірусних програм та дозволяє оцінити зв'язок між вірусними програмами в межах одного класу. Таким чином, база поведінкових сигнатур, окрім частот виклику критичних API-функцій, повинна містити параметр приналежності до вірусного класу.

Процес визначення параметру приналежності до вірусного класу заснований на відмінності між кількістю API-викликів, які здійснюють вірусні програми та довірені застосунки в процесі власного функціонування. Тому, розмежування між класами вірусних програм та довірених застосунків можливе за їх поведінкою, тобто послідовністю критичних API-викликів. Розглянемо детальніше процес визначення параметру приналежності до вірусного класу.

З метою побудови поведінки класу вірусних програм $C_i = \{c_i^1, c_i^2, ..., c_i^x\}$ на основі частот виклику критичних API-функцій представимо поведінку екземпляра цього класу у вигляді кортежу (c_i^j – екземпляр класу C_i , $j = \overline{1, x}$, де x – кількість екземплярів вірусних програм, що визначають поведінку класу C_i) за формулою (5.3):

$$c_i^j = \langle f_1, f_2, ..., f_{26} \rangle, \tag{5.3}$$

де $f_1, f_2, ..., f_{26}$ – частоти критичних API-функцій.

Згрупуємо всі значення частот виклику критичних API-функцій c_i^j ($\forall c_i^j \in C_i$) та представимо їх у вигляді матриці R_{C_i} за формулою (5.4):

$$R_{C_i} = \begin{bmatrix} c_i^1 = \langle f_1, f_2, ..., f_{26} \rangle \\ c_i^2 = \langle f_1, f_2, ..., f_{26} \rangle \\ ... \\ c_i^x = \langle f_1, f_2, ..., f_{26} \rangle \end{bmatrix}, \tag{5.4}$$

На основі сформованої матриці R_{C_i} визначимо поведінку вірусних програм класу C_i як множину середніх значень викликів кожного класу критичних API-функцій за формулою (5.5):

$$S_{C_i} = \langle F_1, F_2, ..., F_{26} \rangle, \tag{5.5}$$

де кожне значення F_i визначається за формулою (5.6):

$$F_i = \frac{1}{x} \sum_{j=0}^{x} f_j, \tag{5.6}$$

де j – клас критичних API-функцій.

На основі отриманої поведінки вірусного класу S_{C_i} здійснюється визначення параметру приналежності до вірусного класу C_i з використанням χ^2–тесту. χ^2–тест визначає максимальну ймовірність тесту статистичної значущості, яка вимірює різницю між пропорціями в двох незалежних зразках.

Представимо сигнатуру S_{C_i} для класу вірусних програм C_i та кожну з поведінок екземплярів c_i^j у вигляді таблиці спряження (табл. 5.11).

Таблиця 5.11

Таблиця спряження для сигнатури S_{C_i} для класу вірусних програм C_i та поведінки екземпляру c_i^j

Клас	Class A	Class B	…	Class Z	Разом:
Сигнатура S_{C_i}	…	…	…	…	…
Екземпляр c_i^j	…	…	…	…	…
Разом:	…	…	…	…	…

Тоді, для отримання параметру приналежності до вірусного класу C_i за

допомогою χ^2–тесту визначимо різницю між пропорціями в сигнатурі вірусного класу S_{C_i} та кожної з поведінки екземплярів c_i^j з поправкою на неперервність (з поправкою Йєтса) за формулою (5.7):

$$\chi_j^2 = \sum_{l=1}^{26} \frac{(|\, c_{i,l}^j - S_{C_{i,l}} \,| - 0.5)^2}{S_{C_{i,l}}}, \tag{5.7}$$

де l – відповідний клас критичних API-викликів.

В результаті виконання даного етапу отримаємо множину пар значень (χ_i^2, c_i^j).

Наступний етап методу передбачає визначення усередненого значення параметру приналежності до вірусного класу C_i . З цією метою середнє значення визначатимемо за формулою (5.8):

$$\mu_{C_i} = \frac{1}{x} \sum_{i=1}^{x} \chi_i^2. \tag{5.8}$$

Таким чином, параметр μ_{C_i} визначає ступінь приналежності екземпляру c_i^j до вірусного класу та дозволяє оцінити рівень зв’язку вірусних програм в межах одного класу C_i .

Після створення сигнатури поведінки програми на основі трасування API-викликів та визначення параметру приналежності для кожного класу вірусних програм та довірених застосунків розглянемо послідовність кроків, необхідних для виявлення вірусної програми, що представляється заданою сигнатурою.

Нехай для кожного класу вірусних програм та корисних застосунків визначено параметр приналежності μ_{C_i} до відповідного класу та сигнатура підозрілої програми S, аналіз якої дозволить зробити висновок про присутність вірусу визначеного типу.

Перший крок методу виявлення [389] передбачає визначення

приналежності підозрілої програми до одного із класу вірусних програм або корисних застосунків (рис. 5.5). З цією метою за допомогою χ^2–тесту (формула (5.7)) визначається різниця між пропорціями частоти критичних API-викликів підозрілої програми (перша частина сигнатури S) та частотою критичних API-викликів кожного класу (формула (5.5)).

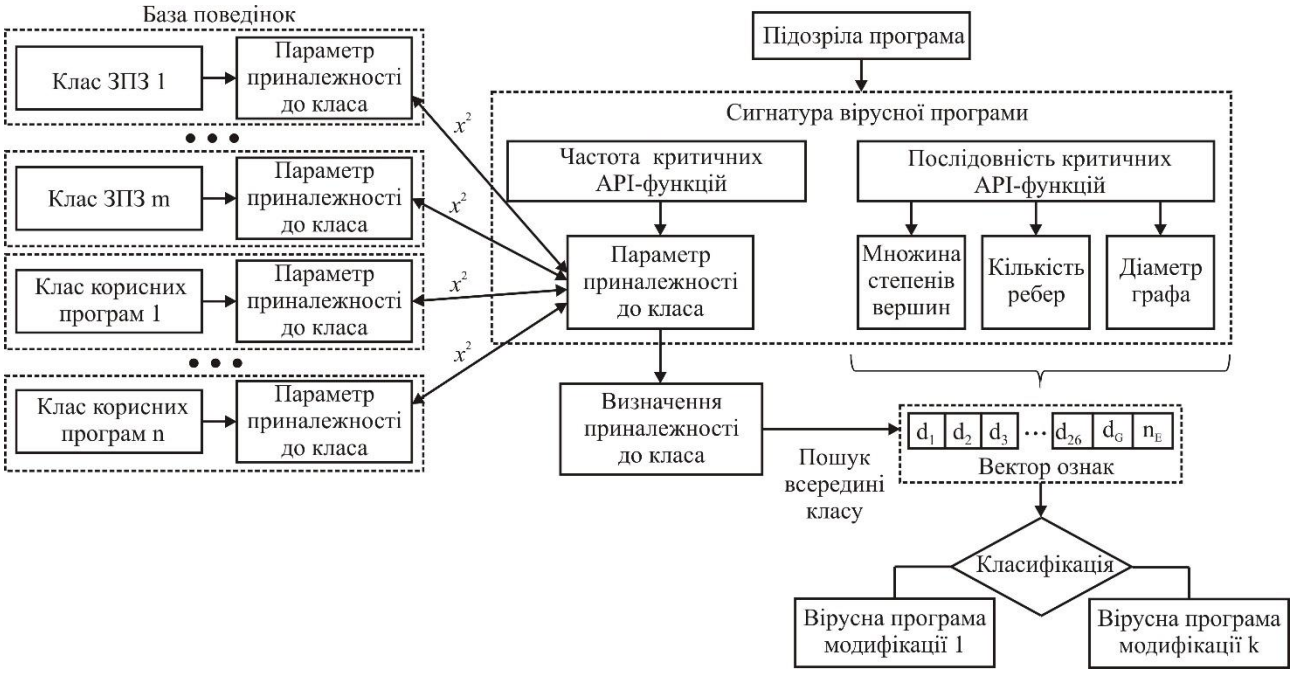


Рис. 5.5. Процес виявлення вірусних програм з використанням розробленої поведінкової сигнатури

В результаті даного кроку буде отримано множину значень параметру приналежності підозрілої програми до кожного із класів. Тоді клас, що найкраще відповідає заданій сигнатурі, визначається виходячи з виконання умови заданої формулою (5.9):

$$\min(|\ \mu_{S_j} - \mu_{C_i}\ |)\ , \tag{5.9}$$

де μ_{S_j} – значення параметру приналежності підозрілої програми до кожного із j класів.

В результаті буде визначено клас C_i, який за частотою критичних API-

викликів відповідає частоті викликів критичних API-функцій підозрілої програми.

Наступний крок методу передбачає пошук вірусної сигнатури в межах класу C_i . Опишемо другу складову сигнатури з формули (5.2) у вигляді вектору ознак V за формулою (5.10):

$$V = \langle D, d_G, n_E \rangle \tag{5.10}$$

Зазначений вектор складається з 28 числових ознак, в якому 26 ознак визначають степені вершин графа (кожна вершина графа визначається класом критичних API-викликів), а останні дві – діаметр графа та кількість ребер.

Після формування вектору ознак здійснюється його класифікація засобами машинного навчання, що дозволяє віднести досліджувану підозрілу програму до однієї з модифікацій вірусу.

Метод виявлення файлового зловмисного програмного забезпечення на основі динамічного формування поведінкової сигнатури [389] шляхом відстеження викликів API представимо такими основними кроками:

1. Формування сигнатури поведінки програми.

1.1. Формування сигнатури для класу зловмисних програм на основі відстеження викликів API кожного екземпляру файлового ЗПЗ.

1.2. Визначення ступеня входження кожного зразка до класу зловмисних програм. Тобто, знаходження усередненої оцінки для кожного класу файлового ЗПЗ, яка показує відхилення значень в класі.

1.3. Створення бази даних для класів поведінки ЗПЗ та ступенів його приналежності для кожного класу.

2. Виявлення зловмисної програми, представленої поведінковою сигнатурою, на основі трасування викликів API-функцій.

2.1. Моніторинг виконуваних файлів та їх відстеження API-викликів з використанням емулятора ПМ РБС.

2.2. Побудова поведінкової сигнатури підозрілої програми засобами ПМ.

2.3. Пошук сигнатур вірусів у класі та визначення того, чи належить підозріла програма до одного з класів зловмисних програм. Тобто, знаходження значення χ^2–тесту за формулою (5.7).

2.4. Віднесення ЗПЗ до відповідного класу зловмисних програм на основі формули (5.9).

Розроблений метод виявлення файлового зловмисного програмного забезпечення на основі динамічного формування поведінкової сигнатури шляхом відстеження викликів API може бути використаний для виявлення інших видів вірусних програм, зокрема нових версій існуючих вірусів. Для здійснення процесу виявлення нових сімейств, екземпляри вірусів повинні бути присутні у тестовій вибірці для навчання системи. Метод формування сигнатури вірусної програми на основі трасування API-викликів встановлює порядок формування поведінкової сигнатури для використання її в процесі дослідження файлового ЗПЗ.

Сигнатура поведінки програми на основі трасування API-викликів може бути представлена у вигляді сукупності двох складових: частота виклику та характер взаємодії критичних API-викликів. Аналіз першої складової дозволяє визначити розподіл критичних API-викликів за групами шкідливої активності та відображає кількісну складову сигнатури. Друга складова сигнатури передбачає відображення у векторний простір характеру взаємодії критичних API-функцій вірусної програми та описує взаємозв’язок між критичними API-функціями. Аналіз другої складової сигнатури надає можливість розмежувати вірусні програми від корисних застосунків не тільки за наявністю критичних API-викликів, але й за їх взаємодією між собою.

Для здійснення перевірки файлів виконуваних програм ПМ РБС створює для кожного програмного об’єкту процес, в якому виконується агент. Кількість таких агентів, тобто породжених командою для перевірки файлів виконуваних програм відповідає кількості необхідних для перевірки програмних об’єктів. В процесі постановки такого завдання для перевірки ПМ здійснює самоконтроль

для уникнення блокування або сповільнення роботи КС. При цьому частина агентів виконають свою роботу або будуть виконувати її, а частина агентів ПМ буде підготовлена для виконання завдання. Після виконання завдання агент повідомляє ПМ результат і зникає (знищується як процес). Таким чином, ПМ породжує велику кількість типових процесів з різними вхідними даними і завершення цих процесів закінчується їх знищенням. Ці процеси вважатимемо програмними агентами першого типу, які здійснюють завдання сканування і при його успішному завершенні зникають. До агентів другого типу віднесемо ті, які викликаються, тобто породжуються агентами першого типу, за потреби детальнішого сканування з використанням решти ПМ РБС. Після завершення своєї роботи агенти другого типу теж зникають. Програмні агенти третього типу вирішують завдання самостійно в межах КС без залучення інших ПМ РБС. Результатом їх роботи може бути або виявлення ЗПЗ або підтвердження, що ЗПЗ немає. Тоді вони повідомляють про результат роботи і теж зникають. Обробку програмного об’єкту в іншій КС, отриманого від агенту першого типу, здійснює агент четвертого типу, який здійснює побудову поведінкової сигнатури із залученням засобів КС за підтримки ПМ цієї КС. Таким чином, чотири типи агентів формують для завдання перевірки файлів виконуваних програм мультиагентну систему, яка в межах РБС визначає спільні цілі, вирішує спільні завдання. Наприклад, використання агентів в мультиагентній системі [227, 367] дозволяє скординувати роботу всієї системи для здійснення пошуку проявів бот-мереж в різних КС.

Реалізація методу виявлення файлового зловмисного програмного забезпечення на основі динамічного формування поведінкової сигнатури шляхом відстеження викликів API засобами РБС дозволить його багатократне одночасне застосування в різних КС і в кожній КС (за наявності однакових файлів в різних каталогах) та обробку отриманих результатів з різних ПМ для прийняття рішення про наявність в програмному об’єкті вірусного коду.

5.2.2. Метод виявлення файлового ЗПЗ на основі аналізу функцій обфускації

З метою виявлення файлового ЗПЗ з наявним в них поліморфним та метаморфним функціоналом [174, 389] здійснимо аналіз потенційно підозрілої поведінки програм в різних КС та аналіз особливостей заплутування коду (обфускації), які демонструються під час функціонування програми. Функції обфускації можна отримати на основі еквівалентного функціонального блокового пошуку в підозрілій програмі та її модифікованому варіанті. Висновок про наявність поліморфних та метаморфних вірусів проводиться ПМ РБС з використанням зібраної інформації від всіх ПМ мережі, в яких воно досліджувалось.

Метод включає наступні кроки: попередню обробку даних; локалізацію місця для пошуку еквівалентних функціональних блоків (ЕФБ) у виконуваному файлі; пошук ЕФБс, вибір уточнення ЕФБс; одержання кількісних ознак заплутування коду, що базується на порівнянні ЕФБс з ймовірно зловмисною програмою та її видозміненою версією; висновок про наявність поліморфних або метаморфних вірусів. Схема кроків методу зображена на рис. 5.6.

Основні кроки методу такі:

1. Попередня обробка даних.

Для виявлення поліморфних і метаморфних вірусів основною стадією є знаходження особливостей обфускації. Розглядатимемо кількісні особливості обфускації коду. Це дозволить оцінити різницю між двома копіями поліморфного або метаморфного вірусу. Для цього виконуємо розбирання підозрілої програми та отримання зразка коду перед емуляцією. Для створення модифікованої версії підозрілої програми здійснюємо її запуск в емуляторах процесорів в різних КС. Ця дія виконується за підтримки програмних модулів РБС. Результатом стадії попередньої обробки даних є декілька (більше одного) списків операційних кодів підозрілої програми та її модифіковані версії.

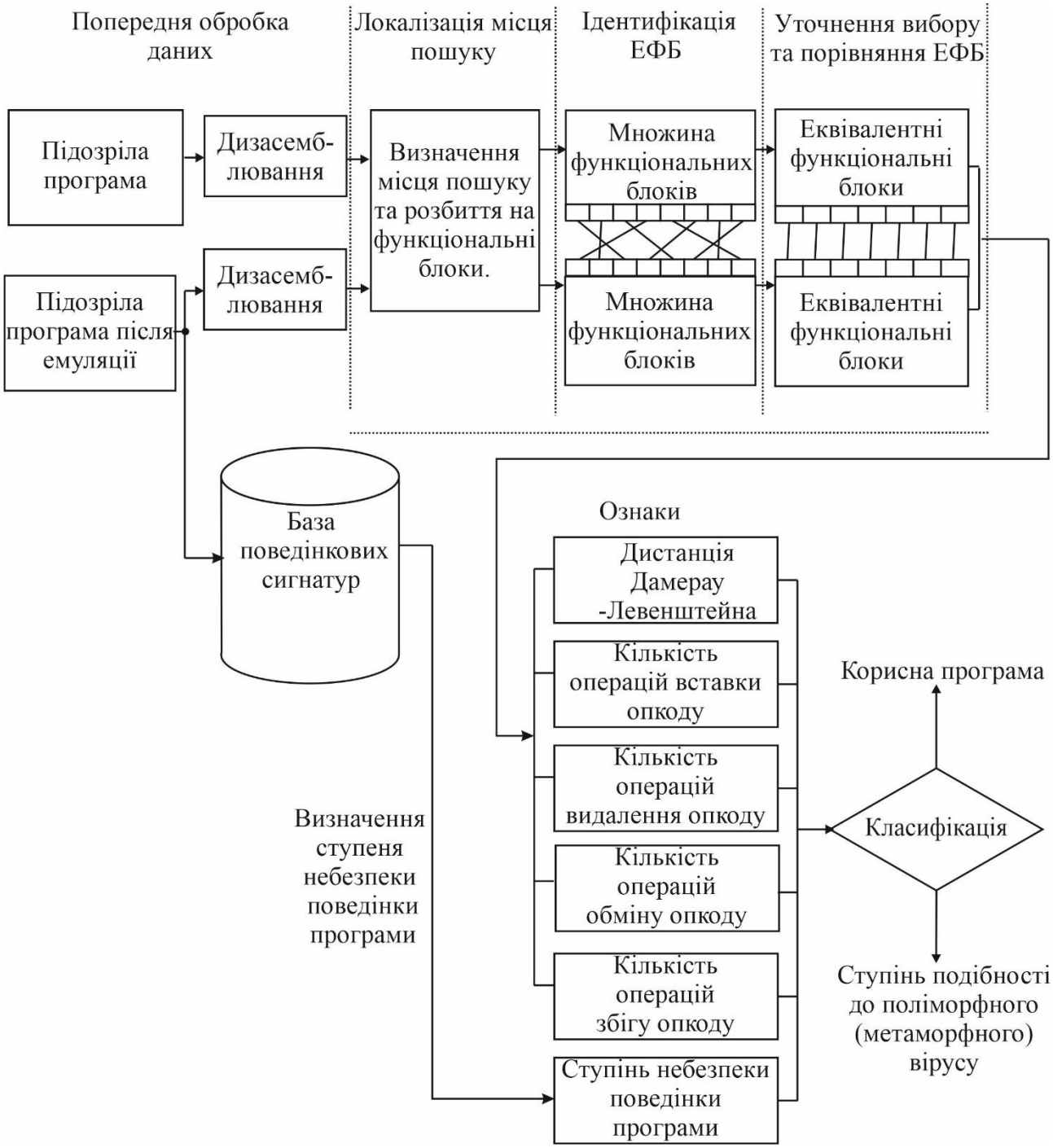


Рис. 5.6. Схема кроків методу

2. Локалізація місця для пошуку еквівалентних функціональних блоків у виконуваному файлі.

Особливості знаходження поліморфних і метаморфних вірусів вимагають локалізації місця для пошуку ЕФБ у виконуваному файлі. З цією метою виконується визначення точки входу програми та вибору розділу, в якому він знаходиться. Виберемо локалізацію місця для пошуку ЕФБ в таких випадках:

якщо в точці входу програми існують інструкції виклику або переходу, а операнди містять адресу іншої секції; якщо є розділ з нестандартним ім'ям; якщо існує виконуваний атрибут доступу розділу.

3. Пошук еквівалентних функціональних блоків.

Після того, як визначено місце пошуку ЕФБ ймовірно зловмисної програми та її модифікованої версії, наступний етап включає розбиття списків операційних кодів на блоки. Процес розбиття списків операційного коду на функціональні блоки (ФБ) передбачає розбирання підозрілої програми та її модифікованої версії, а також поділ послідовностей операційних кодів, розташованих між командами умовних або безумовних переходів.

Ймовірно зловмисну програму і її модифіковані версії представимо як вектори їх ознак. Щоб представити отримані ФБ як статистичну модель, побудуємо рейтингову матрицю появи кодів для кожного ФБ. З цією метою до кожного функціонального блоку застосовуємо статистичну метрику частоти входження компонентів у вектори, яка задається за формулою (5.11):

$$s = \frac{n_i}{\sum_{i=1}^{k} n_i} * \ln \frac{n_i+1}{N+1}, \tag{5.11}$$

де n_i - число і-го вигляду коду операції в функціональному блоці; k - кількість операційних кодів у функціональному блоці, де N - загальна кількість інструкцій асемблера.

В якості метрик подібності [174] можна використати, також, такі метрики: евклідову метрику (формула (5.12)), квадрат Евклідової метрики (формула (5.13)), відстань таксі (формула (5.14)), відстань Чебишева (формула (5.15)) і відстань Мінковського (формула (5.16)).

$$m1 = E_r(F_a, F_b) = E_p(F_a, F_b) = \sqrt{\sum_{i=1, j=1}^{\max(m,n)} (F_{a_i} - F_{b_j})^2} \tag{5.12}$$

$$m2 = E_r(F_a, F_b) = E_p(F_a, F_b) = \sum_{i=1, j=1}^{\max(m,n)} (F_{a_i} - F_{b_j})^2 \quad (5.13)$$

$$m3 = E_r(F_a, F_b) = E_p(F_a, F_b) = \sum_{i=1, j=1}^{\max(m,n)} | F_{a_i} - F_{b_j} | \quad (5.14)$$

$$m4 = E_r(F_a, F_b) = E_p(F_a, F_b) = \max(| F_{a_i} - F_{b_j} |) \quad (5.15)$$

$$m5 = E_r(F_a, F_b) = E_p(F_a, F_b) = \sum_{i=1, j=1}^{\max(m,n)} | F_{a_i} - F_{b_j} |^r , \quad (5.16)$$

де r=3, m і n - номери блоків F_h в підозрілій програмі, і її модифікована версія, відповідно. Результати порівняння використання різних метрик для визначення подібності представлено в [174].

На наступному кроці, матриці рейтингів появи кодових опкодів у ФБ з ймовірно зловмисною програмою та її видозміненою версією представимо матричними виразами за формулами (5.17):

$$F_a = \begin{pmatrix} F_{a_1} \\ F_{a_2} \\ \dots \\ F_{a_n} \end{pmatrix} \Rightarrow \begin{pmatrix} s_{11} & s_{21} & \dots & s_{k1} \\ s_{12} & s_{22} & \dots & s_{k2} \\ \dots & \dots & \dots & \dots \\ s_{1n} & s_{2n} & \dots & s_{kn} \end{pmatrix}$$

$$F_b = \begin{pmatrix} F_{b_1} \\ F_{b_2} \\ \dots \\ F_{b_m} \end{pmatrix} \Rightarrow \begin{pmatrix} s_{11} & s_{21} & \dots & s_{g1} \\ s_{12} & s_{22} & \dots & s_{g2} \\ \dots & \dots & \dots & \dots \\ s_{1m} & s_{2m} & \dots & s_{gm} \end{pmatrix} \quad (5.17)$$

В результаті виконання таких кроків методу отримуємо два набори ФБ, які представлені у вигляді матриць оцінок появи кодів операцій у функціональних блоках ймовірно зловмисної програми та її видозміненої версії. Пошук ЕФБ передбачає попарне порівняння кожного ФБ з множини з кожним ФБ з використанням метрики подібності. В результаті порівняння отримано оцінку подібності між двома ФБ. Якщо значення показника подібності для двох ФБ

менше, ніж заданий поріг, то виконується перерахунок оцінки подібності між функціональним блоком програми і наступним блоком, який слідує за блоком. Ці дії повторюються до тих пір, поки значення оцінки подібності не буде меншим або рівним пороговому значенню. Порогове значення визначається експериментально.

На етапі визначення еквівалентних функціональних блоків може виникнути подія, коли декілька функціональних блоків можуть відповідати одному ФБ. Нехай q - число ФБ з множини, які еквівалентні ФБ, для певного порогового значення подібності для двох функціональних блоків. Тому, щоб усунути невизначеність і визначити еквівалентні ФБ недвозначно, необхідно виконати вибір уточнення ЕФБ.

4. Вибір уточнення еквівалентних функціональних блоків і його порівняння.

Для виконання вибору уточнення ЕФБ необхідно створити матрицю ймовірностей для послідовності операційних кодів функціональних блоків, які визначені як еквівалентні. Стовпці та рядки матриці визначають коди операцій, які присутні у функціональному блоці. Кожна клітинка матриці полягає у відношенні числа виникнення пари опкодів до загального числа кодів операцій в рядку.

Процедура вибору уточнення ЕФБ включає порівняння матриць ймовірностей для послідовності операційних кодів кожного функціонального блоку з кожним ФБ з набору, які є еквівалентними, і вибором мінімального значення оцінки подібності.

Для оцінки подібності використовуємо метрику подібності. Для оцінки кількісних особливостей обфускації коду здійснюємо порівняння отриманих пар ЕФБ з використанням метрики, наприклад Дамерау-Левенштейна. В результаті отримуємо кількісні особливості обфускації: відстань Дамерау-Левенштейна, кількість вставок, видалення, транспозиції та збігів опкодів.

Поділ ЗПЗ з поліморфним та метаморфним навантаженням на класи представлено в [378]. Для виділених рівнів, що відповідають класам, розроблено

набір типової поведінки, в якому описують типові дії поліморфних вірусів, зокрема «команд-сміття», розшифровувача та тіла вірусу. Застосування розроблених моделей надало змогу створити поведінкові сигнатури такого типу файлового ЗПЗ.

Отримання висновку про ступінь подібності ймовірно зловмисної програми до поліморфного та метаморфного вірусу на основі отриманих ознак може бути здійснено за допомогою системи нечіткого логічного висновку [202]. На результат достовірності виявлення впливатиме метрика для визначення подібності. Підбір такої метрики необхідно здійснювати з метрик (5.12)-(5.16) на основі зразків тестового ЗПЗ.

Розроблений метод виявлення поліморфних та метаморфних вірусів на основі аналізу функцій обфускації базується на використанні РБС для отримання різних модифікованих версій ймовірно зловмисної програми. Для встановлення факту обфускації здійснюється аналіз подібності функціональних блоків на еквівалентність, які отримані на основі пошуку у ймовірно зловмисній програмі та її модифікованій версії. В результаті порівняння враховують такі кількісні ознаки обфускаціїі: відстань Дамерау-Левенштейна [174], кількість вставок, видалення, транспонування і збіги операційних кодів. На ефективність виявлення може впливати вибір метрик подібності на етапах пошуку і вибору уточнення еквівалентних функціональних блоків.

5.3. Метод виявлення файлового ЗПЗ у локальних комп’ютерних мережах

Метод виявлення файлового зловмисного програмного забезпечення в локальних комп’ютерних мережах [386, 381] полягає в поєднанні роботи програмних агентів, що здійснюють виявлення зловмисного програмного забезпечення в окремих комп’ютерних системах, відповідно до імплементованих в них методів: динамічного формування поведінкової сигнатури шляхом відстеження викликів прикладного програмного інтерфейсу, знаходження

поліморфного та метаморфного програмного коду, сканування виконуваних програм шляхом створення для них автономних процесів та відповідних програмних агентів в розподіленій системі. Це дозволяє покращити аналіз та підвищити достовірність виявлення зловмисного програмного забезпечення. Метод складається з таких основних кроків:

1. Здійснити сканування виконуваних файлів із створенням окремих процесів для кожного досліджуваного виконуваного файлу. Всі процеси створюються запуском одного компоненту ПМ у КС, який на основі закладеного функціоналу для сканування приймає рішення про потребу застосування відповідних методів виявлення.

2. Здійснити збір даних моніторингу після виявлення певних ймовірно зловмисних проявів в КС у вектор.

3. Сформувати вектор ознак ймовірно підозрілих дій для зібраних даних, компонентами якого є API-функції.

4. Застосувати метод виявлення файлового ЗПЗ на основі динамічного формування поведінкової сигнатури шляхом відстеження викликів API-функцій.

5. Прийняти рішення про місце обробки вектору ймовірно зловмисних дій.

6. Якщо аналіз завантаженості ресурсів КС показав невеликий відсоток завантаженості, тоді здійснити обробку в цій КС, інакше надіслати в іншу визначену ПМ КС.

7. Аналіз результатів кроку 6.

7.1. Якщо встановлено віднесення такого вектору до файлового ЗПЗ, тоді здійснити застосування методу виявлення файлового ЗПЗ на основі аналізу функцій обфускації для перевірки на наявність поліморфного та метаморфного вірусу.

7.2. Якщо перевірка встановила, що досліджуваний вектор не містить зловмисного навантаження, тоді здійснити зупинку дослідження процесу, на основі якого він був сформований.

7.3. Якщо встановлено, що досліджуваний вектор містить зловмисне навантаження, тоді здійснити зупинку відповідного процесу та розпочати

дослідження файлів з таким же іменем в інших КС.

7.4. Здійснити пошук і дослідження на основі отриманих відомостей аналогічних файлів в інших КС мережі, де встановлена РБС її програмними модулями. Сканування файлів в одній КС викликає такі ж сканування в інших ПМ, оскільки однакові файли, якщо це організація чи підприємство, міститимуться в різних КС. Тоді, якщо це так, то здійснювати порівняння їх поведінкових сигнатур.

8. Обробка варіантів з табл. 4.3 та табл. А.2 додатку А із залученням решти ПМ РБС. На основі варіантів подій, заданих в табл. 4.3, а також в табл. А.2 додатку А здійснити виокремлення варіантів, в яких можливе відключення зловмисним програмним забезпеченням ПМ РБС, і встановлення такої події для оцінки іншими компонентами РБС. В цьому випадку здійснити вилучення ПМ з РБС.

8.1. Для варіантів 1-256 задіяти стратегію «1» на основі прийняття рішення рештою ПМ та здійснити вилучення ПМ з РБС.

8.2. Для варіантів 257-320 задіяти стратегію «2» на основі прийняття рішення рештою ПМ РБС.

8.3. Для варіантів 321-340 задіяти стратегію «3» на основі прийняття рішення рештою ПМ.

8.4. Для варіантів 341-484 задіяти стратегію «4» на основі прийняття рішення всіма ПМ з РБС та обміну інформацією між ПМ.

8.5. Для варіантів 485-502 задіяти стратегію «5» на основі прийняття рішення всіма ПМ з РБС та обміну інформацією між ПМ.

8.6. Для варіантів 503-512 задіяти стратегію «6» на основі прийняття рішення ПМ та обміну інформацією з іншими ПМ РБС.

9. Обчислення значення ймовірностей в станах ПМ і вимога для інших ПМ здійснити обчислення ймовірності бути ураженої для всієї РБС. Цей крок здійснюється позапланово через дослідження наявного зловмисного прояву в одній з КС.

10. Здійснення оптимізації вектору, що додається в базу зловмисних дій та

атак, за генетичним алгоритмом.

11. Формування ймовірностей перебування в станах для надсилання іншим ПМ для визначення стану РБС.

12. Залучити засоби для здійснення самоконтролю, використовуючи внутрішній планувальник ПМ РБС, та забезпечення стійкості ПМ в КС при групі подій з кроку 8, які відносяться до зовнішніх впливів.

Таким чином, згідно розробленого методу виявлення файлового ЗПЗ у локальних комп'ютерних мережах програмні модулі РБС [386, 381] можуть здійснити вилучення ймовірно уражених ПМ з РБС, встановити відношення до файлового ЗПЗ на основі обміну і обробки знань, сканувати виконувані файли створенням для них окремих процесів. ПМ на основі такого методу виявлення файлового ЗПЗ у локальних комп'ютерних мережах приймають рішення про зміну архітектури РБС та визначають уражену КС.

Застосування методу виявлення файлового ЗПЗ у локальних комп'ютерних мережах до поліморфних та метаморфних вірусів може бути розширено та бути застосовано до інших типів файлового ЗПЗ, зокрема до троянських програм [229, 382]. Троянські програми використовуються не тільки як самостійні програми для вирішення задач зловмисників, але вони можуть бути ефективно застосовними і для поширення бот-мереж. Для виявлення троянських програм в окремій КС використаємо механізм нечіткого логічного висновку [305], який базується на продукційних правилах прийняття рішення. В основі методу пошуку троянських програм використаємо елементи пошуку аномалій та експертних систем. Введемо нечіткі множини та функції приналежності, яка зв'язує їх лінгвістичною змінною. Дії троянських програм розглядатимемо на етапах проникнення в КС, активізації та виконання деструктивних дій. Ці етапи утворюють один трьохрівневий життєвий цикл. Для троянських програм логічний висновок формуємо на основі нечіткої моделі, яка базується на семантичному описі та аналізі множини їх можливих станів. Для кожного етапу життєвого циклу будуємо матрицю відношень досліджуваного програмного об'єкту і компонентів операційної системи. Поведінка троянських програм на

етапах життєвого циклу багатоваріантна і має нечіткий характер, що важко піддається прогнозу. Тому, функції приналежності формуються на основі оптимізації матриць нечітких відношень із залученням експертів на попередньому етапі. Результатом є коефіцієнт небезпеки інфікування системи троянськими програмами, який порівнюється з деяким нормованим коефіцієнтом, заданим стратегією політики безпеки. Метод виявлення троянських програм може мати модифікації в частині активного та пасивного моніторингу подій [225, 228], зокрема, він може бути застосовним в режимі монітору [229] для проведення постійного моніторингу подій та дослідження файлового ЗПЗ. Отриманий результат РБС в одній з КС з використанням методу виявлення файлового ЗПЗ надасть можливість здійснити обмін його результатом з іншими ПМ РБС. Крім того, поєднання роботи програмних агентів, що здійснюють виявлення зловмисного програмного забезпечення в окремих комп'ютерних системах, відповідно до імплементованих в них методів на кроці 4 розробленого методу дозволить його застосування в РБС і впливатиме на її подальші дії.

Розроблений метод виявлення файлового ЗПЗ у локальних мережах базується на поєднанні двох методів [174, 381, 389], які здійснюють побудову поведінкової сигнатури [337] та її подальший аналіз на наявність файлового ЗПЗ і розбиття на блоки виконуваної програми та дослідження її на наявність поліморфного та метаморфного вірусу. Отримання результату підвищення достовірності виявлення згідно розробленого методу досягається залученням до процесу програмних модулів РБС, що дозволяє здійснювати більш детальний аналіз програмного коду.

Висновки до п'ятого розділу

Розроблені методи виявлення файлового ЗПЗ використовують поведінкові сигнатури і для їх представлення в РБС використовується база поведінкових сигнатур, яка попередньо наповнена зразками типів ЗПЗ [225, 228, 348, 362].

Поведінкові сигнатури формуються на основі матриць інцидентності поширення та деструктивних дій файлового ЗПЗ [366, 373], які уточнюються послідовностями викликів API-функцій.

Розроблений метод виявлення файлового зловмисного програмного забезпечення базується на основі динамічного формування поведінкової сигнатури шляхом відстеження викликів API [386]. Він може бути використаний для виявлення інших видів вірусних програм, зокрема нових версій існуючих вірусів. Метод включає формування сигнатури вірусної програми на основі трасування API-викликів, що дозволяє здійснити виявлення вірусної програми, яка представлена розробленою поведінковою сигнатурою з бази сигнатур. Поведінкова сигнатура включає критичні API-викликів за групами шкідливої активності та відображає частоту їх входження, а також характер взаємодії критичних API-функцій вірусної програми та описує взаємозв’язок між критичними API-функціями. Це надає можливість розмежувати вірусні програми від корисних застосунків не тільки за наявністю критичних API-викликів, але й за їх взаємодією між собою. Для здійснення виявлення використовується класифікація.

Для файлового ЗПЗ, яке використовує техніки заплутування свого коду, розроблено метод виявлення поліморфних [200] та метаморфних вірусів на основі аналізу функцій обфускації [174]. Особливістю методу є аналіз програмного об’єкту та його модифікованих версій, отриманих від різних ПМ РБС [226] і подальший аналіз на основі пошуку еквівалентних функціональних блоків [224]. Це дозволяє здійснити детальніший аналіз коду програмного об’єкту на наявність поліморфних та метаморфних вірусів.

Метод виявлення [386, 381, 305] файлового зловмисного програмного забезпечення в локальних комп’ютерних мережах, який полягає в поєднанні роботи програмних агентів, що здійснюють виявлення зловмисного програмного забезпечення в окремих комп’ютерних системах, відповідно до імплементованих в них методів: динамічного формування поведінкової сигнатури шляхом відстеження викликів прикладного програмного інтерфейсу, знаходження

поліморфного та метаморфного програмного коду, сканування виконуваних програм шляхом створення для них автономних процесів та відповідних програмних агентів в розподіленій системі, дав змогу покращити аналіз і підвищити достовірність виявлення зловмисного програмного забезпечення.

Основні результати розділу опубліковані у працях [174, 200, 224–226, 228, 229, 305, 337, 348, 362, 366, 372, 373, 377, 381, 386, 389, 382].

РОЗДІЛ 6
СТРУКТУРА, ОЦІНКА ДОСТОВІРНОСТІ ТА ЕФЕКТИВНОСТІ РОЗПОДІЛЕНОЇ БАГАТОРІВНЕВОЇ СИСТЕМИ ВИЯВЛЕННЯ ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖАХ

Вибір існуючих антивірусних мережних засобів здійснюється на основі результатів порівняння достовірності їх роботи на заздалегідь підготовлених тестових зразках. Ці тестові зразки повинні охоплювати всі типи ЗПЗ і використовуватись для перевірки на виявлення згідно плану проведення експерименту. Для залучення порівняння розробленої РБС виявлення ЗПЗ необхідною є її програмна (або апаратно-програмна) реалізація, яка крім мережної частини містила б на відповідних рівнях реалізовані методи виявлення певних типів ЗПЗ. Також, для розробленої РБС необхідно здійснити оцінку її ефективності, яка включатиме багато типових параметрів складних розподілених систем, з метою встановлення можливості її використання.

6.1. Програмна реалізація розподіленої системи виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах

Розроблене програмне забезпечення (далі Distributed Multilevel System) [384, 303, 304, 363] функціонує у вигляді однакових програмних модулів у комп'ютерних системах у локальній комп'ютерній мережі в автономному режимі і сукупно всі програмні модулі утворюють розподілену багаторівневу систему. Для запуску Distributed Multilevel System користувачу потрібно встановити програмне забезпечення в кожну комп'ютерну систему у локальній мережі. В кожній комп'ютерній системі необхідно виконати наступне:

1) запустити застосунок DistributedApp.exe;

2) якщо ввімкнено брандмауер Windows, дозволити застосунку

використовувати мережні інтерфейси;

3) налаштувати порт для комунікації між комп'ютерними системами (рис.6.1), який повинен бути однаковим для усіх застосунків у кожній комп'ютерній системі.

Розроблене програмне забезпечення складається з програмних модулів, які встановлені в комп'ютерних системах локальної комп'ютерної мережі, що функціонують автономно, вирішуючи поставлені завдання, та разом утворюють цілісну розподілену програмну систему, яка виконує завдання по виявленню мережного та файлового зловмисного програмного забезпечення на основі аналізу та порівняння поведінкових сигнатур, отриманих з різних комп'ютерних систем мережі. В кожному ПМ містяться підсистеми. Для загального функціонування РБС важливими є такі підсистеми: підсистема оцінки поширення ЗПЗ у КС [384], підсистема організації взаємозв'язку компонентів системи [304], підсистема для пошуку та визначення еквівалентних функціональних блоків у виконуваних програмах [303]. Оскільки вони визначають загальне функціонування системи, то розглянемо їх детальніше.

Розроблена підсистема РБС для оцінки [384] поширення ЗПЗ у локальних комп'ютерних мережах, є необхідною і використовується для прогнозування часу і напрямку розповсюдження ЗПЗ з врахуванням топології мережі та системного програмного забезпечення, встановленого в комп'ютерних системах. Її використання може здійснюватись адміністратором та автономно самою РБС. Після встановлення ПМ РБС в комп'ютерні системи цією підсистемою здійснюється первинна оцінка ймовірностей бути ураженими для всіх КС окремо і РБС в цілому.

Підсистема для пошуку та визначення еквівалентних функціональних блоків у виконуваних програмах [303] є необхідною для виявлення файлового ЗПЗ, яке містить поліморфний або метаморфний код. Ця підсистема реалізована на одному з підрівнів ПМ.

Оскільки виявлення ЗПЗ відбуватиметься у локальній комп'ютерній мережі, тоді представимо структурну розподіленість архітектури розробленої

системи схемою, зображеною на рис. 6.1.

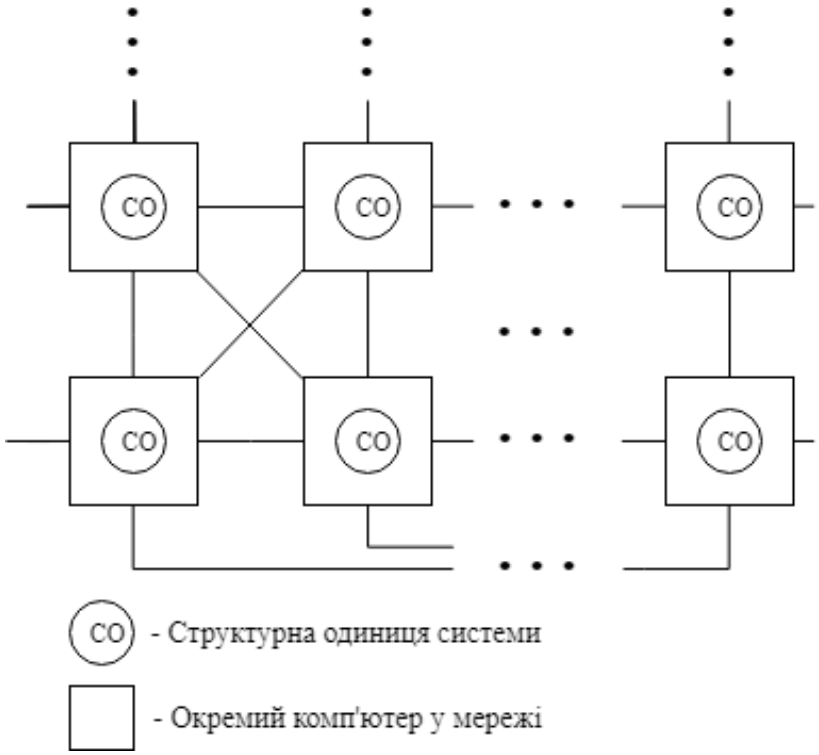


Рис. 6.1. Схема структурної розподіленості архітектури

Кожна одиниця системи в окремих комп’ютерах в мережі є автономною та повноцінною, з’єднаною з кожною іншою. Таким чином, жодна з них не виступає у ролі центрального керівного пункту прийняття рішень. Система здатна розширюватися підсистемами, що можуть бути приєднані до основної архітектури. Децентралізованість архітектури представлено узагальненою структурною схемою складових системи, яка зображена на рис. 6.2.

Узагальнена схема складових системи відображає реалізовану ідеологію децентралізованості та розподіленості у просторі. Прийняття рішень не здійснюється єдиним центром, проте виконується у внутрішніх рівнях ПМ та узгоджується між ними через інтерфейси прийняття рішень. Кожна окрема одиниця є автономною у прийнятті рішень. Описана взаємодія між внутрішнім та зовнішнім рівнем відображена на рис. 6.3.

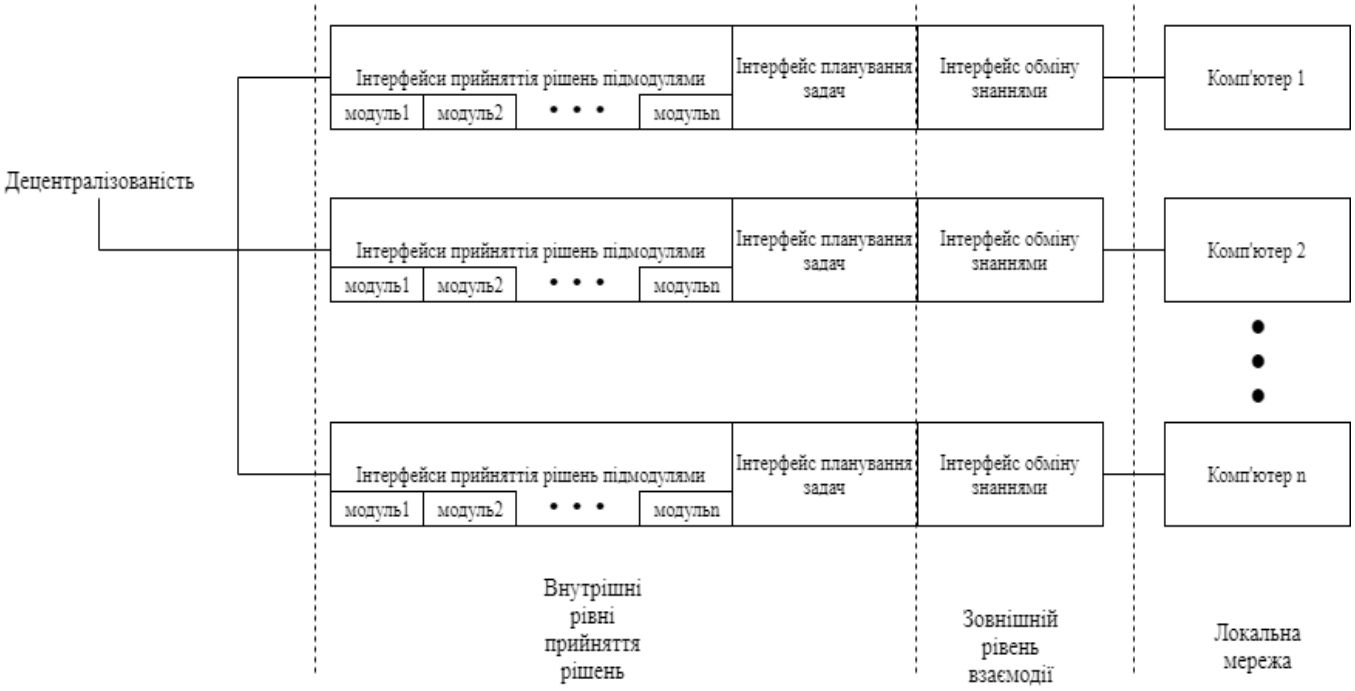


Рис. 6.2. Узагальнена схема складових системи

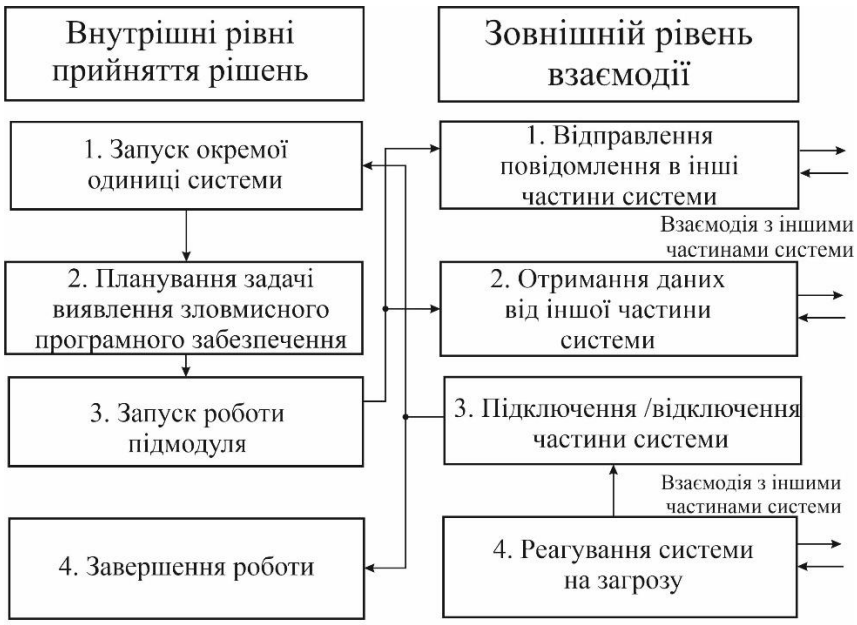


Рис. 6.3. Взаємодія між внутрішнім та зовнішнім рівнями

Частини системи взаємодіють між собою за допомогою повідомлень, структура яких представлена в табл. 6.1.

Таблиця 6.1

Узагальнена структура повідомлення взаємодії ПМ

Заголовок повідомлення		Тіло
ПМ-ініціатор повідомлення	Тип повідомлення	Дані повідомлення

Згідно представленої будови кожне повідомлення складається із заголовку та тіла. В якості заголовку повідомлення виступає ПМ-ініціатор повідомлення – ідентифікатор ПМ, який надіслав повідомлення. Як ідентифікатор ПМ використовується IP-адреса комп'ютерної системи, в якій цей модуль встановлений. Тіло повідомлення містить два поля:

1) тип повідомлення – ідентифікатор типу повідомлення;

2) дані – безпосередньо дані повідомлення.

Тип повідомлення використовується для отримання інформації про дані, що міститимуться безпосередньо у тілі повідомлення. Можливі типи повідомлень представлено в табл. 6.2. Залежно від типу повідомлення його тіло міститиме різні дані.

Таблиця 6.2

Типи повідомлень

№	Тип повідомлення	Опис типу
1	2	3
1	Модуль активовано	Повідомлення, що надсилається усім компонентам системи у випадку активації модуля в конкретній комп'ютерній системі. Виступає у якості прохання відгукнутися усім активним модулям у системі.
2	Модуль деактивовано	Повідомлення, що надсилається усім компонентам системи у випадку деактивації модуля в конкретній комп'ютерній системі
3	Повідомлення-вітання	Повідомлення, що надсилається тому модулю, який запросив відгукнутися усіх інших активних ПМ системи, з метою виявлення усіх активних ПМ системи.

Продовження таблиці 6.2

1	2	3
4	Зміна задачі	Повідомлення, що надсилається усім ПМ системи у випадку зміни виконуваної задачі конкретним ПМ.
5	Задача виконана	Повідомлення, що надсилається усім ПМ системи у випадку завершення роботи над виконуваною задачею конкретним ПМ.
6	Повідомлення-опитування	Повідомлення, що надсилається усім компонентам системи з метою отримання інформації про поточний стан кожної частини.
7	Повідомлення-стан	Повідомлення, що надсилається тому модулю, який запросив відгукнутися усіх решту активних ПМ системи, з метою отримання інформації про поточний стан кожної частини.
8	Виявлено загрозу	Повідомлення, що надсилається усім ПМ системи з метою інформування про виявлення загрози.

Кожен запис у базі даних має спеціальну мітку, залежно від типу даних, які він представляє:

1) звичайне “технічне” повідомлення;

2) попередження;

3) помилка у роботі певного підмодуля ПМ;

4) критичний стан ПМ;

5) повідомлення про результат виконання завдання.

Загальний алгоритм роботи реалізованої системи в конкретній КС зображено на рис. 6.4. Кожен ПМ позначено як “Defender Application” за назвою реалізованої системи.

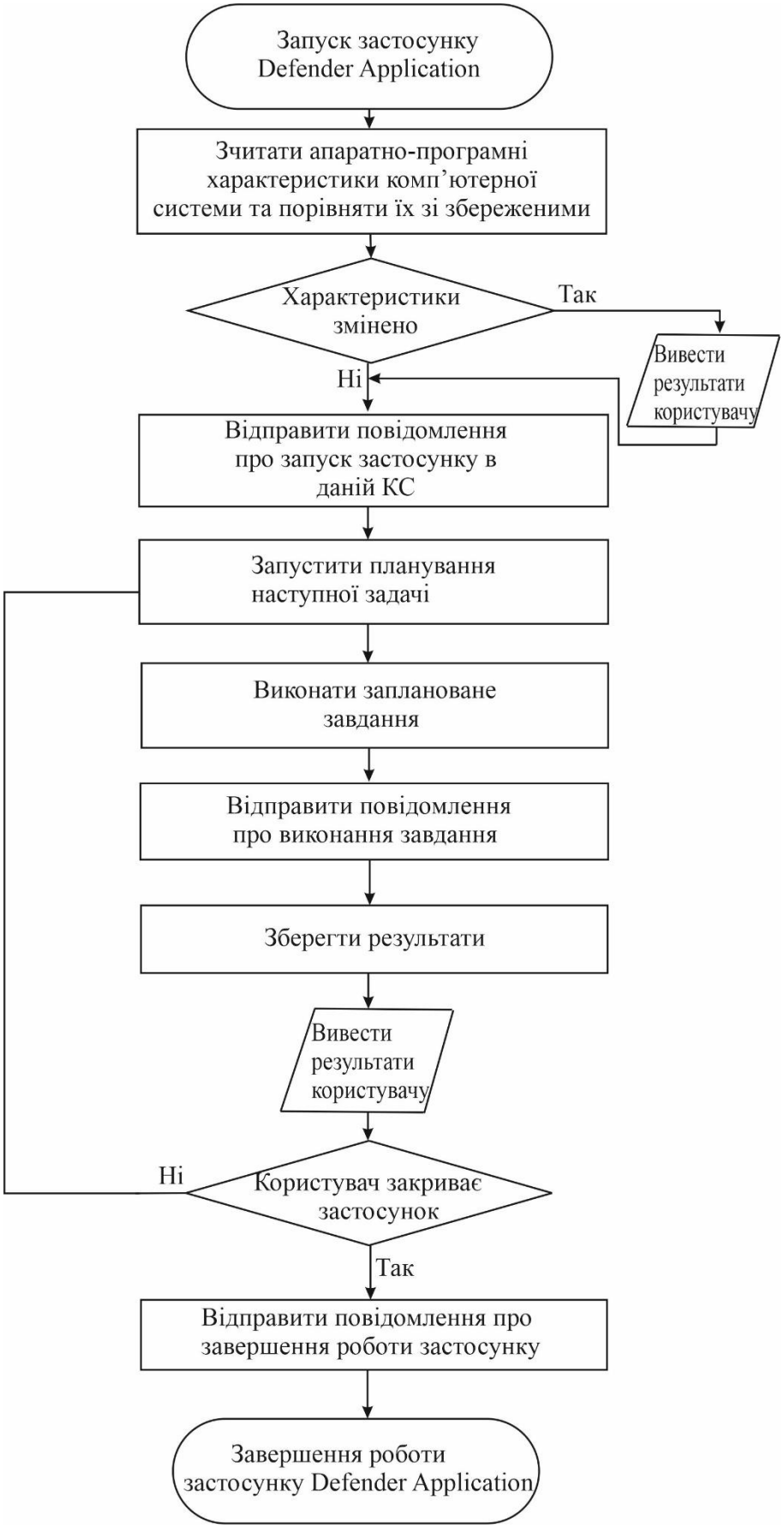


Рис. 6.4. Загальний алгоритм роботи розробленої системи

Перелік функціональних можливостей, якими володіє розроблена система з погляду користувача цієї системи відображено use-case діаграмою, яка зображена на рис. 6.5.

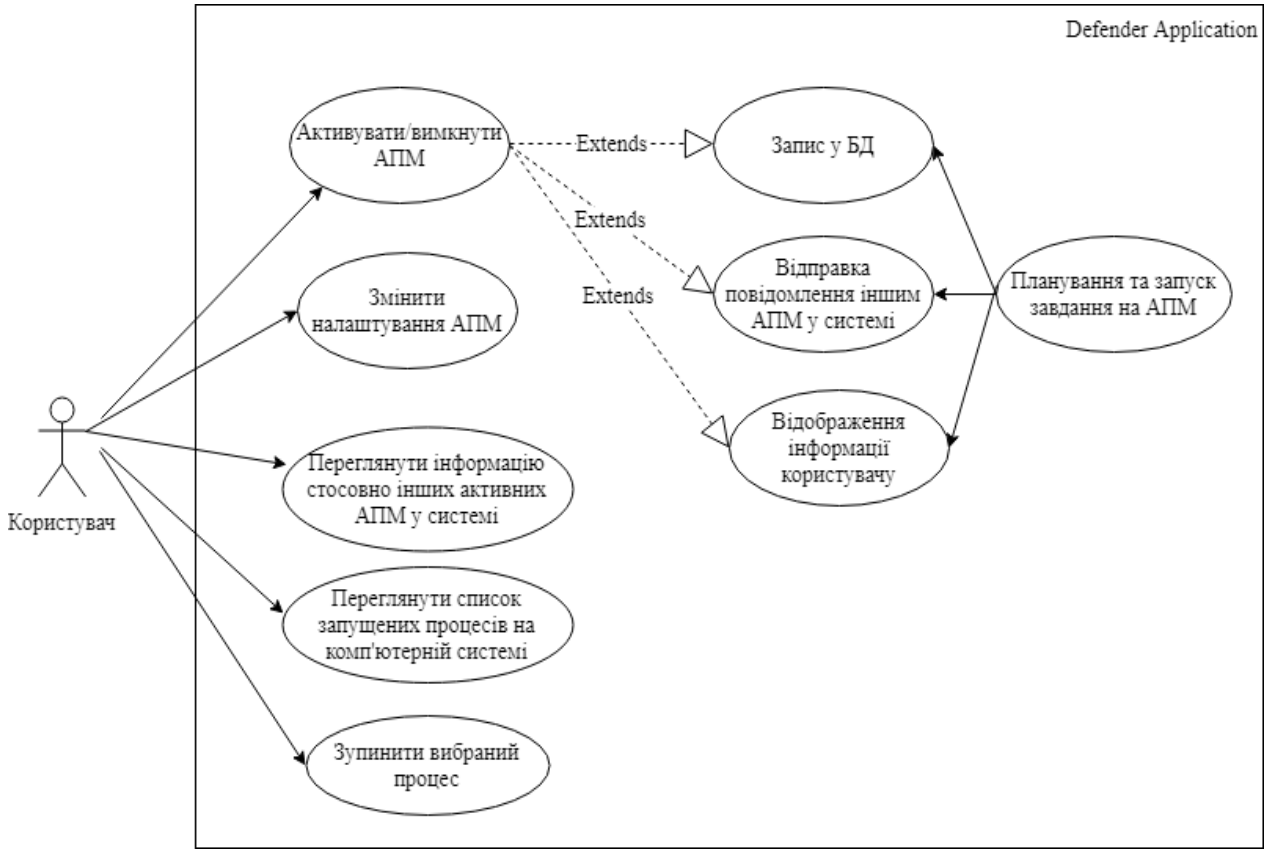


Рис. 6.5. Use-case діаграма розробленої системи

Взаємодія між користувачем та розробленою системою зображена на рис.6.6 у вигляді діаграми послідовностей.

В діаграмі послідовностей відображено, що кожен окремий модуль системи встановлений в конкретній комп'ютерній системі є автономним, тобто не потребує впливу ззовні для прийняття рішень стосовно планування завдань та обміну повідомленнями з іншими модулями у системі. Користувач може лише переглядати результати, змінювати деякі налаштування та використовувати додатковий функціонал. Наприклад, зупиняти запущені процеси.

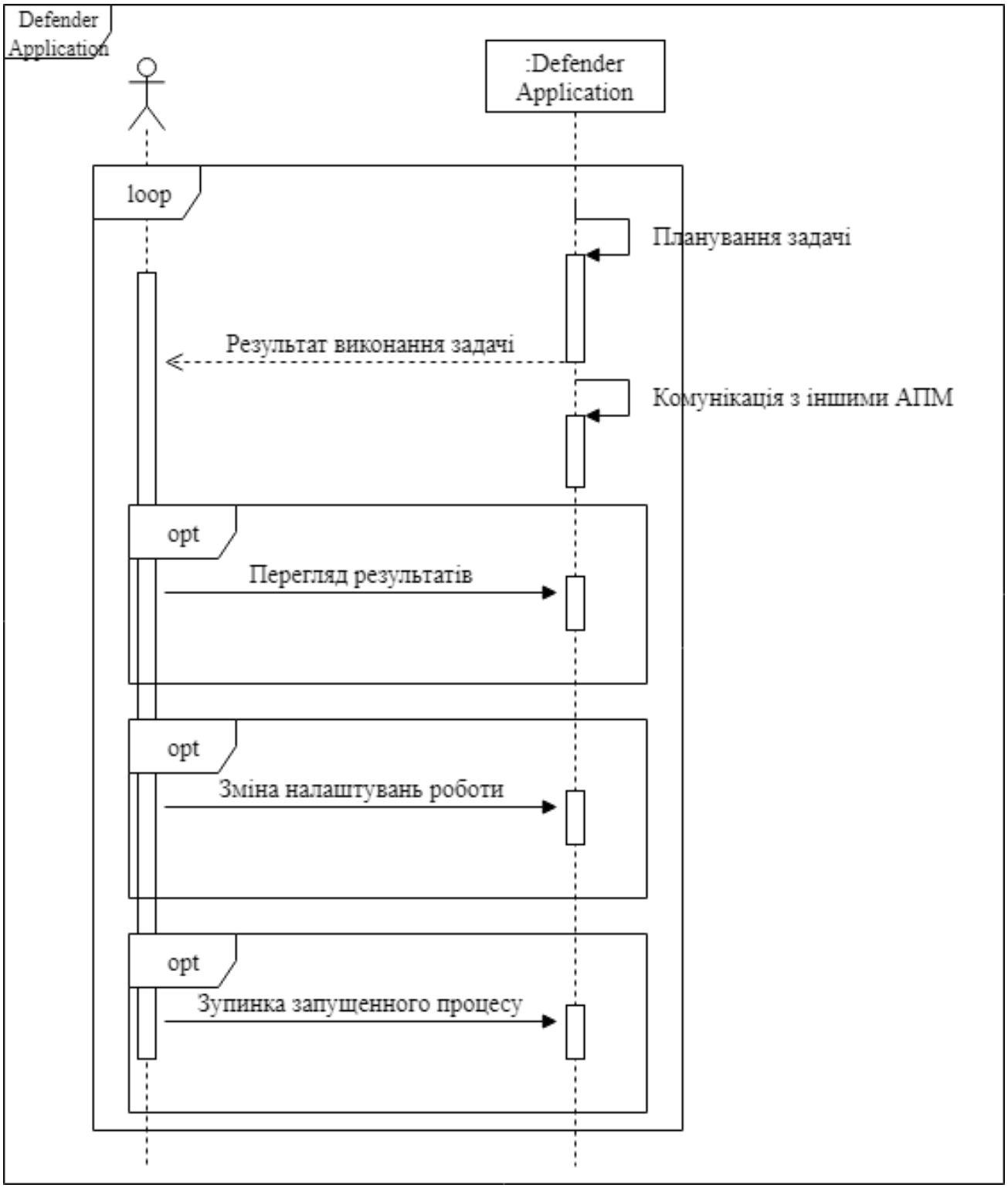


Рис. 6.6. Діаграма послідовностей розробленої системи

На рис. 6.7 зображено діаграму компонентів, яка відображає узагалальнену структурну схему реалізованого програмного забезпечення. Розроблений застосунок складається з декількох рівнів, які взаємодіють між собою. Інтерфейс

користувача та основний функціонал реалізовано окремими компонентами. Функціонал, що відповідає за обмін повідомленнями між ПМ у системі, інкапсульовано у компоненті «Data transmitting». Розроблені модулі використовують декілька пакетів бібліотеки Qt: QtGui, QtCore, QtNetwork.

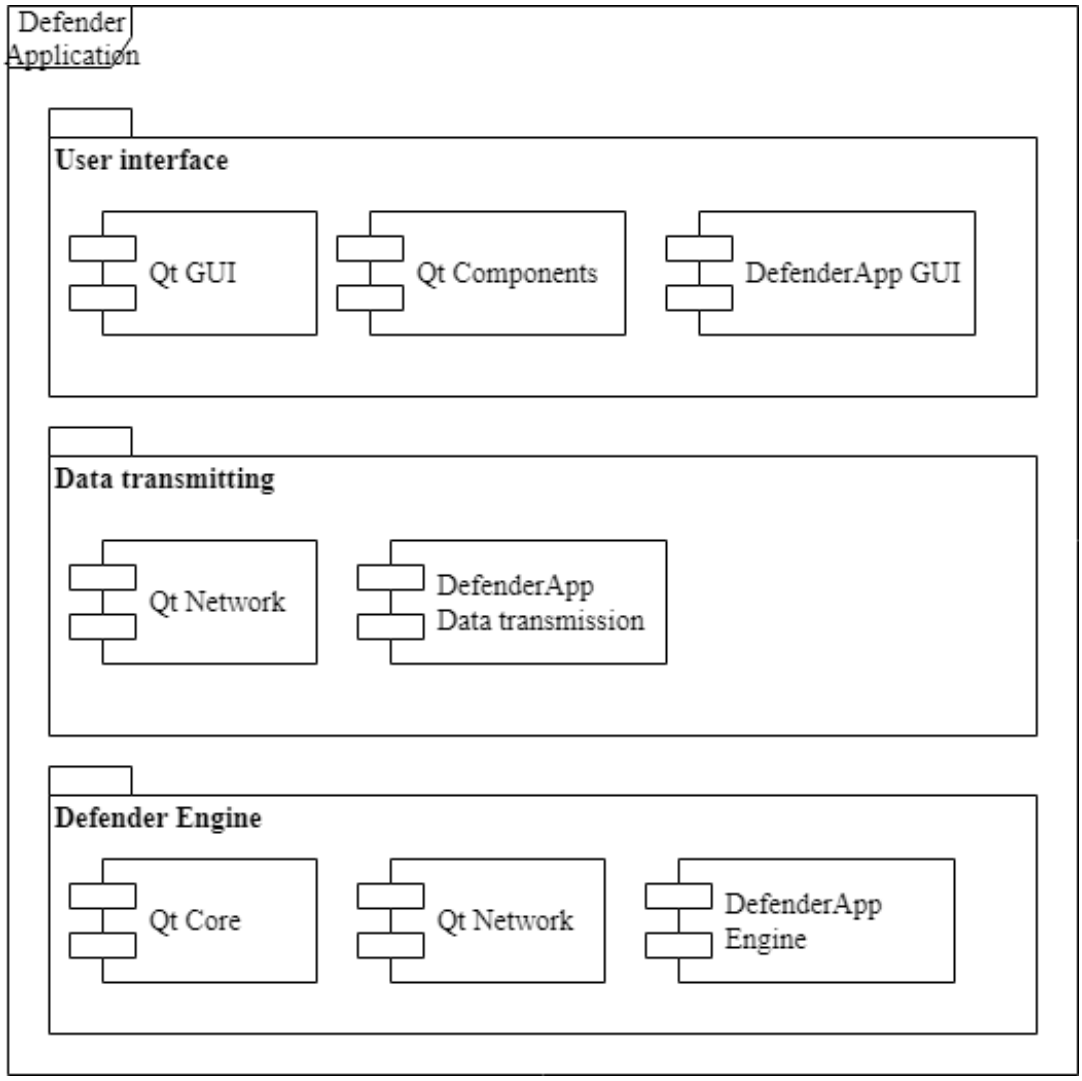


Рис. 6.7. Діаграма компонентів розробленої системи

Детальну будову кожного розробленого модуля представлено діаграмами класів, які зображено на рис. 6.8 – 6.10.

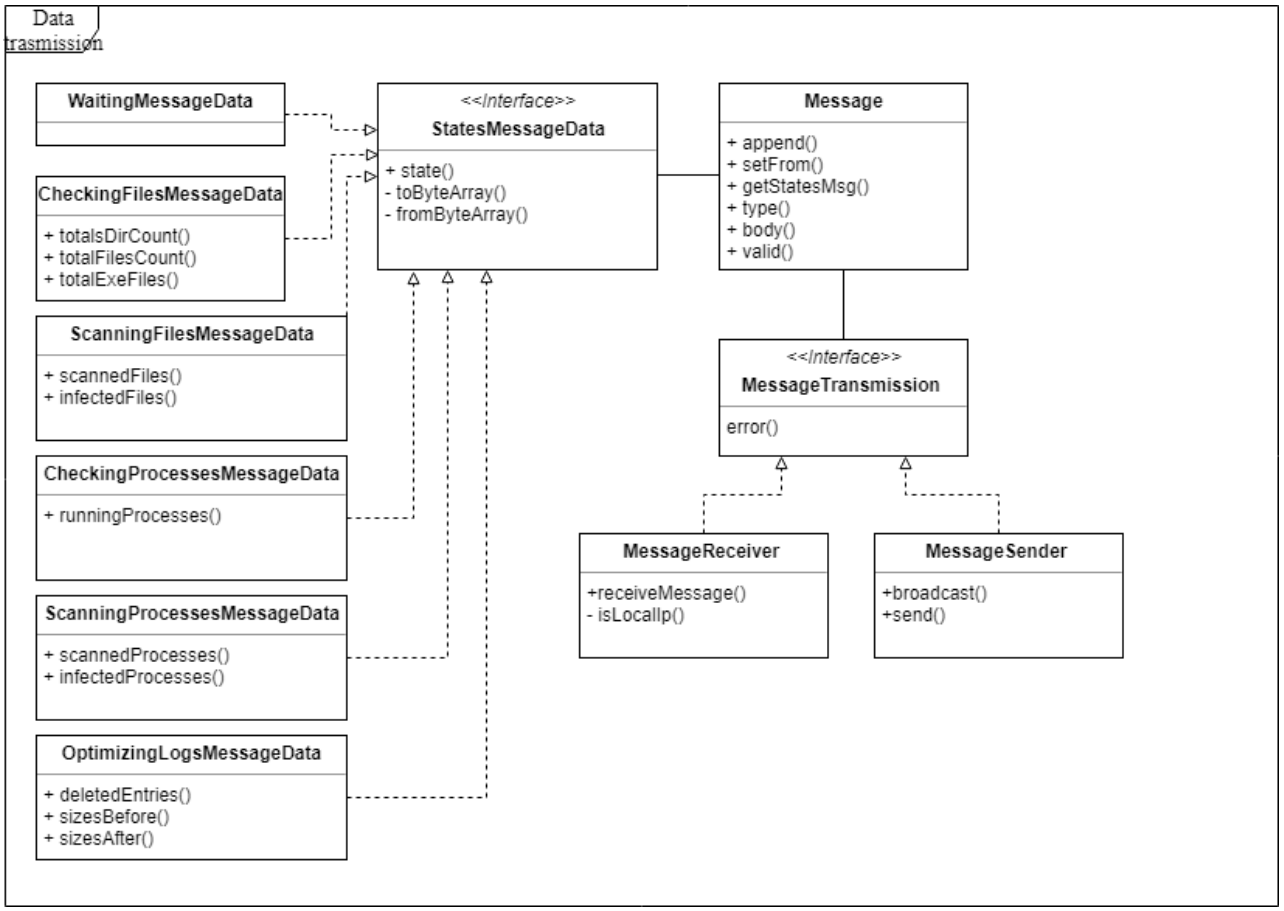


Рис. 6.8. Діаграма класів модуля DefenderApp DataTransmission

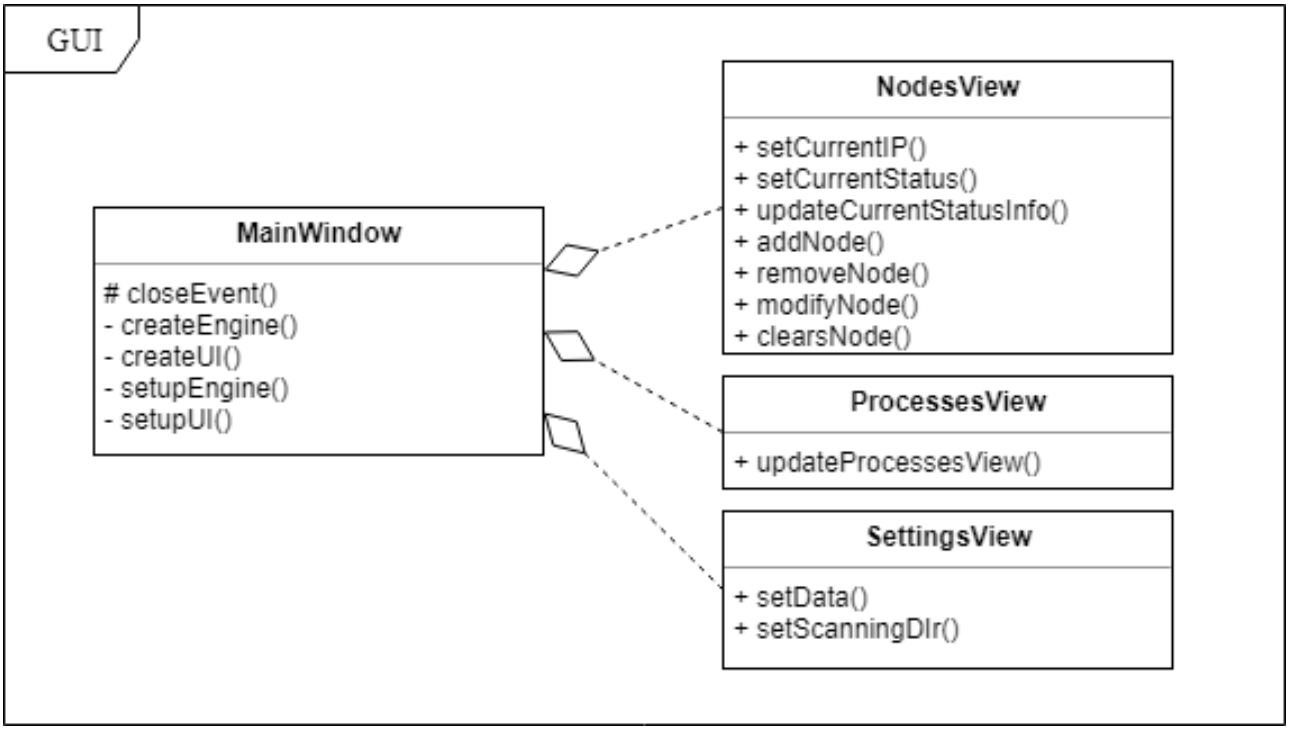


Рис. 6.9. Діаграма класів модуля DefenderApp GUI

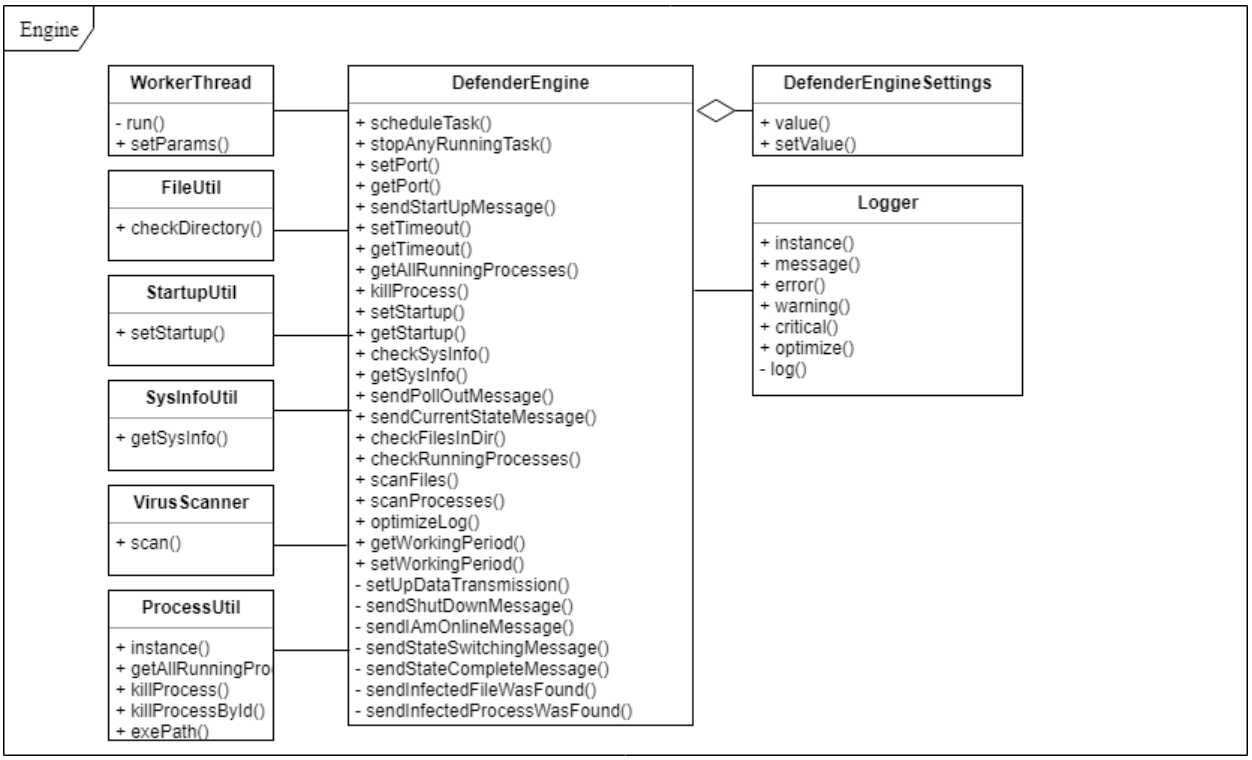


Рис. 6.10. Діаграма класів модуля DefenderApp Engine

Розроблена система функціонує на двох рівнях: внутрішньому та зовнішньому. Автономність кожної одиниці на внутрішньому рівні включає здатність прийняти рішення про виявлення спроби атаки зловмисним програмним забезпеченням та передачу повідомлення на зовнішній рівень. Таким чином, система здатна оперативно реагувати на отримані результати роботи.

Для програмної реалізації РБС виявлення ЗПЗ (далі Defender App) було вибрано технологію C++, яка дозволяє виконувати розробку застосунків на високому рівні і водночас мати низькорівневий доступ до пам’яті, що дозволило реалізувати ефективні алгоритми виявлення зловмисного програмного забезпечення.

Для збереження реакції та часу відгуку головного вікна інтерфейсу на дії користувача, заплановані завдання виконуються в окремому потоці, оскільки потенційно завдання може бути ресурсоємним.

Defender App планує наступні п’ять завдань:

1) перевірку файлів на жорстому диску, що полягає у збиранні статистичної інформації про файли на жорсткому диску, наприклад, кількості виконуваних файлів;

2) перевірку запущених процесів, що полягає у збиранні статистичної інформації про кількість запущених процесів в оперативній пам'яті;

3) сканування фалів на жорсткому диску, яке виконується за допомогою класу VirusScanner, що реалізує метод виявлення сигнатур вірусів та порівняння їх із базою сигнатур;

4) сканування запущених процесів, що виконується за допомогою класу VirusScanner, який реалізує метод виявлення сигнатур вірусів та порівняння їх із базою сигнатур;

5) оптимізація файлу-журналу, яка полягає у видаленні застарілої та неактуальної інформації.

Таким чином, розроблене програмне забезпечення РБС Distributed Multilevel System [303, 304, 384] дозволяє здійснювати його доповнення новими методами, реалізує зв'язуючу частину розподіленої системи і може бути використано для проведення експериментальних досліджень. Більш детальна інформація про розроблене програмне забезпечення РБС Distributed Multilevel System представлено в додатку Б.

Компоненти РБС представлені програмними модулями за потреби можуть формуватись як апаратно-програмні [381], причому на апаратний рівень може бути винесено частину функцій, які реалізовано програмно або його використано для захисту частини інформації компоненти РБС з метою підвищення рівня безпеки всієї системи через посилення безпеки її компонент. Тоді, РБС виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах в залежності від вимог політики безпеки організації (підприємства) може використовуватись в двох режимах. Перший режим її функціонування полягає у використанні тільки програмного забезпечення, яке виконує всі функції. Другий режим роботи розподіленої системи передбачає використання апаратно-програмних пристроїв для підтримки її роботи в більш захищеному

режимі в залежності від вимог політики безпеки організації (підприємства). Для цього кожна КС, в яку встановлено компоненту розробленої розподіленої системи, комплектується апаратно-програмним пристроєм. Тобто, всі КС локальної мережі в цьому режимі міститимуть такі однакові пристрої, які надають можливість програмним модулям системи здійснювати початкову ідентифікацію в КС та проводити обчислення рівнів безпеки компоненти і системи в цілому. В зв'язку з тим, що функціонування РБС передбачено в двох режимах, то метод взаємодії компонентів РБС використовується однаково в обох режимах, бо він регламентує порядок взаємодії компонентів розподіленої системи тільки для верхнього рівня програмної частини, тобто зовнішнього рівня. Наявність апаратно-програмних пристроїв для покращення захищеності розподіленої системи передбачає виконання ними тільки функцій внутрішнього рівня [380]. Тому вони не здійснюють взаємодію із програмною частиною системи зовнішнього рівня, яка є зв'язуючим програмним забезпеченням РБС. Детальніша архітектура апаратно-програмного пристрою представлена в додатку В.

Таким чином, РБС може бути реалізована у вигляді тільки програмного забезпечення [384, 303, 304] або програмного забезпечення, яке підтримує її цілісність в ЛКМ, та апаратно-програмних пристроїв [381], які встановлені в КС. Ці розроблені програмний засіб та програмно-апаратний засіб в складі з апаратно-програмними пристроями демонструють можливість реалізації запропонованих теоретичних основ та практики створення розподілених систем виявлення ЗПЗ у ЛКМ. Крім того, програмний засіб дозволяє використати його для порівняння з існуючими антивірусними засобами з метою перевірки ефективності виявлення ЗПЗ.

6.2. Постановка і проведення експериментальних досліджень застосування розробленої РБС

Експеримент з розробленою системою Distributed Multilevel System було проведено в локальній комп'ютерній мережі, в комп'ютерних системах якої було

встановлені модулі Defender App [172, 173, 385, 389]. Модулі працювали у фоновому режимі та фіксували результати виконання завдань у спеціальних файлах-журналах. Основна інформація про результати роботи розробленої системи отримується при виконанні аналізу та порівняння файлів-журналів декількох модулів Defender App. У журналах було зафіксовано результати роботи протягом одного робочого дня і наведено фрагменти з двох файлів-журналів (рис. 6.11, рис. 6.12, додаток Г).

```
WorkingState: 04.04.2018 10:47:05.412 10.0.2.15 started working on:
Scanning files in hard disk
WorkingState: 04.04.2018 10:47:05.412 10.0.2.15 completed working on
task: Scanning files in hard disk, results: scanned files = 16, infected
files = 0
WorkingState: 04.04.2018 10:57:05.412 10.0.2.15 started working on:
Waiting next task
WorkingState: 04.04.2018 11:06:05.632 Scheduling next task. Working
period = 30 minutes.
WorkingState: 04.04.2018 11:06:05.632 Start working on task: Optimizing
stored data
WorkingState: 04.04.2018 11:06:05.678 Complete work on task: Optimizing
stored data, results: deleted entries = 3, file size before = 8611, file
size after = 8325
```

Рис. 6.11. Фрагмент з файлу-журналу модуля DefenderApp в комп’ютерній системі 10.0.2.14

```
WorkingState: 04.04.2018 14:00:05.025 Start working on task: Checking
files in hard disk in C:/Users/8.1x64/AppData/Local
WorkingState: 04.04.2018 14:00:05.526 10.0.2.14 started working on:
Checking running processes
WorkingState: 04.04.2018 14:00:05.526 10.0.2.14 completed working
on task: Checking running processes, results: running processes = 40
WorkingState: 04.04.2018 14:00:05.526 10.0.2.14 started working on:
Waiting next task
WorkingState: 04.04.2018 14:00:05.760 Complete work on task: Checking
files in hard disk in C:/Users/8.1x64/AppData/Local, results: total dirs
count = 802, total files count = 835, total exe files = 16
WorkingState: 04.04.2018 14:10:05.041 Scheduling next task. Working
period = 10 minutes.
WorkingState: 04.04.2018 14:10:05.041 Start working on task: Checking
running processes
WorkingState: 04.04.2018 14:10:05.041 Complete work on task: Checking
running processes, results: running processes = 21
WorkingState: 04.04.2018 14:10:05.041 Start working on task: Scanning
running processes
WorkingState: 04.04.2018 14:10:05.041 Complete work on task: Scanning
running processes, results: scanned processes = 21, infected processes =
0
```

Рис. 6.12. Фрагмент з файлу-журналу модуля DefenderApp в комп’ютерній системі 10.0.2.15.

Здійснимо порядкову інтерпретацію даних з першого фрагменту журналу,

які були записані модулем Defender App в комп’ютерній системі 10.0.2.14. Розбиваючи перший запис з наведеного фрагменту на групи, з яких будуються усі повідомлення, отримуємо розбір представлений у табл. 6.5.

Таблиця 6.5

Розбір запису у файлі-журналі

Тип запису	Дата запису	Час запису	Інформація
WorkingState	04.04.2018	10:47:05.412	10.0.2.15 started working on: Scanning files in hard disk

Тип запису повідомляє, що інформація у повідомлені стосується того, що модуль розпочав або завершив роботу над певним завданням. Після типу запису міститься часова мітка, яка відповідає моменту запису повідомлення до журналу. Основна інформація повідомляє, що модуль Defender App в комп’ютерній системі з IP-адресою 10.0.2.15 розпочав роботу над скануванням файлів на жорсткому диску. Отримавши такого роду повідомлення, даний модуль Defender App очікує повідомлення від 10.0.2.15 комп’ютерної системи з результатами сканування.

Розбираючи наступні записи у журналі, можна побачити, що у результаті сканування в комп’ютерній системі 10.0.2.15 було просканово 16 файлів, з яких не було виявлено жодного інфікованого зловмисним програмним забезпеченням.

Подальші записи у журналі повідомляють, що після цього модуль Defender App перебував певний час у стані очікування наступного завдання, і о 11:06:05 розпочав задачу оптимізації збережених даних у власному журналі. Виконання цього завдання зайняло 46 мс, в результаті було видалено 3 (три) застарілих записи, що зменшило об’єм файлу-журналу приблизно на 300 Кб.

Подібним чином здійснимо аналіз другого фрагменту журналу модуля DefenderApp в комп’ютерній системі 10.0.2.15, який показує, що у період з 14:00:05 по 14:10:05 було виконано наступні завдання:

1) перевірено 802 директорії та 835 файлів на жорсткому диску, з яких 16 – виконуваних; операція зайняла 735 мс;

2) перевірено та просканово 21 запущенй процес, у результаті не виявлено жодного інфікованого.

Подібні повідомлення про хід виконання завдань кожним модулем Defender App розсилаються усім іншим активним на даний момент модулям у системі. Таким чином, кожна частина цілої системи володіє інформацією про те, яким завданням на поточний момент зайнятий кожен модуль у мережі. Це дозволяє усій системі відповідним чином реагувати на отримані результати.

Вдосконалення системи звітування можна реалізувати завдяки розробленій системі формування повідомлень, основними класами якої є Message та StatesMessageData. Розширені повідомлення можна легко інтегрувати до системи, додавши класи, що наслідують StatesMessageData, та реалізувавши методи, які б аналізували розширені отримані результати.

Структурно архітектуру Defender App було розділено на декілька модулів, що дозволило інкапсулювати функціонал різних підмодулів системи в окремі бібліотеки, які надають методи високорівневого інтерфейсу.

Було проведено експеримент із розробленою системою в локальній мережі. Результати роботи системи було збережено у файли-журнали. Отримані результати було детально проаналізовано та показано спосіб, за допомогою якого кожен модуль обмінюється знаннями та результатами з рештою модулів системи. Також, було виділено декілька напрямів подальшого вдосконалення системи: реалізація більш стійкого до втрат даних каналу обміну повідомленнями у мережі, розширення спектру виконуваних завдань із реалізацією додаткових методів виявлення зловмисного програмного забезпечення та покращення звітування про результати виконання завдань із зазначенням більш деталізованішої інформації.

На основі використання розробленої системи було проведено експерименти з виявлення бот-мереж та файлових вірусів, на основі запуску відповідних підсистем РБС.

Метою експериментів з виявлення бот-мереж була перевірка застосування методу виявлення, роботи класифікатору в структурі розподіленої системи та визначення залежності відсотку виявлених вузлів бот-мережі від їх представлення векторами та різними класифікаторами.

Для підготовки до проведення експериментів було здійснено конструювання 28 штучних бот-мереж та отримано коди відомих виявлених бот-мереж. Всі згенеровані бот-мережі згруповано за класами, в яких виділено 25 структурних елементів на трьох стадіях функціонування та 81 функцію. Слід відзначити, що не всі отримані таким чином бот-мережі містили повністю всі структурні елементи та функції.

Кожну функцію задано векторами зловмисних дій та атак з врахуванням варіацій та на основі яких побудовано зразки для їх включення в підкласи і класи. Експеримент проводився для класифікатора без додавання екземплярів створених бот-мереж та з ними, тобто здійснювалась перевірка без навчання класифікатору на створених зразках і з попереднім віднесенням зразків за класами. Другий варіант є необхідним для перевірки точності віднесення до класів тестових зразків, з яких ці класи були сформовані, оскільки при здійсненні моніторингу API-функцій можуть бути похибки.

А також, для встановлення величини різниці у двох випадках без попереднього навчання і з ним. Це необхідно, щоб перевірити залежність виявлення за векторами для кожного класу і загальну кількість виявлених вузлів бот-мереж та точність класифікації. Експеримент здійснювався в межах 19 комп’ютерних систем локальної мережі. Кожна з КС містила ПМ РБС. Інших антивірусних засобів в КС не встановлено. Спочатку було встановлено ПМ з класифікатором, в якому не було зразків створених бот-мереж. В одній з КС було розміщено командно-контролюючий центр, в трьох КС було розміщено контролюючі центри до кожного з яких було під’єднано по п’ять вузлів в кожній з п’ятнадцяти КС. Встановлення штучно згенерованих бот-мереж здійснювалось почергово. Після завершення експерименту з окремо згенерованою бот-мережею здійснювалось повне оновлення всіх КС і при цьому класифікатор залишався без

змін для кожного випадку. Тривалість моніторингу КС становила 96 годин для кожного екземпляру бот-мережі кожного з двох класифікаторів. Атака з вузлів бот-мережі не здійснювалась. Вузли бот-мережі працювали тільки в режимі контролю КС та підтримки структури бот-мережі через відправлені повідомлення. Таким чином, для ПМ РБС об'єктами дослідження були запущені в КС процеси і відповідно була здійснена побудова векторів за ними. Для проведення експерименту було обрані бот-мережі, які використовують стратегію отримання повного контролю в КС. Для здійснення експерименту засобами API-моніторингу в КС було отримано вектори, які почергово оброблено класифікатором ПМ. Результати обробки представлено в табл. 6.6.

Таблиця 6.6

Результати експерименту з виявлення бот-мереж

Показники експерименту, %	Отриманні значення для різних класів, %							Середні значення, %
	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	
1	2	3	4	5	6	7	8	9
$P_{1,1}$	90,74	84,29	73,66	86,30	94,04	94,18	96,60	89,44
$P_{1,2}$	75,93	63,57	60,22	70,32	68,77	67,60	69,36	67,71
$\lvert P_{1,1}-P_{1,2}\rvert$	14,81	20,72	13,44	15,98	25,27	26,58	27,24	21,73
$P_{2,1}$	85,80	83,57	72,58	85,39	98,88	93,92	96,60	88,42
$P_{2,2}$	74,69	63,57	59,14	70,32	67,37	66,58	67,66	66,80
$\lvert P_{2,1}-P_{2,2}\rvert$	11,11	20	13,44	15,07	31,57	27,34	28,94	21,62
$P_{3,1}$	92,11	84,21	71,93	89,47	90,53	88,42	93,68	87,72
$P_{3,2}$	76,32	57,89	63,16	64,91	71,58	54,74	75,79	65,89
$\lvert P_{3,1}-P_{3,2}\rvert$	15,79	26,32	8,77	24,56	18,95	33,68	17,89	21,83

Продовження таблиці 6.6

1	2	3	4	5	6	7	8	9
$P_{4,1}$	7,89	14,47	28,07	10,53	7,37	11,58	6,32	11,70
$P_{4,2}$	21,05	40,79	36,84	31,58	24,21	44,21	22,11	31,97
$\lvert P_{4,1}-P_{4,2}\rvert$	13,16	26,32	8,77	21,05	16,84	32,63	15,79	20,27
$P_{5,1}$	0	1,32	0	0	2,11	0	0	0,01
$P_{5,2}$	2,63	1,32	0	3,51	4,21	1,05	2,11	2,14
$\lvert P_{5,1}-P_{5,2}\rvert$	2,63	0	0	3,51	2,1	1,05	2,11	2,13

Експерименти передбачали визначення наступних показників ефективності виявлення вузлів бот-мереж для класів і підкласів Баєсівського класифікатору:

1) $P_{1,1}$ – відсоток векторів зловмисних дій та атак для вузлів бот-мереж, що належать даному класу відносно всіх тестових зразків, які система віднесла до цього класу з використанням попереднього навчання;

2) $P_{1,2}$ – аналогічно до п. 1, однак без використання попереднього навчання;

3) $P_{2,1}$ – відсоток векторів зловмисних дій та атак для вузлів бот-мереж, що належать даному підкласу класу відносно всіх тестових векторів, які система віднесла до цього підкласу класу в тестовій вибірці (ті, які були правильно віднесені до підкласів) з використанням попереднього навчання;

4) $P_{2,2}$ – аналогічно до п. 3, однак без використання попереднього навчання;

5) $P_{3,1}$ – відсоток правильно виявлених вузлів бот-мереж з використанням попереднього навчання;

6) $P_{3,2}$ – аналогічно до п. 5, однак без використання попереднього навчання;

7) $P_{4,1}$ – відсоток неправильно класифікованих вузлів бот-мереж, як

корисних додатків (помилка 1-го роду) з використанням попереднього навчання;

8) $P_{4,2}$ – аналогічно до п. 7, однак без використання попереднього навчання;

9) $P_{5,1}$ – відсоток неправильно класифікованих вузлів бот-мереж, як таких що є вузлами бот-мереж, але віднесені не до того класу (помилка 3 - го роду) з використанням попереднього навчання;

10) $P_{5,2}$ – аналогічно до п. 9, однак без використання попереднього навчання;

Результати оцінки ефективності виявлення програмного забезпечення вузлів бот-мереж на основі роботи двох класифікаторів для введених класів та підкласів у класифікаторі наведено у табл. 6.6.

В результаті проведення експерименту отримано віднесення до потрібного підкласу та класу отриманих на основі моніторингу векторів з точністю до 66 % для класифікатору без введених векторів 28 штучно згенерованих бот-мереж та 88 % для класифікатору, в який попередньо було додано вектори шляхом здійснення його навчання, зберігаючи в ньому шаблони попередніх наповнень. Перевірка здійснювалась окремо для класів, їх підкласів та в цілому для вузлів. Результати були усереднені і їх дисперсія відносно середнього значення становить 1 %. Різниця відхилення для двох класифікаторів по кожному класу, підкласу і вузлах бот-мереж в цілому складає 21,5 %. Відхилення між різницями відхилень для двох класифікаторів складає по кожному окремому класу, підкласу та вузлу бот-мережі становить менше 5%, що вказує на точність визначення в різних класах і підкласах. Це означає, що результат виявлення програмного забезпечення вузлів бот-мереж збігається в розрізі класів та підкласів для векторів зловмисних дій та атак.

Помилки 1-роду склали для першого і другого класифікаторів 11,7 % та 31,97 %, що пояснюється їх різним наповненням. Помилки 3-го роду – 0,01 % та 2,14 % відповідно, що пояснюється більш ширшим полем класифікації другого класифікатора через менший обсяг навчальної вибірки. В цілому результати

роботи класифікаторів показують можливість їх застосування для задач виявлення бот-мереж.

Результати розробленого класифікатора показують, що представлення зразків бот-мереж для різних класів і підкласів достатньо для ефективного виявлення бот-мереж. Результати експерименту показали, що точність виявлення бот-мереж досягає 88 %.

Розглянемо результати використання розробленої розподіленої багаторівневої системи на основі функціонування її підсистеми, що дозволяє виявляти файлові віруси [172, 174] за рахунок дослідження підозрілого коду, розміщеними в різних комп'ютерах локальної комп'ютерної мережі.

Для визначення ефективності розробленої системи було проведено ряд експериментів. Для цього було залучено локальну комп'ютерну мережу, що складалась з 20 комп'ютерних систем. Кожна комп'ютерна система була обладнана віртуальним середовищем на основі Qemu, яке задіювалось ПМ розробленої системи для дослідження поведінки ймовірно зловмисних програм і отримання викликів API-функцій. Дослідження виконуваних програм здійснювалось на трьох етапах їх функціонування: потрапляння в КС, активізації та виконання закладених функцій. Кожен ПМ використовував базу поведінкових моделей файлового ЗПЗ на її різних етапах функціонування. Для розрахунку достовірності виявлення файлового ЗПЗ був проведений експеримент з різними типами файлового ЗПЗ: файлові віруси, поліморфні віруси, метаморфні віруси, троянські програми. Було згенеровано 600 програмних об'єктів з функціональним навантаженням чотирьох розглядуваних типів файлового ЗПЗ по 150 кожного. В досліджувану підмножину файлових вірусів було включено лише ті, які не містили поліморфного або метаморфного коду і не були троянськими програмами. Для отримання тестових зразків поліморфних та метаморфних вірусів [174] було використано генератори NGVCK, PS-MPC, VCL32 та G2. Всі метаморфні варіації вірусів, які створювались за допомогою цих генераторів були скомпільовані з опціями anti-debugging та anti-emulation. Кожна із згенерованих варіацій метаморфних вірусів застосовувала основні

техніки заплутування коду: вставку сміттєвих команд, використання еквівалентних інструкцій та переміщування блоків інструкцій.

Сигнатури згенерованих вірусів відсутні в базах сигнатур АПЗ та розробленої системи. Всі програмні об’єкти було поділено на групи для задання способу їх потрапляння в КС, щоб врахувати усі можливі шляхи проникнення в КС:

1) програмні об’єкти скопійовані на жорсткий диск кожної КС;

2) програмні об’єкти завантажені на флеш-носії і підключені до кожної КС;

3) програмні об’єкти завантажені на попередньо створений web-сайт;

4) програмні об’єкти архівовані та відправлені на попередньо створені електронні адреси;

5) програмні об’єкти завантажені на попередньо створений ftp-сервери всіх КС.

Запуск на виконання згенерованих програмних об’єктів був здійснений спеціальною програмою, яка встановлена в кожну КС і запустила по одному програмному об’єкту із ЗПЗ в кожній КС одночасно. Потім все розпочиналось знову і вибирався інший програмний об’єкт. Запуск корисних програм в усіх КС не виконувався. Після ввімкнення всіх КС завантажувались ОС та всі програми, які потрібні і прописані для автоматичного запуску. Всі КС містили однакове апаратне та програмне забезпечення.

Результати проведеного експерименту та оцінки достовірності виявлення ЗПЗ розподіленою системою Distributed Multilevel System представлено в табл. 6.7 та зображено діаграмою на рис. 6.13. Достовірність виявлення за чотирма групами файлового ЗПЗ відображає приблизно однакове налаштування РБС до виявлення різних типів ЗПЗ. Це дозволяє її застосування в експериментальних дослідженнях для порівняння з існуючими АПЗ. Крім того, за результатами проведено експерименту було також встановлено кількість ПМ, які залучались для дослідження протягом всього експерименту, та кількість ПМ, які були заблоковані іншими ПМ розробленої системи, під час виявлення. Це підтверджує застосування інших компонентів розподіленої системи в процесі

виявлення ЗПЗ окремим ПМ.

Таблиця 6.7

Результати експерименту для ЗПЗ

Програмні об'єкти з наявним в них ЗПЗ		Кількість програм, виявлених як підозрілі	Відсоток виявлення, %	Кількість ПМ, які залучались для дослідження протягом всього експерименту	Кількість ПМ, які були заблоковані іншими ПМ розробленої системи, під час виявлення
Файлові віруси	150	146	97,3	0	2
Поліморфні віруси	150	144	96,0	57	7
Метаморфні віруси	150	138	92,0	24	3
Троянські програми	150	142	94,7	2	5
Разом	600	570	95,0	83	17
Середні значення				20,75	5,7

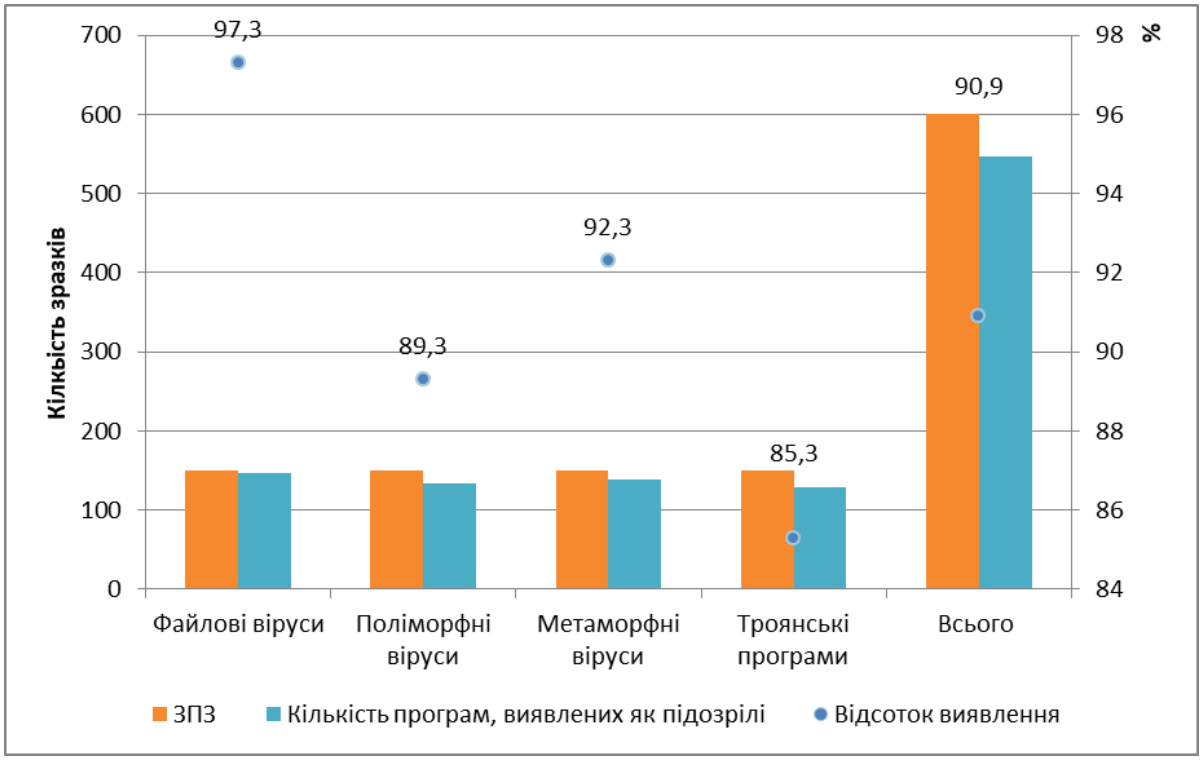


Рис. 6.13. Результати експерименту

Для проведення порівняльного аналізу з розробленою РБС було обрано наступні відомі антивірусні (хостові) засоби: ESET Smart Security (версія 10.1.204.0), Avast (версія 17.5.2303), Comodo Antivirus (версія 8.2.0.4674), Kaspersky (версія 17.0.0.61), McAfee Internet Security (версія 10.1.0), Dr.Web (версія 11.0), Microsoft Security Essentials (версія 4.11.15063.446), Avira Antivirus (версія 10.0). Також, було протестовано шість мережних антивірусних засобів: Symantec Endpoint Protection (версія 14.2.1), Panda Endpoint Protection (версія 8.42.00), Malwarebytes Endpoint Security (версія 3.8.3), McAfee Endpoint Protection Suite (версія 10.6.0), AVG Internet Security Business Edition (версія 19.1.2360). Проведений розрахунок на основі формул ROC-аналізу достовірності виявлення відомими хостовими та мережними АПЗ на згенерованому наборі ЗПЗ представлено відповідними діаграмами на рис. 6.14 та рис. 6.15. В них, зокрема, відображено порівняльний аналіз розробленої системи Distributed Multilevel System з існуючими АПЗ стосовно помилок першого роду (хибні спрацювання) та помилок другого роду (хибно-негативні результати).

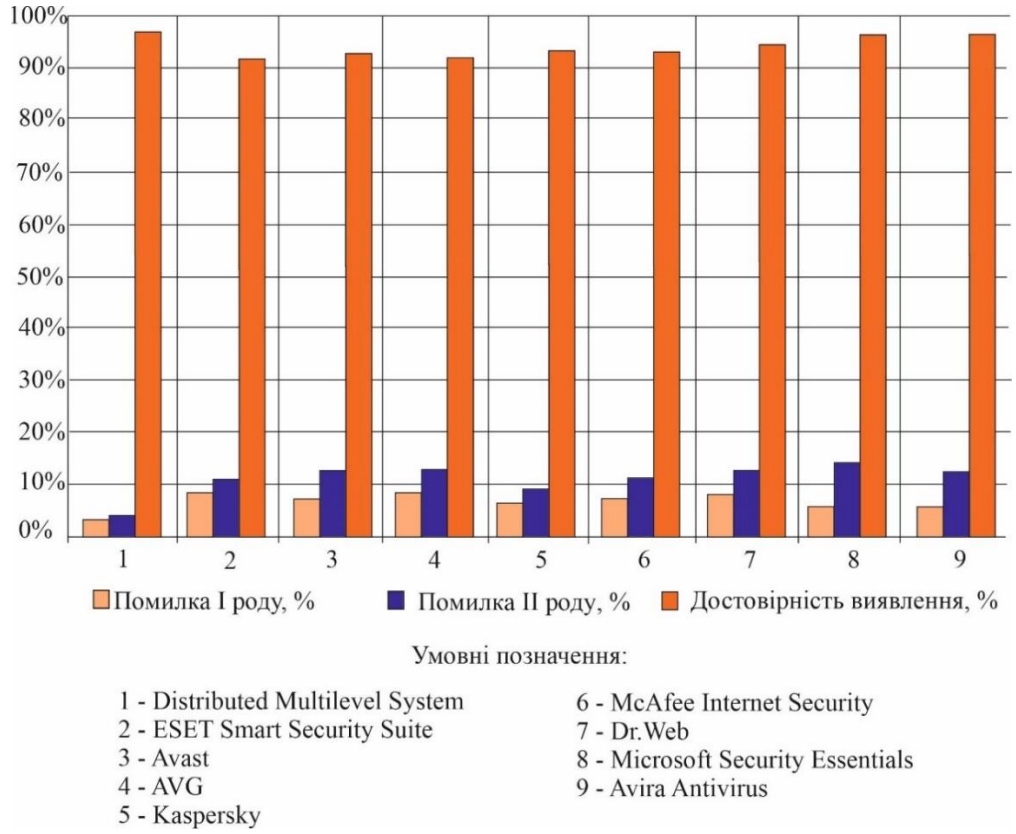


Рис. 6.14. Порівняльний аналіз розробленої системи Distributed Multilevel System з існуючими хостовими АПЗ

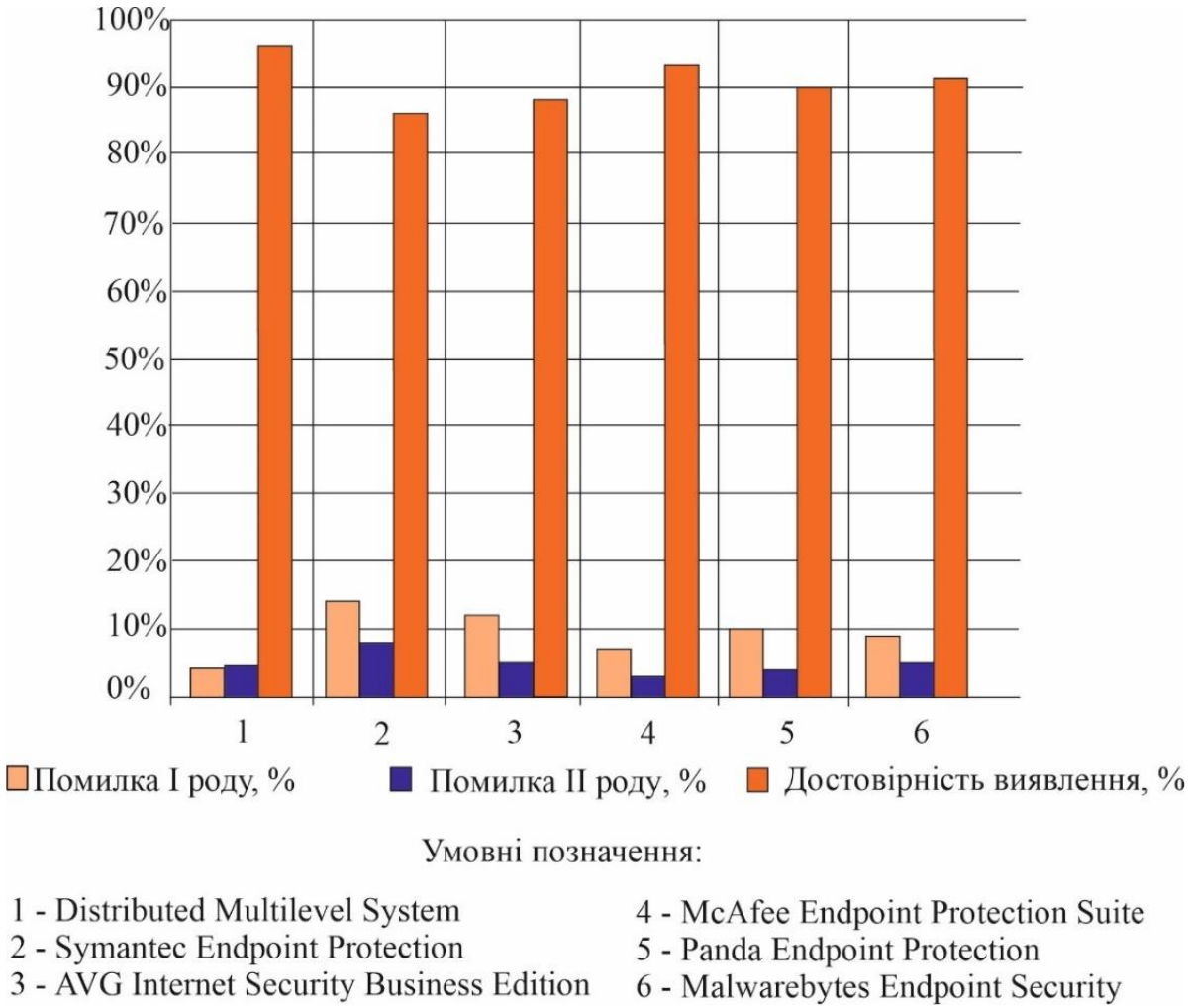


Рис. 6.15. Порівняльний аналіз розробленої системи Distributed Multilevel System з існуючими мережними АЗ

Результати експериментальних досліджень з використанням розробленої системи Distributed Multilevel System [363, 381, 385, 389] підтверджують правильність наукових положень розроблених методів та ефективність архітектури РБС, оскільки її впровадження підвищує достовірність виявлення на 5-12 % в мережному представленні порівняно з хостовим, та на 2-4 % у порівнянні з існуючими мережними АПЗ виявлення файлового ЗПЗ.

Результати проведених експериментальних досліджень показують, що рівень достовірності виявлення ЗПЗ при застосуванні розробленою системою Distributed Multilevel System складає близько 95 %, що на 2-4 % вище порівняно з існуючими антивірусними програмними засобами (рис. 6.14 та рис. 6.15).

Застосування розробленої системи Distributed Multilevel System дозволяє досягти зниження рівня помилок першого роду до 5 %, що на 3-4 % нижче

порівняно з існуючими мережними антивірусними програмними засобами (рис. 6.14 та рис. 6.15).

Таким чином, результати використання розробленої РБС Distributed Multilevel System [363, 381, 385, 389] та її підсистем для виявлення певних типів ЗПЗ у КС локальних комп'ютерних мереж є достатніми порівняно з аналогічними мережними АПЗ і підтверджують ефективність запропонованих теоретичних моделей та розроблених методів.

6.3. Оцінки ефективності та достовірності мережних засобів виявлення зловмисного програмного забезпечення

Здійснення оцінки ефективності розробленої РБС проведемо порівняно з її використання в хостовому представленні одним програмним модулем та системою, в якій більше одного програмного модуля. Таке порівняння дозволить встановити додаткові витрати на операції обміну між програмними модулями та витратами на залучення додаткових ресурсів КС. Визначення загальної оцінки ефективності розробленої РБС виявлення ЗПЗ у КС локальної мережі здійснимо на основі її суттєвих характеристик та показників, які впливають на її функціонування, завантаженість ресурсів КС та трафіку в мережі. До критеріїв ефективності для розробленої РБС віднесемо такі: витрати часу на здійснення виявлення хостом; витрати часу на здійснення виявлення всією РБС; оперативність в прийнятті рішень; ресурсоспоживання; достовірність виявлення.

Оскільки, РБС при прийнятті рішень використовує інформацію з ПМ окремо, а також залучає групу ПМ для прийняття рішень, то здійснимо визначення ефективності її ПМ окремо. Визначимо час, який витрачається на здійснення виявлення ЗПЗ у КС локальної мережі, за формулою (6.1):

$$t_{\text{ПМ}} = \sum_{i=1}^{10} t_i \, , \tag{6.1}$$

де $t_{\text{ПМ}}$ – час роботи окремого ПМ РБС; t_1 – час перебування ПМ в стані «1»; t_2 -

час витрачений на перевірку всіх виконуваних програм в окремій КС програмним модулем РБС без залучення інших ПМ, тобто перебування ПМ в стані «2»; t_3 – час перевірки процесів та мережних пакетів одним ПМ у КС, тобто передування в стані «3»; t_4 – час витрачений на перевірку в окремій КС програмним модулем РБС без залучення решти ПМ, тобто перебування ПМ у стані «4»; t_5 – час витрачений на перевірку вибраних виконуваних програм в окремій КС програмним модулем РБС із залученням декількох ПМ, тобто перебування ПМ в стані «5»; t_6 – час перевірки процесів та мережних пакетів одним ПМ у КС із залученням декількох ПМ, тобто перебування в стані «6»; t_7 – час витрачений на перевірку в окремій КС програмним модулем РБС із залучення декількох ПМ, тобто перебування ПМ в стані «7»; t_8 – час витрачений на здійснення оптимізації інформації в окремій КС програмним модулем РБС без залучення решти ПМ, тобто перебування ПМ в стані «8»; t_9 – час витрачений на відправку пакетів із завданнями для декількох ПМ РБС; t_{10} – час витрачений на обробку завдань від інших ПМ РБС. Якщо ПМ перебував в стані «1» постійно, тоді час витрачений на його функціонування збігається з часом перебування в стані «1». Час перебування ПМ в станах «1» – «4» та «8» відповідає одноосібній роботі ПМ РБС.

Співвідношення, яке визначається за формулою (6.2), встановлює частку часу p_1 роботи ПМ одноосібно в складі РБС:

$$p_1 = \frac{\sum_{i=1}^{4} t_i + t_8}{t_{\text{ПМ}}}. \quad (6.2)$$

Якщо ПМ не здійснював зв’язку з іншими ПМ РБС, тоді $p_1 = 1$. В протилежному випадку буде зафіксована частка часу, яка дозволить встановити час взаємодії ПМ з іншими ПМ РБС. Аналогічно введемо частку часу p_2 роботи ПМ на залучення інших ПМ РБС, яку визначимо за формулою (6.3):

$$p_2 = \frac{\sum_{i=5}^{7} t_i + t_9}{t_{\text{ПМ}}}. \quad (6.3)$$

А також, введемо частку часу p_3 на обробку запитів від інших ПМ РБС і визначимо її за формулою (6.4):

$$p_3 = \frac{t_{10}}{t_{\text{ПМ}}}. \quad (6.4)$$

Час витрачений на обробку запитів від інших ПМ РБС включає тривалість передачі одиниці обсягу даних для окремого ПМ з решти ПМ мережі, тривалість дослідження програмного коду ймовірного ЗПЗ та тривалість відправлених решті ПМ в мережі результатів. Тому, введемо для цих показників відповідний час, позначивши $t_{10,1}$ – тривалість передачі, $t_{10,2}$ – тривалість дослідження, $t_{10,3}$ – тривалість відправлення результатів. Тоді, $t_{10} = t_{10,1} + t_{10,2} + t_{10,3}$. В результаті витрати часу на здіснення процесу дослідження програмних об'єктів всією РБС отримуємо за формулою (6.5):

$$t_{\text{РБС}} = \sum_{i=1, i\neq j}^{n} a_i * t_{i,10} + t_k, \quad (6.5)$$

де $t_{\text{РБС}}$ – витрати часу всією РБС на вирішення завдань, які виникають в одному з ПМ; a_i – бінарні коефіцієнти, які вказують на залучення/не залучення i – го ПМ для вирішення завдань j – го ПМ; $t_{i,10}$ – витрати часу i – го ПМ для вирішення завдань j – го ПМ; t_k - час витрачений j – им ПМ для вирішення завдання в k – му стані; n – кількість ПМ у РБС. В результаті, досягнення ефективності роботи всієї РБС порівняно з одним ПМ полягає у визначенні відношення частки між ними, яке представлено формулою (6.6):

$$\frac{t_{\text{РБС}}}{t_k} = 1 + \frac{\sum_{i=1, i\neq j}^{n} a_i * t_{i,10}}{t_k}, \quad (6.6)$$

де t_k – час витрачений j – им ПМ для вирішення завдання в k – му стані; n – кількість ПМ у РБС.

Критерій ефективності полягає в мінімізації величини $k_e = \frac{t_{\text{РБС}}}{t_k}$. Якщо величина в чисельнику дорівнює нулеві, тобто інші ПМ РБС не залучались до

виконання завданння одного з ПМ, і при цьому завдання виконано, то результат є найефективнішим. Якщо ж величина в чисельнику відмінна від нуля, тобто залучався для виконання завданнь одного з ПМ хоча б один з ПМ РБС додатково, то тоді порівняння таких значень вказує на ефективність роботи РБС при вирішенні завдань як певного типу, що вимагає перебування в певному стані ПМ, так і кількості залучених ПМ. Таким чином, показник ефективності РБС за часом визначається за формулою (6.6).

Оперативність в прийнятті рішень розробленою РБС полягає в мінімізації часу від моменту виявлення ЗПЗ до його блокування не тільки в одній КС, в якій було виявлено, але і в усіх інших КС, в яких необхідне дослідження на наявність такого зловмисного програмного коду. Крім того, оперативність включатиме мінімізацію часу вимкнення РБС ураженої КС, в якій міститься ПМ РБС. Показники оперативності $p_{o,1}$, $p_{o,2}$ визначаються за формулами (6.7):

$$p_{o,1} = min\left|max\left|a_i * t_{i,10}\right| + t_k\right|, \qquad (6.7)$$
$$p_{o,2} = min\left|t_{j,11} + max\left|t_{i,12}\right|\right|,$$

де a_i – бінарні коефіцієнти, які вказують на залучення/не залучення i – го ПМ для вирішення завдань j – го ПМ; $t_{i,10}$ – витрати часу i – го ПМ для вирішення завдань j – го ПМ; t_k - час витрачений j – им ПМ для розв'язання завдання в k – му стані; $t_{j,11}$ – витрати часу j – го ПМ для вимкнення j – ої КС; $t_{i,12}$ – витрати часу i – им ПМ для прийняття рішення про вимкнення j - го ПМ. Враховуючи частки часу на обробку запитів, відношення ефективності за часом всієї РБС, показники оперативності за часом, показник ефективності витрат часу k_e визначимо за формулою (6.8):

$$k_e = (p_1 + p_2 + p_3) * \frac{t_k}{t_{\text{РБС}}} * \frac{p_{o,1}}{|max|a_i * t_{i,10}| + t_k|} * \frac{p_{o,2}}{|t_{j,11} + max|t_{i,12}||}, \qquad (6.8)$$

де якщо $t_{j,11} = 0$ та $t_{i,12} = 0$ або $a_i * t_{i,10} = 0$ та $t_k = 0$, тоді відповідні множникп, в яких вони присутні задаються такими, що дорівнюють одиниці.

Ресурсоспоживання окремою КС та всіма КС у локальній комп'ютерній мережі, в які встановлені ПМ РБС, залежить від залучених ресурсів для дослідження програмного коду ймовірно зловмисного ПЗ у КС. Такими характеристиками є середня тривалість емуляції j-ї інструкції і-тої КС мережі та прогнозована кількість при обробці завдань за час роботи ПМ в КС. Ресурсоспоживання r_e визначимо на основі середніх обсягів витраченої оперативної пам'яті, завантаженості процесора для виконання процесів системи виявлення ЗПЗ та середній обсяг даних, які передані каналами мережі.

Достовірність виявлення розраховуємо за формулами із загальноприйнятої стандартної методики ROC - аналізу. Для визначення показника, що використовуватиметься при розрахунку загальної ефективності, врахуємо час, що витрачено на помилки першого, другого та третього роду, який визначимо за формулою (6.9):

$$t_{d,\text{РБС}} = t_{d,1} + t_{d,2} + t_{d,3}, \tag{6.9}$$

де $t_{d,\text{РБС}}$ – час витрачений на помилки при виявленні; $t_{d,1}$ – час витрачений на помилки першого роду; $t_{d,2}$ – час витрачений на помилки другого роду; $t_{d,3}$ – час витрачений на помилки третього роду.

Також, потрібно врахувати виникнення цих помилок в різних КС при дослідженні одного і того ж програмного об'єкту. Показник достовірності полягатиме в мінімізації величини p_d, яка визначається за формулою (6.10):

$$p_d = min \left|1 - \frac{t_{d,\text{РБС}}}{t_{\text{РБС}}}\right|. \tag{6.10}$$

Визначимо показник ефективності витрат часу на здійснення процесу виявлення ЗПЗ у локальних комп'ютерних мережах за формулою (6.11) наступним чином: унормуємо показники p_d, k_e, r_e та знайдемо їх добуток.

$$E = p_{d,norm} * k_{e,norm} * r_{e,norm}, \tag{6.11}$$

де $p_{d,norm}$ – унормований показник достовірності виявлення; $k_{e,norm}$ – унормований показник ефективності витрат часу; $r_{e,norm}$ – унормований показник ефективності ресурсоспоживання.

Отже, дослідження характеристик РБС виявлення ЗПЗ надало можливість розробити методику визначення ефективності мережних АПЗ. Отримана методика дозволить здійснити порівняння ефективності розробленої РБС.

В ході проведених експериментів було отримано наступні показники: показник ефективності витрат часу k_e; показники оперативності $p_{o,1}$, $p_{o,2}$; показник ефективності ресурсоспоживання r_e; показник достовірності виявлення за часом p_d.

Таким чином, загальна ефективність роботи розробленої РБС [318] у локальній комп'ютерній мережі з використанням емуляторів процесора із зміненими налаштуваннями складає: $E \approx 0{,}91 \cdot 0{,}97 \cdot 0{,}94 \approx 0{,}83$.

Отже, результати дослідження достовірності розробленої РБС у локальній комп'ютерній мережі показують, що застосування розробленого програмного забезпечення дозволяє підвищити рівень достовірності виявлення на 5-12 % порівняно з існуючими антивірусними програмними засобами, досягти зниження рівня помилок першого роду до 5 % та покращити ефективність виявлення.

Висновки до шостого розділу

Програмне забезпечення РБС [303, 304, 363] дозволяє здійснювати її доповнення новими методами, реалізує зв'язуючу частину розподіленої системи і може бути використано для проведення експериментальних досліджень.

Використання розробленої РБС Distributed Multilevel System та її підсистем для виявлення певних типів ЗПЗ у локальних комп'ютерних мережах є достатніми [381, 385, 389] порівняно з аналогічними мережними АПЗ і підтверджують ефективність запропонованих теоретичних моделей та

розроблених методів.

Результати дослідження достовірності розробленої РБС у локальній мережі показують, що застосування розробленого програмного забезпечення дозволяє підвищити рівень достовірності виявлення на 5-12 % порівняно з існуючими антивірусними програмними засобами, досягти зниження рівня помилок першого роду до 5 % та покращити ефективність її функціонування при виявленні.

Основні результати розділу опубліковані у працях [303, 304, 363, 381, 385, 389].

ВИСНОВКИ

У дисертаційній роботі вирішено актуальну науково-технічну проблему – здійснено розвиток теорії і практики створення розподілених систем виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах для покращення ефективності його виявлення. Вирішення даної проблеми має важливе значення в усіх галузях, де активно застосовуються локальні комп’ютерні мережі. При цьому отримано такі основні наукові й практичні результати:

1. Проведено аналіз сучасних мережних систем виявлення зловмисного програмного забезпечення, який показав невисоку ефективність таких систем, що зумовлено низькою достовірністю виявлення нового та існуючого зловмисного програмного забезпечення, через використання ними комбінації технологій приховування своєї присутності та поширення. В якості напряму дослідження проблеми було вибрано розподілені системи виявлення в локальних комп’ютерних мережах.

2. Розроблена удосконалена модель архітектури розподіленої системи виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах базується на комплексному врахуванні вимог розподіленості, децентралізованості, багаторівневості та самоорганізованості, що дозволяє створювати на її основі розподілені системи та їх компоненти, які функціонуватимуть автономно і самостійно прийматимуть рішення про наявність зловмисного програмного забезпечення та нарощення своїх функціональних можливостей.

3. Розроблена модель архітектури типових компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення на основі структур Кріпке з представленням компонентів через стани, в яких вони можуть перебувати під час функціонування. Вона дозволила враховувати перебування програмних модулів у різних станах і стала основою для визначення стану безпеки всієї розподіленої системи та її компонентів. Модель її архітектури

дозволяє здійснювати збільшення кількості рівнів системи без її зміни. Основою архітектури РБС виступають програмні модулі з однаковими архітектурами, але при цьому кожен з них може самостійно приймати рішення на основі різних даних, зібраних з різних КС локальної мережі.

4. Розроблений метод взаємодії компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення на основі підтримки її цілісності та визначення порядку передачі знань між її компонентами і використання встановлених аналітичних залежностей між рівнями безпеки програмних модулів та рівнем безпеки всієї розподіленої багаторівневої системи, дозволяє системі автономно змінювати свою архітектуру та функції без втручання користувача, а також визначати стратегію своєї подальшої роботи. Метод є основою для розробки зв'язуючої частини програмного забезпечення, яка організовує взаємодію компонентів розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах.

5. Розроблені алгебраїчні системи та алгебри, з введеними операціями на множині ЗПЗ, стали основою для створення поведінкових сигнатур ЗПЗ з метою їх формалізованого представлення в системах виявлення. Особливістю розроблених алгебраїчних систем є структуризація ЗПЗ за типами, яка дозволяє здійснювати їх розподіл і віднесення до підмножин на основі характеристичних властивостей ЗПЗ для проведення ідентифікації та класифікації. Формалізовані властивості ЗПЗ відображені в розроблених алгебрах і заданих моделях. Удосконалені моделі типів зловмисного програмного забезпечення представлені їх алгебрами поведінки стали основою створення базису поведінкових сигнатур, враховують особливості їх функціонування в локальних комп'ютерних мережах і дозволяють здійснити класифікацію за типами поведінки.

6. Розроблена еталонна модель бот-мереж, в основі якої її типові компоненти, задані відповідними функціями, відображають типові дії мережного зловмисного програмного забезпечення в локальних комп'ютерних мережах. Вона базується на основі компонентів трьох рівнів, що дало можливість

представити відомі бот-мережі, які були виявлені і певним чином класифіковані за характерними ознаками. Розроблені для компонентів еталонної моделі та моделей типових бот-мереж числові характеристики стали основою для застосування методу виявлення бот-мереж у локальних комп'ютерних мережах.

7. Розроблено метод виявлення бот-мереж у локальних комп'ютерних мережах, суть якого полягає в здійсненні активного моніторингу системних подій та узгодженій взаємодії компонентів розподіленої системи при прийнятті рішення, надав змогу створювати засоби, які здатні інтегруватись у розподілену систему та класифікувати бот-мережі за їх поведінковими сигнатурами, що формуються закладеними в їх компоненти функціями.

8. Розроблено метод виявлення файлового зловмисного програмного забезпечення в локальних комп'ютерних мережах, який полягає в поєднанні роботи програмних агентів, що здійснюють виявлення зловмисного програмного забезпечення в окремих комп'ютерних системах, відповідно до імплементованих в них методів: динамічного формування поведінкової сигнатури шляхом відстеження викликів прикладного програмного інтерфейсу; знаходження поліморфного та метаморфного програмного коду; сканування виконуваних програм шляхом створення для них автономних процесів та відповідних програмних агентів у розподіленій системі. Це дозволило покращити аналіз і підвищити достовірність виявлення зловмисного програмного забезпечення.

9. Розроблено метод виявлення файлового зловмисного програмного забезпечення, який базується на основі динамічного формування поведінкової сигнатури шляхом відстеження API-викликів. Він може бути використаний для виявлення інших видів вірусних програм, зокрема, нових версій існуючих вірусів. Метод включає формування сигнатури вірусної програми на основі трасування API-викликів, що дозволяє здійснити виявлення вірусної програми, яка представлена розробленою поведінковою сигнатурою з бази сигнатур. Поведінкова сигнатура включає критичні API-виклики за групами зловмисної активності та відображає частоту їх входження, а також характер взаємодії критичних API-функцій вірусної програми та описує взаємозв'язок між

критичними API-функціями. Це надає можливість розмежувати вірусні програми від корисних застосунків не тільки за наявністю критичних API-викликів, але й за їх взаємодією між собою. Для здійснення виявлення використовується класифікація.

10. Для файлового ЗПЗ, яке використовує техніки заплутування свого коду, розроблено метод виявлення поліморфних та метаморфних вірусів на основі аналізу функцій обфускації. Особливістю методу є аналіз програмного об’єкта та його модифікованих версій, отриманих від різних ПМ РБС, і подальший аналіз на основі пошуку еквівалентних функціональних блоків. Це дозволяє здійснити більш детальний аналіз коду програмного об’єкта на наявність поліморфних та метаморфних вірусів.

11. Розроблено програмне забезпечення РБС Distributed Multilevel System, яке реалізує запропоновані теоретичні основи розподілених систем та підтверджує можливість їх практичного створення. Воно дозволяє здійснити перевірку достовірності виявлення ЗПЗ на основі запропонованих рішень, проведенням експериментів, порівняно з існуючими мережними АПЗ.

12. Впровадження розподіленої багаторівневої системи Distributed Multilevel System виявлення зловмисного програмного забезпечення в локальних комп’ютерних мережах здійснено в Державному підприємстві «Новатор» (відділ автоматизованих систем управління), товаристві з обмеженою відповідальністю «ITT telecommunication company», товаристві з обмеженою відповідальністю «ЮКС++», софтовій компанії CYPRESS і дозволило підвищити достовірність виявлення нового та існуючого ЗПЗ порівняно з відомим мережними антивірусними засобами на 5–12 % та досягти зниження рівня помилок першого роду до 5 %.

Результати дисертаційної роботи можуть бути запропоновані для використання науковими організаціями і підприємствами, які займаються розробкою та впровадженням мережних антивірусних засобів з метою покращення ефективності їх функціонування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Adleman L. An Abstract Theory of Computer Viruses / L. Adleman // Proceedings of the Conference on the Theory and Application of Cryptography. – Santa Barbara, CA (USA), 21–25 August, 1988. – Pp. 354–374.

2. Adleman L. A method for obtaining digital signatures and public-key cryptosystems / L. Adleman, R. Rivest, A. Shamir // Proceedings of the Communications of the ACM - Special 25-th Anniversary Issue. – Vol. 26. – 1983. – Pp. 1–3.

3. Akiyama M. A proposal of metrics for botnet detection based on its cooperative behavior / M. Akiyama, T. Kawamoto, M. Shimamura, T. Yokoyama, Y. Kadobayashi, S. Yamaguchi // International Symposium on Applications and the Internet Workshops (SAINTW'07), Hiroshima (Japan) January 15–19, 2007. – Pp. 82–85.

4. Alazab M. Malware Detection Based on Structural and Behavioral Features of API Calls / M. Alazab, R. Layton, S. Venkataraman, P. Watters Proceedings of the 1-st International Cyber Resilience Conference. – Perth Weste (Australia), August 23, 2010. – Pp. 1–8.

5. Alazab M. Spam and criminal activity / M. Alazab, R. Broadhurst // Trends and Issues in Crime and Criminal Justice (Australian Institute of Criminology). – December 3, 2016 . – Vol. 526 – Pp. 1–20.

6. Al-Duwairi B. GFlux: A google-based system for Fast Flux detection/ B. Al-Duwairi, A. Al-Hammouri, M. Aldwairi, V. Paxson// 2015 IEEE Conference on Communications and Network Security (Florence, Italy). – September 28, 2015. – Pp. 755-756.

7. Alieyan K. A survey of botnet detection based on DNS // K. Alieyan, A. ALmomani, A. Manasrah, M. M Kadhum // Neural Computing and Applications. – 2017. – Vol. 28. – No. 7. – Pp. 1541–1558.

8. Alieyan K. An overview of DDoS attacks based on DNS/ K. Alieyan, M. M Kadhum, M. Anbar, S. Ul Rehman, N. KA Alajmi // 2016 International Conference on Information and Communication Technology Convergence (ICTC). – Jeju Island (South Korea), October 19, 2016. – Pp. 276–280.

9. Alieyan K. Botnets Detecting Attack Based on DNS Features/ K. Alieyan, M. Anbar, A. Almomani, R. Abdullah, M. Alauthman // 2018 International Arab Conference on Information Technology (ACIT) - Islamic University of Lebanon, (Lebanon), November 28, 2018. – Pp. 1–4.

10. Alkhateeb F. Multi-Agent Systems - Modeling, Interactions, Simulations and Case Studies / F. Alkhateeb, E.A. Maghayreh, I. Doush. – 2011. – 522 p.

11. Alqurashi S. A Comparison of Malware Detection Techniques Based on Hidden Markov Model / S. Alqurashi, O. Batarfi // Journal of Information Security. – 2016. – Vol. 7. – Pp. – 215–223.

12. Alqurashi S. A comparison between API call sequences and opcode sequences as reflectors of malware behavior/ S. Alqurashi, O. Batarfi// 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST) (London, UK). –11.12.2017 – Pp. 105–110.

13. Alqurashi S. Automated behavioral malware analysis system/ S. Alqurashi, O. Batarfi // Information Technology: New Generations. – 2016. – Pp. 1243–1248.

14. Anderson B. Graph-based malware detection using dynamic analysis / B. Anderson, D. Quist, J. Neil, C. Storlie, T. Lane // Journal in Computer Virology. – 2011. – Vol. 7. – Issue 3. – Pp. 247–258.

15. Anderson B. Deciphering malware's use of TLS (without decryption)/ B. Anderson, S. Paul, D. McGrew// Journal of Computer Virology and Hacking Techniques – 1.08.2018. – Vol. 14. – No. 3. – Pp. – 195-211.

16. Anderson B. Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity / B. Anderson, D. McGrew // Proceedings of the 23rd ACM SIGKDD International Conference on

Knowledge Discovery and Data Mining (London, UK.). – August 13, 2017. – Pp. 1723–1732.

17. Anderson B. OS fingerprinting: New techniques and a study of information gain and obfuscation / B. Anderson, D. McGrew // 2017 IEEE Conference on Communications and Network Security (CNS). – 9.10.2017. – Pp. 1–9.

18. Antonakakis M. Building a Dynamic Reputation System for DNS / M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, N. Feamster // Proceedings of the 19th USENIX conference on Security. – Washington, DC (USA), August 11-13, 2010. – Pp. 273-290.

19. Antonakakis M. From throw-away traffic to bots: Detecting the rise of dga-based malware / M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, D. Dagon // Proceedings of the 21st USENIX conference on Security symposium. – Bellevue, WA (USA), August 08-10, 2012. – Pp. 491-506.

20. Antonakakis M. Understanding the mirai botnet/ / M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, Yi. Zhou // 26th {USENIX} Security Symposium ({USENIX} Security 17)(Vancouver, BC). – 2017. – Pp. 1093-1110.

21. Attaluri S. Profile Hidden Markov Models and Metamorphic Virus Detection / S. Attaluri, S. McGhee, M. Stamp // Journal in Computer Virology. – 2009. – Vol. 5. – Issue 2. – Pp. 151-169.

22. Austin T. H. Exploring Hidden Markov Models for Virus Analysis: A Semantic Approach / T. H. Austin, E. Filiol, S. Josse, M. Stamp // Proceedings of the 46-th Hawaii International Conference on System Sciences. – Wailea (USA), January 7-10, 2013. – Pp. 5039-5048.

23. Austin T. H. Multiple facets for dynamic information flow with exceptions / T. H. Austin, T. Schmitz, C. Flanagan// ACM Transactions on Programming Languages and Systems (TOPLAS) - July 20, 2017. – Vol. 39, No. 3. – Pp. 10-14.

24. Avast! [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://www.avast.com/index (Viewed on April 2, 2019). – Title from the screen.

25. AVG [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.avg.com (Viewed on April 2, 2019). – Title from the screen.

26. Avira [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.avira.com (Viewed on April 2, 2019). – Title from the screen.

27. Bacher P. The Honeynet Project. Know your enemy: Tracking botnets [Electronic resources] / P. Bacher, T. Holz, M. Kotter [et al] // Mode of access: http://www.honeynet.org/papers/bots (Viewed on March 25, 2019). – Title from the screen.

28. Bakotech [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://bakotech.ua/ (Viewed on April 2, 2019). – Title from the screen.

29. Basheer N. Fast Flux Watch: A mechanism for online detection of fast flux networks / N. Basheer, B. AI-Duwairi, Ahmad T. AI-Hammouri // Journal of Advanced Research. – 2014. – Pp. 473-479.

30. Bedratyuk L. The Star Sequence and the General First Zagreb Index // L. Bedratyuk, O. Savenko / MATCH Communications in Mathematical and in Computer Chemistry. – 2018. – Vol. 79, № 2. – Pp.407-414.

31. Bhatia J.S. Botnet Command Detection using Virtual Honeynet / J.S. Bhatia, R.K. Sehgal, S. Kumar // International Journal of Network Security & Its Applications. – 2011. – Vol. 3. – No. 5. – Pp. 177-189.

32. Bilge L. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis / L. Bilge, E. Kirda, C. Kruegel, M. Balduzzi // Proceedings of the Network and Distributed System Security Symposium, NDSS. – San Diego, California (USA), February 6-9, 2011. – Pp. 1-17.

33. Bilge L. Riskteller: Predicting the risk of cyber incidents / L. Bilge, Y. Han, M. Dell'Amico // Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (Dallas, Texas, USA). — October 30, 2017. – Pp. 1299-1311.

34. Binsalleeh H. On the Analysis of the Zeus Botnet Crimeware Toolkit / H. Binsalleeh, T. Ormerod, A. Boukhtouta [et al] // Proceedings of the 8th Annual Conference on Privacy, Security and Trust, (PST, 2010). – Ottawa, ON (Canada), August 17-19, 2010. – Pp. 31-38.

35. Bitdefender [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.bitdefender.com/(Viewed on April 2, 2019). – Title from the screen.

36. Bonfante G. Abstract detection of computer viruses / G. Bonfante, M. Kaczmarek, J.-Y. Marion // Proceedings of the third Workshop on Applied Semantics (APPSEM'05). – Frauenchiemsee (Germany), September, 2005. – Pp. 8-14.

37. Bonfante G. A Classification of viruses through recursion theorems / G. Bonfante, M. Kaczmarek, J.-Y. Marion // Proceedings of the Computation and Logic in the Real World. – Siena (Italy), 18-23 June, 2007. – Pp. 73–82.

38. Bonfante G. Two function algebras defining functions in NCk boolean circuits / G. Bonfante, R. Kahle J.-Y. Marion, I. Oitavem // Information and Computation. – June, 2016. – Vol. 248. – Pp. 82-103.

39. Borello J.-M. From the design of a generic metamorphic engine to a black-box classification of antivirus detection techniques / J.-M. Borello, E. Filiol, L. Me // Journal in Computer Virology. – 2009. – Vol. 6. – Pp. 277-287.

40. Borup L. Peer-to-peer botnets: A case study on Waledac / L. Borup. – Master's thesis, Technical University of Denmark. – 2009. – 62 p.

41. Branitskiy A. Hybridization of computational intelligence methods for attack detection in computer networks / A. Branitskiy, I. Kotenko // Journal of Computational Science. – 2017. – No.23. – Pp.145–156.

42. Bruschi D. Code normalization for self-mutating malware / D. Bruschi, L. Martignoni, M. Monga // IEEE Security & Privacy. – 2007. – Vol. 5. – No. 2. – Pp. 46-54.

43. Bruschi M. Detecting self-mutating malware using control-flow graph matching / M. Bruschi, L. Martignoni, M. Monga // Proceedings of the conference on Detection of Intrusions and Malware & Vulnerability Assessment. – Berlin (Germany), July 13-14, 2006. – Pp. 129-143.

44. Bhowmick A. Security Mechanisms for Precious Data Protection of Divergent Heterogeneous Grid Computing Resources / A. Bhowmick, G.V.N. Prasad // International Journal of Research and Scientific Innovation. – 2017. – Vol. 4. – Issue 4. – Pp. 39–42.

45. Caglayan A. Real-time detection of fast flux service networks / A. Caglayan, M. Toothaker, D. Drapeau, D. Burke, G. Eaton // Proceedings of the Cybersecurity Applications & Technology Conference for Homeland Security. – Washington, DC, (USA), March 3–4, 2009. – Pp. 285-292.

46. Cai N. Decentralized Modeling, Analysis, Control, and Application of Distributed Dynamic Systems / N. Cai, R. Sabatini, X.-W. Dong, M. J Khan and Y. Yu // Journal of Control Science and Engineering. – 2016. – Vol. 2016. – Pp. 1-2.

47. Cannell J. Nowhere to Hide: Three methods of XOR obfuscation [Electronic resource] / J. Cannel – Electronic data. – Mode of access: https://blog.malwarebytes.com/threat-analysis/2013/05/nowhere-to-hide-three-methods-of-xor-obfuscation/ (viewed on October 2, 2016). – Title from the screen.

48. Carlin D. Dynamic Analysis of Malware Using Run-Time Opcodes/ D. Carlin, P. O'Kane, S.Sezer // Data Analytics and Decision Support for Cybersecurity. – 2017. – Pp. 99-125.

49. Celik Z.B. Detection of Fast-Flux Networks using various DNS feature sets / Z.B. Celik, S. Oktug // Computers and Communications (ISCC). – Split, (Croatia), July 7-10, 2013. – Pp. 868-873.

50. Celik Z.B. Extending Detection with Privileged Information via Generalized Distillation / Z.B. Celik, P. McDaniel// 2018 IEEE Security and Privacy Workshops (SPW). – May 24, 2018. – Pp. 83-88.

51. Cesare S. Malware Variant Detection Using Similarity Search over Sets of Control Flow Graphs / S. Cesare, Y. Xiang // Proceedings of the 10-th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. – Washington, DC (USA), November 16-18, 2011. – Pp. – 181-189.

52. Challoo R. Detection of Botnets Using Honeypots and P2P Botnets / R. Challoo, R. Kotapalli // International Journal of Computer Science and Security. – Vol. 5. – Issue 5. – 2011. – Pp. 496-502.

53. Chandraiah J. Fake anti-virus: The journey from Trojan to a persistent threat [Electronic resource] / J. Chandraiah – Electronic data. – Mode of access: https://nakedsecurity.sophos.com/fake-anti-virus-the-journey-from-trojan-to-a-persistent-threat-4/(viewed on October 10, 2016). – Title from the screen.

54. Chaumette S. Automated extraction of polymorphic virus signatures using abstract interpretation / S. Chaumette, O. Ly, R. Tabary // Proceedings of the 2011 5th International Conference on Network and System Security. – Milan (Italy), September 6-8, 2011. –Pp. 41-48.

55. Chen X. Towards an Understanding of Anti-virtualization and Anti-debugging Behavior in Modern Malware / X. Chen, J. Andersen, Z. M. Mao, M. Bailey, J. Nazario // Proceedings of the 5th IEEE International Conference on Malicious and Unwanted Software. – Anchorage (USA), June 24-27, 2008. – Pp. 177-186.

56. Cho C.Y. Insights from the Inside: A View of Botnet Management from Infiltration / C.Y. Cho, J. Caballero, C. Grier, C. Paxson [et al] // Proceedings of the 3rd Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET‘10). – San Jose, CA (USA), April 27, 2010. – Pp. 1-8.

57. Choi H. Botnet Detection by Monitoring Group Activities in DNS Traffic / H. Choi, H. Lee, H. Lee, H. Kim // Proceedings of the 7-th IEEE International Conference on Computer and Information Technology (CIT 2007). – Aizu-Wakamatsu, Fukushima (Japan), October 16-19, 2007. – Pp. 715-720.

58. Choi H. Identifying botnets by capturing group activities in DNS traffic / H. Choi, H. Lee // Computer Networks. – 2012. – Vol.56. – Pp. 20-33.

59. Christodorescu M. Mining Specification of Malicious Behavior / M. Christodorescu, S. Jha, C. Krugel // Proceeding of the 6th joint meeting of the European Software Engineering Conference. – Dubrovnik (Croatia), September 03 – 07, 2007. – Pp. 5-14.

60. Christodorescu M. Semantics-aware malware detection / M. Christodorescu, S. Jha, S. A. Seshia, D. Song, R. E. Bryant // Proceedings of the 15-th IEEE Symposium on Security and Privacy. – Oakland (USA), May 08–11, 2005. – Pp. 32-46.

61. Christodorescu M. Static analysis of executables to detect malicious patterns / M. Christodorescu, S. Jha // Proceedings of the 12-th USENIX Security Symposium. – Washington DC (USA), August 04-08, 2003. – Pp. 169-186.

62. Christodorescu M. System and method for protection from buffer overflow vulnerability due to placement new constructs in C++ / M. Christodorescu, A. Kundu, A. Mohindra // Patent Num 9600663 (US). – Match 21, 2017.

63. ClamAV [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://www.clamav.net/ (Viewed on April 2, 2019). – Title from the screen.

64. Cohen F. Computational aspects of computer viruses / F. Cohen // Computers and Security. – 1989. – Vol. 8. – Pp. 325–344.

65. Cohen F. Computer viruses: theory and experiments / F. Cohen // Computers and Security. – 1987. – Vol. 6. – Pp. 22–35.

66. COMSS1 [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.comss.ru/page.php?id=2758 (Viewed on April 2, 2019). – Title from the screen.

67. Cooke E. The zombie roundup: understanding, detecting, and disrupting botnets / E. Cooke, F. Jahanian, D. McPherson // Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop, (SRUTI'05). – Cambridge, MA (USA), 2005. – Pp. 1-6.

68. Corporate Endpoint Protection Products Group Test: Socially-Engineered Malware Q2 [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.nsslabs.com/ research/ endpoint-security/ anti-malware/q2-2010-endpoint-protection-product-group-test.html (Viewed on April 2, 2019). – Title from the screen.

69. Dagon D. A Taxonomy of Botnet Structures / D. Dagon, G. Gu, C.P. Lee // Proceedings of the Twenty-Third Annual Computer Security Applications

Conference (ACSAC 2007). – Miami Beach, FL (USA), December 10-14, 2007. – Pp. 143-164.

70. Dagon D. Modeling botnet propagation using time zones / D. Dagon, C. Zou, W. Lee // Proceedings of the 13th Annual Network and Distributed System Security Symposiumю – San Diego, California (USA), 28 February – 2 March, 2006. – Pp. 2-13.

71. DAMBALLA. Botnet communication topologies. Understanding the intricacies of botnet command-and-control [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://www.damballa.com/downloads/r_pubs /WP_Botnet_Communications_Primer.pdf (Viewed on April 2, 2019). – Title from the screen.

72. DAMBALLA. Botnet detection for communications service providers [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https:// www.damballa.com/ downloads/r_pubs/WP_Botnet_Detection_for_CSPs. pdf (Viewed on April 2, 2019). – Title from the screen.

73. DAMBALLA. On the Kraken and Bobax Botnets [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://www.damballa.com/downloads /r_pubs/Kraken_Response.pdf (Viewed on April 2, 2019). – Title from the screen.

74. Daoud E. A. Computer Virus Strategies and Detection Methods / E.A. Daoud, I.H. Jebril, B. Zaqaibeh // International Journal of Open Problems in Computer Science and Mathematics. – 2008. – Vol. 1. – No. 2. – Pp. 29-36.

75. Davis C.R. Structured peer-to-peer overlay networks: Ideal botnets command and control infrastructures? / C.R. Davis, S. Neville, J.M. Fernandez, J.M. Robert // Euro. Symp. Research in Computer Security. – Málaga (Spain), October 6-8, 2008. – Pp. 461-480.

76. Demarest J. Statement before the Senate Judiciary Committee, Subcommittee on Crime and Terrorism, Washington, D.C., 2014, July 15 [Electronic resource] / The Federal Bureau of Investigation. Official Website. – Mode of access: http://www.fbi.gov/news/testimony/taking-down-botnets. – Title from the screen.

77. Desai P. A highly metamorphic virus generator / P. Desai, M. Stamp // International Journal of Multimedia Intelligence and Security. – 2010. – Vol. 1. – No. 4. – Pp. 402-427.

78. Dietrich C.J. On Botnets that use DNS for Command and Control / C.J. Dietrich, C. Rossow, F.C. Freiling [et al] // Proceedings of the 2011 Seventh European Conference on Computer Network Defense. – Gothenburg (Sweden), September 6-7, 2011. – Pp. 9-16.

79. Dinaburg A. Ether: Malware Analysis via Hardware Virtualization Extensions / A. Dinaburg, P. Royal, M. Sharif, W. Lee // Proceedings of the 15-th ACM Conference on Computer and Communications Security. – Alexandria (USA), October 27-31, 2008. – Pp. 51-62.

80. Dittrich D. The "Stacheldraht" distributed denial of service attack tool / D. Dittrich // Technical report. University of Washington. – 2013.

81. Dittrich D. Toward Smarter Vulnerability Discovery Using Machine Learning / D. Dittrich, G. Grieco // AISec '18 Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security (Toronto, Canada). — October 15 - 19, 2018. – Pp. 48-56.

82. Dittrich D. Taking Down Botnets-Background / D. Dittrich // Taking Down Botnets-Background. – 2016.

83. Dr. WEB Anti-virus [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.drweb.com/ (Viewed on April 2, 2019). – Title from the screen.

84. English Oxford Living Dictionaries – Definition of virus in English [Electronic resource]: [Web-site]. – Electronic data. – Mode of Access: https://en.oxforddictionaries.com/ definition/virus (Viewed on May 9, 2017). – Title from the screen.

85. Enterprise End Point Protection Comparative Analysis - Socially Engineered Malware: Report Overview [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: www.nsslabs.com/reports (Viewed on April 2, 2019). – Title from the screen.

86. Erdogan O. Hash-AV: Fast Virus Signature Scanning by Cache-Resident Filters / O. Erdogan, P. Cao // Proceedings of The 5-th IEEE Global Telecommunications Conference. – St. Louis (USA), 28 November – 2 December, 2005. – Pp. 101-110.

87. ESET Endpoint Security [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.eset.com/ (Viewed on April 2, 2019). – Title from the screen.

88. Eskandari M. A graph mining approach for detection unknown malwares / M. Eskandari., S. Hashemi // Journal of Visual Languages & Computing. – 2012. – Vol. 23. – Issue 3. – Pp. 154-162.

89. Eslahi M. Correlation-based HTTP Botnet detection using network communication histogram analysis / M. Eslahi, W.Z. Abidin, and M.V. Naseri // 2017 IEEE Conference on Application, Information and Network Security (AINS). – Miri (Malaysia), November 13-14, 2017. – Pp. 7-12.

90. Eslahi M. Cooperative network behavior analysis model for mobile HTTP botnet detection / M. Eslahi // Institute of Graduate Studies, UiTM. – 2017. - Vol.12. – No.12. – P. 23.

91. Eslahi M. Mobile botnet detection model based on retrospective pattern recognition // M. Eslahi, M. Yousefi, M.V. Naseri, Y.M. Yussof, N.M. Tahir, H. Hashim // International Journal of Security and Its Applications. – January 1, 2016. – Vol. 10. – No. 9. – Pp. 39-44.

92. Evading anti-virus's script emulator. Kaspersky [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://malwaretips.com/threads/evading-anti-viruss-script-emulator-kaspersky.33086 (viewed on October 10, 2016). – Title from the screen.

93. Farnham G. Detecting DNS tunneling / G. Farnham, A. Atlasis // SANS Institute InfoSec Reading Room. – 2013. – Pp. 1-32.

94. Faust J. Mitigating Browser Based Exploits through Behavior Based Defenses and Hardware Virtualization / J. Faust. – SANS Institute: Information Security Reading. – 2011. – 43 p.

95. Faust J. Distributed Analysis of SSH Brute Force and Dictionary Based Attacks / J. Faust // Culminating Projects in Information Assurance. – May 25, 2018. – P. 56.

96. Feily M. A Survey of Botnet and Botnet Detection» / M. Feily, A. Shahrestani // Proceedings of the 2009 Third International Conference on Emerging Security Information, Systems and Technologies. – Washington, DC (USA), June 18-23, 2009. – Pp. 268-273.

97. Ferguson R. The Botnet Chronicles: a journey to infamy / R. Ferguson // Trend Micro Whitepaper. – 2010. –13 p.

98. Filiol E. Malware pattern scanning schemes secure against black-box analysis / E. Filiol // Journal in Computer Virology. – 2006. – Vol. 2. – Pp. 35-50.

99. Filiol E. Proactive Detection of Unknown Binary Executable Malware / E. Filiol // Advances in Security in Computing and Communications. – July, 2017. – Pp. 3-31.

100. Florian C. The Most Vulnerable Operating Systems and Applications in 2011 [Electronic resources] / C. Florian // Mode of access: https://techtalk.gfi.com/report-most-vulnerable-operating-systems-and-applications - in-2013/ (Viewed on March 25, 2019). – Title from the screen.

101. Gandotra E. Malware Analysis and Classification: A Survey / E. Gandotra, D. Bansal, S. Sofat // Journal of Information Security. – 2014. – Vol. 5. – Pp. 56-64.

102. Gandotra E. Clustering Morphed Malware using Opcode Sequence Pattern Matching / E. Gandotra, S. Singla, D. Bansal, S. Sofat // Recent Patents on Engineering. – 1.04.2018. – Vol. 12. – No. 1. – Pp. 30-36.

103. Gandotra E. Malware threat assessment using fuzzy logic paradigm / E. Gandotra, D. Bansal, S. Sofat // Cybernetics and Systems. – January 2, 2017. - Vol. 48. – No. 1. - Pp. 29-48.

104. Gandotra E. Zero-day malware detection / E. Gandotra, D. Bansal, S. Sofat // 2016 Sixth International Symposium on Embedded Computing and System Design (ISED), Patna, (India), December 15, 2016. – Pp. 171-175.

105. Gao H. An empirical reexamination of global dns behavior / H. Gao, V. Yegneswaran, Y. Chen, P. Porras, S. Ghosh, J. Jiang, H. Duan // Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM. – Hong Kong (China), August 12-16, 2013. – Pp. 267-278.

106. Gao H. Reexamining DNS from a global recursive resolver perspective / H. Gao, V. Yegneswaran, J. Jiang, Y. Chen, P. Porras, S. Ghosh, H. Duan // IEEE/ACM Transactions on Networking (TON). – February 1, 2016. –Vol. 24. – No. 1. - Pp. 43-57.

107. Ghiasi M. Dynamic malware detection using registers values set analysis / M. Ghiasi, A. Sami, Z. Salehi // Proceedings of the 9-th Information Security and Cryptology International ISC Conference. – Tabriz (Iran), September 13-14, 2012. – Pp. 54-59.

108. Goebel J. Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation / J. Goebel, T. Holz // Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets. – Berkeley, CA (USA), 2007. – Pp. 1-8.

109. Golovko V. Neural Network Artificial Immune System for Malicious Code Detection / V.Golovko, S.Bezobrazov // Brest State Technical University. –2015. – Pp. 1-7.

110. Grizzard J.B. Peer-to-Peer Botnets: Overview and Case Study / J. B. Grizzard, V. Sharma, C. Nunnery [et al] // Proceedings of the First Workshop on Hot Topics in Understanding Botnets, (HotBots, 2007). – Cambridge (USA), April 10. – 2007.

111. Gu G. BotHunter: detecting malware infection through IDS-driven dialog correlation / G. Gu, P. Porras, V. Yegneswaran, M. Fong, W. Lee // Proceedings of the 16-th USENIX Security Symposium. – Boston (USA), 2007. – Pp. 167-182.

112. Gu G. BotSniffer: Detecting botnet command and control channels in network traffic / G. Gu, J. Zhang, and W. Lee // Proceedings of the 15thAnnual Network and Distributed System Security Symposium. – San Diego (USA), 10-13 February, 2008.

113. Guerid H. Privacy-preserving domain-flux botnet detection in a large scale network / H.Guerid, K. Mittig, A. Serhrouchni // Proceedings of the 2013 Fifth International Conference on Communication Systems and Networks (COMSNETS). – Bangalore (India), January 7-10, 2013. – Pp. 1-9.

114. Guharoy R. A theoretical and detail approach on grid computing a review on grid computing applications / R. Guharoy et al // Proceedings of 8th Annual Industrial Automation and Electromechanical Engineering Conference. – Bangkok (Thailand), August 16-18, 2017. – Pp. 142–146.

115. Holz T. Know your enemy: Tracking botnets [Electronic resources] / P. Bacher, T. Holz, M. Kotter // Mode of access: http://www.honeynet.org/papers/bots (Viewed on March 25, 2019). – Title from the screen.

116. Holz T. Measurements and mitigation of peer-to-peer-based botnets: A case study on Storm worm. / T. Holz, M. Steiner, F. Dahl [et al] // Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats. – San Francisco, California (USA), April 15 - 15, 2008.

117. Hsieh Y.-Y. Bitcoin and the rise of decentralized autonomous organizations / Y.-Y. Hsieh, J.-P. Vergne, P. Anderson [et al] // Journal of Organization Design. – 2018. – 1-16.

118. Ichise H. Detection Method of DNS-based Botnet Communication Using Obtained NS Record History / H. Ichise, J. Yong, K. Iida // Proceedings of the Computer Software and Applications Conference (COMPSAC). – Taichung (Taiwan), July 1-5, 2015. – Pp. 676-677.

119. Information technology. Security techniques. Code of practice for information security management: ISO/IEC 17799:2005. – [Introduced 15.06.2005]. – Geneva (Switzerland): ISO, 2005. – 12 p. – (International standard).

120. Information technology. Security techniques. Information security management systems. Overview and vocabulary: ISO/IEC 27000:2018. – [Introduced 02.2018]. – Geneva (Switzerland): ISO, 2018. – 27 p. – (International standard).

121. Information technology. Security techniques. Information security management guidelines for financial services: ISO/IEC 27015:2015. – [Introduced 01.1.2015]. – Geneva (Switzerland): ISO, 2015. – 9 p. – (International standard).

122. Jacob G. Malwares as interactive machines: A new framework for behavior modelling / G. Jacob, E. Filiol, and H. Debar // Journal in Computer Virology. – 2008. – Vol. 4. – No. 3, 2008. – Pp. 235-250.

123. Jang D.S. Decentralized Message Passing Algorithm for Distributed Minimum Sensor Cover / D.S. Jang and H.-L. Choi // Journal of Aerospace Information Systems. – 2017. – Vol. 14. – No. 7. – Pp 373-390.

124. Kanich C. Spamalytics: An Empirical Analysis of Spam Marketing Conversion / C. Kanich, C. Kreibich, Levchenko K. [et al] // Proceedings of the 15-th ACM Conference on Computer and Communications Security, (CCS, 2008). – Alexandria, VA, (USA), October 27 - 31, 2008. – Pp. 99-107.

125. Kanich C. The costs and benefits of treating the browser like an operating system: how ad blockers help us better understand how different browser features are used and who they benefit / C. Kanich // ACM SIGCAS Computers and Society. – July 23, 2017. – Vol. 47. – No. 2. – Pp. 16-18.

126. Karasaridis A. Detection of DNS anomalies using flow data analysis / A. Karasaridis, K.S. Meier-Hellstern, D.A. Hoeflin // IEEE Globecom 2006. – San Francisco, CA (USA), 27 November-1 December, 2006. – Pp. 1-6.

127. Karasaridis A. Artificial Intelligence for Cybersecurity / A. Karasaridis, B. Rexroad, P. Velardo // Artificial Intelligence for Autonomous Networks. – September 25, 2018. – Pp. 243-274.

128. Kaspersky Lab [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.kaspersky.ru (Viewed on April 2, 2019). – Title from the screen.

129. Kaspersky Security Bulletin 2019. Overall Statistics for 2019[Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://securelist.com/analysis/kaspersky-security-bulletin/58265/kaspersky-security-bulletin-2019-overall-statistics-for-2019/ (Viewed on April 2, 2019). – Title from the screen.

130. Kelley R. A Hybrid Real-time Zero-day Attack Detectionand Analysis System / R. Kelley, M.Singh // International Journal Computer Network and Information Security. – 2015. – Vol. 9. – Pp. 19-31.

131. Kelley R. Unmasking Android Obfuscation Tools Using Spatial Analysis / R. Kelley, Y. Ning, H. Gonzalez, N. Stakhanova // 2018 16th Annual Conference on Privacy, Security and Trust (PST). – Pp. 1-10.

132. Khalkhali I. Host-based Web Anomaly Intrusion Detection System, an Artificial Immune System Approach / I. Khalkhali, R. Azmi, M. Azimpour-kivi, M. Khansari // International Journal of Computer Science Issues. – 2011. – Vol. 8. – Issue 5. – No 2. – Pp. 14-24.

133. Khedikar S. Security in Grid Computing Using Globus and Legion / S. Khedikar, K. Jadhav, G. Kumar, A. Rajanale // International Research Journal of Engineering and Technology. – 2017. – Vol. 4. – Issue 4. – Pp. 2013–2017.

134. Ki Y. A Novel Approach to Detect Malware Based on API Call Sequence Analysis / Y. Ki, E. Kim, H.K. Kim // International Journal of Distributed Sensor Networks - Special issue on Advanced Big Data Management and Analytics for Ubiquitous Sensors. – 2015. – Vol. 2015. – No 4. – Pp. 120-129.

135. Kim K. Malware detection based on dependency graph using hybrid genetic algorithm / K. Kim, B. Moon // Proceedings of the 12-th annual conference on genetic and evolutionary computation. – Portland (USA), July 07-11, 2010. – Pp. 1211-1218.

136. Knysz M. Good guys vs. Bot Guise: Mimicry attacks against fast-flux detection systems / M. Knysz, X. Hu, K. Shin // Proceedings IEEE INFOCOM. – Shanghai (China), April 10-15 2011. – Pp. 1844-1852.

137. Kolter J. Learning to detect and classify malicious executables in the wild / J. Kolter, M. Maloof // The Journal of Machine Learning Research. – 2006. – Vol. 7. – Pp. – 2721-2744.

138. Komar M. Compression of Network Traffic Parameters for Detecting Cyber Attacks Based on Deep Learning / M. Komar, A. Sachenko, V. Golovko, V. Dorosh // Proceedings of 2018 IEEE 9-th International Conference on Dependable

Systems Services and Technologies DESSERT '2018. – Kiev (Ukraine), May 24-27, 2018. – Pp. 44-47.

139. Komar M. High performance adaptive system for cyber attacks detection / M. Komar, V. Kochan, L. Dubchak, A. Sachenko, V. Golovko, S. Bezobrazov, I. Romanets // Proceedings of the 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. – Bucharest (Romania), 21-23 September, 2017. – Vol. 2. – Pp. 853-858.

140. Koret J. The antivirus hacker's handbook / J. Koret, E. Bachaalany. – John Wiley & Sons, 2015. – 384 p.

141. Koroniotis N. Towards Developing Network Forensic Mechanism for Botnet Activities in the IoT Based on Machine Learning Techniques / N. Koroniotis, N. Moustafa, E. Sitnikova, J. Slay // Proceedings of the 9th International Conference Mobile Networks and Management. – Melbourne (Australia), December 13-15, 2017. – Pp. 30-44.

142. Kotenko I. Experiments With Simulation Of Botnets And Defense Agent Teams / I. Kotenko // Proceedings of the 27th European Conference on Modelling and Simulation (ECMS 2013). – Ålesund (Norway), May 27-30, 2013. – Pp. -Proceedings edited by: W. Rekdalsbakken, R. T. Bye, H. Zhang, European Council for Modeling and Simulation. – Pp. 61-67.

143. Kotenko I. Simulation of botnets and protection mechanisms against them: software environment and experiments / I. Kotenko, A. Konovalov, A. Shorov // Proceedings of the 16th Nordic Conference on Secure IT-Systems. – Tallinn (Estonia), October 26-28, 2011. – Pp. 119-126.

144. Kotenko I. Simulation of Protection Mechanisms against Botnets on The Basis of “Nervous Network” Framework / I. Kotenko, A. Shorov // Proceedings of the 2nd International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2012). – Rome (Italy), July 28-31, 2012. – Pp. 164-169.

145. Kotenko I. Attack Detection in IoT Critical Infrastructures: A Machine Learning and Big Data Processing Approach / I. Kotenko, I. Saenko, A. Kushnerevich,

A. Branitskiy // 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP) Pavia, (Italy). - February 13, 2019. – Pp. 340-347.

146. Kotenko I. Modeling the Impact of Cyber Attacks / I. Kotenko, I. Saenko, O. Lauta // Cyber Resilience of Systems and Networks. – 2019. – Pp. 135-169.

147. Kuhrer M. Going Wild: Large-Scale Classification of Open DNS Resolvers / M. Kuhrer, T. Hupperich, J. Bushart, C. Rossow and T. Holz // ACM Internet Measurement Conference (IMC). – Tokyo (Japan), October 28. - 30, 2015. – Pp. 355-368.

148. Kuriakose J. Unknown Metamorphic Malware Detection: Modelling with Fewer Relevant Features and Robust Feature Selection Techniques / J. Kuriakose, P. Vinod // IAENG International Journal of Computer Science. – 2015. –Vol. 42. – Issue 2. – Pp. 139-151.

149. Kuriakose J. A Comparative Analysis of Attacks in Wireless Network / J. Kuriakose, MD Lakshmi, A Khan // Journal of Network Security. – 2016. - Vol. 2. – No. 3. – Pp. 22-28.

150. Lapsely D. Botnet Detection Based on Network Behavior / D. Lapsely, Robert Walsh, Carl Livadas, [et al] // Botnet Detection Countering the Largest Security Threat. – 2008. – Pp. 1-24.

151. Lee J. Detecting metamorphic malwares using code graphs / J. Lee, K. Jeong, H. Lee // Proceedings of the 10-th Symposium on Applied Computing. – New York (UAS), March 22-26, 2010. – Pp. 1970-1977.

152. Li S.H. A Network Behavior-Based Botnet Detection Mechanism Using PSO and K-means / S.H. Li, Y.C. Kao, Z.C. Zhang, Y.P. Chuang and D.C. Yen // Journal ACM Transactions on Management Information Systems. – 2015. – Vol. 6. – Issue 1. – Pp. 1-30.

153. Li Z. Honeynet-based Botnet Scan Traffic Analysis / Z. Li, A. Goyal, Y. Chen // Botnet Detection: Countering the Largest Security Threat. – 2008. – Pp. 25-44.

154. Li Z. Detection of suspicious domains through graph inference algorithm processing of host-domain contacts / Z. Li, A.M. Oprea, S.H. Chin, T.-F. Yen // Laboratory Inc.- 2017.- Pp. 578-586.

155. Li Z. An Attack-oriented Task Execution Collaborative Defense Protocol/ Z. Li, S. Lei, Y. Yu, X. Li, C. Xia // Proceedings of the 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing. – Chongqing (China), September 22 - 23, 2017. – Pp. 169-172.

156. Lim H. Detecting Malicious Behaviors of Software through Analysis of API Sequence k-grams / H. Lim // Computer Science and Information Technology. – 2016. – Vol. 4. – No 3. – Pp. 85-91.

157. Lim H. Suspicious traffic sampling for intrusion detection in software-defined networks/ H. Lim, T.Ha, S.Kim, N. An, J. Narantuya, C. Jeong, J.W.,Kim, - Gwangju Institute of Science and Technology (GIST). – 2016. – Pp. 172-182.

158. Lin D. Hunting for undetectable metamorphic viruses / D. Lin, M. Stamp // Journal in Computer Virology. – 2011. – Vol. 7. – Issue 3. – Pp. 201-214.

159. Lindorfer M. Detecting Environment-Sensitive Malware / M. Lindorfer, C. Kolbitsch, P.M. Comparetti // Proceedings of the 14-th International Workshop on Recent Advances in Intrusion Detection. – Menlo Park (USA), September 20-21, 2011. – Pp. 338-357.

160. Lindorfer M. GuardION: Practical Mitigation of DMA-Based Rowhammer Attacks on ARM/ M. Lindorfer, V. Veen, Y. Fratantonio, H. P. Pillai, G. Vigna, C. Kruegel, H.t Bos, K. Razavi // International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. – 2018. – Pp.92-113.

161. Lu C. P2P hierarchical botnet traffic detection using hidden Markov models / C. Lu, R. Brooks // Proceedings of the 2012 Workshop on Learning from Authoritative Security Experiment Results. – Arlington, Virginia (USA), July 18 - 19, 2012. – Pp. 41-46.

162. Lysenko S. Botnet detection technique based on support vector mashine / S. Lysenko, O. Savenko and A. Kryshchuk // 1-st International Academic Conference

"Science and Education in Australia, America and Eurasia: Fundamental and Applied Science". – Melbourne (Australia) June 25, 2014. – No. 1. – Pp. 24-31.

163. Lysenko S. DNS-based Anti-evasion Technique for Botnets Detection / S. Lysenko, O. Pomorova, O. Savenko, A. Kryshchuk and K. Bobrovnikova // Proceedings of the 8-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. – Warsaw (Poland), September 24-26, 2015. – Pp. 453-458.

164. Lysenko S. Information technology for botnets detection based on their behaviour in the corporate area network / S. Lysenko, O. Savenko, K. Bobrovnikova, A. Kryshchuk, B. Savenko // Communications in Computer and Information Science. – 2017. – Vol. 718. – Pp. 166-181.

165. Lysenko S. Self-adaptive system for the corporate area network resilience in the presence of botnet cyberattacks / S. Lysenko, O. Savenko, K. Bobrovnikova, A. Kryshchuk // Communications in Computer and Information Science. – 2018. – Vol. 860. – Pp. 385-401.

166. Mahaver D.K. Metamorphic malware detection using base malware identification approach / D.K. Mahaver, A. Negaraju // Security and Communications Networks. – 2014. – Vol. 7. – Issue 11. – Pp. 1719-1733.

167. Mahmoud M. A Survey on Botnet Architectures, Detection and Defences / M. Mahmoud, M.P. Nir, A. Matrawy // International Journal of Network Security. – 2014. – Vol. 17. – No.3. – Pp. 264-281.

168. Malan D.J. Rapid Detection of Botnets through Collaborative Networks of Peers / D.J. Malan. – Ph.D. Thesis. Harvard University, School of Engineering and Applied Sciences, Cambridge, Massachusetts, 2007. – 104 p.

169. Malan D.J. How to Achieve Early Botnet Detection at the Provider Level?/ D.J. Malan D, C. Dietz, A. Sperotto, G. Dreo, A. Pras// IFIP International Conference on Autonomous Infrastructure, Management and Security. – 2016. – Pp. 142-146.

170. Malwarebytes Endpoint Security [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://ru.malware bytes. com/business/endpoint security (Viewed on April 2, 2019). – Title from the screen.

171. Manasrah A.M. Detecting Botnet Activities Based on Abnormal DNS traffic / A.M. Manasrah, A. Hasan, O.A. Abouabdalla, S. Ramadass // International Journal of Computer Science and Information Security. – 2009. – Vol. 6. – No. 1. – Pp. 97-104.

172. Markowsky G. Distributed Malware Detection System Based on Decentralized Architecture in Local Area Networks / G. Markowsky, O. Savenko, A. Sachenko // Advances in Intelligent Systems and Computing. – 2019. – Vol. 871. – Pp.582-598.

173. Markowsky G. Distributed System for Detecting the Malware in LAN / G. Markowsky, O. Savenko, A. Sachenko // Proceedings of the 2018 IEEE 13th International Scientific and Technical Conference on Computer Science and Information Technologies (CSIT), CSIT'2018. – Lviv (Ukraine), September 11-14, 2018. – Pp. 306-309.

174. Markowsky G. The technique for metamorphic viruses' detection based on its obfuscation features analysis / G. Markowsky, O. Savenko, S. Lysenko, A. Nicheporuk // CEUR-WS. – 2018. – Vol. 2104. – Pp. 680-687.

175. Martinez-Bea S. Real-time malicious fast-flux detection using DNS and bot related features / S. Martinez-Bea, S. Castillo-Perez, J. Grcia-Alfaro // Proceedings of the 2013 Eleventh annual international conference on privacy, security and trust (PST). – Tarragona (Catalonia), July 10-12, 2013. – Pp. 369-372.

176. Martinsen E.A. Detection of Junk Instructions in Computer Viruses / E.A. Martinsen. – Master thesis, 2008. – 132 p.

177. Martynyuk O. Evolutionary Network Model of Testing of the Distributed Information Systems / O. Martynyuk, A. Sugak, D. Martynyuk, O. Drozd // Proceedings of the 2017 IEEE 9th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2017. – 2017. – Vol. 2. - Pp. 888-893.

178. McAfee [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http:// www.mcafee.com/ (Viewed on April 2, 2019). – Title from the screen.

179. Morales J.A. Analyzing DNS activities of bot processes / J.A. Morales, A. Al-Bataineh, S. Xu, R. Sandhu // Proceedings of the 4th International Conference on Malicious and Unwanted Software (MALWARE). – Montreal, Quebec (Canada), October 13-14, 2009. – Pp. 98-103.

180. Melnyk A. Self-Configurable FPGA-Based Computer Systems / A. Melnyk, V. Melnyk // Advances in Electrical and Computer Engineering. – 2013. - Vol. 13, № 2. - Pp. 33-38.

181. Melnyk A. Improvement of Heterogeneous Systems Efficiency Using Self-Configurable FPGA-based Computing / Melnyk, A., Melnyk // Proceedings of the First Workshop on Sustainable Ultrascale Computing Systems (NESUS 2014). - Porto, August 27-28, 2014. - Pp. 55-60.

182. Melnyk A. UNIX-like operating system extension for real-time FPGA-based SCCS support / Melnyk, A., Melnyk, V., Kit, A. // Proceedings of the 2017 IEEE 9th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2017. – 2017. – Vol.1- Pp. 20-25.

183. Melnyk A. Tasks scaling with Chameleon© C2HDL design tool in self-configurable Computer Systems based on partially reconfigurable FPGAs / Melnyk, A., Melnyk, V., Tsyhylyk, L. // Publishing House of Lviv Polytechnic National University. – 2016. – №1.

184. Mukhin V. Models for Analysis and Prognostication of the Indicators of the Distributed Computer Systems' Characteristics / V. Mukhin, H. Loutskii, O. Barabash, Y. Kornaga, V. Steshyn // International Review on Computers and Software (IRECOS). - Vol. 10, № 12, 2015. - Pp. 1216 – 1224.

185. Nachenberg, C. Understanding heuristics: Symantec's bloodhound technology / C. Nachenberg // Symantec White Paper Series. – 1998. – Vol. 34. – P. 17.

186. Nazario J. As the net churns: fast-flux botnet observations / J. Nazario, T. Holz // Conference on Malicious and Unwanted Software (Malware'08). – Alexandria, Virginia (USA). - October 7-8, 2008. – Pp. 24-31.

187. Network Admission Control [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://www.cisco. com/web/RU/products/hw/ wireless/secure/cnac.html (Viewed on April 2, 2019). – Title from the screen.

188. O'Kane P. SVM Training Phase Reduction Using Dataset Feature Filtering for Malware Detection / P. O'Kane, S. Sezer, K. McLaughlin, E. G. Im // Proceedings of the 2013 IEEE Transactions on Information Forensics and Security. – March 2013. – Pp. 500-509.

189. OpenDNS. Security Whitepaper. The Role of DNS in Botnet Command & Control [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https:// http://info.opendns.com/rs/opendns/images/ OpenDNS_Security Whitepaper-DNSRoleInBotnets.pdf (Viewed on April 2, 2019). – Title from the screen.

190. Panda Security [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.pandasecurity.com (Viewed on April 2, 2019). – Title from the screen.

191. Park Y. Fast malware classification by automated behavioral graph matching / Y. Park, D. Reeves, V. Mulukutla, B. Sundaravel // Proceedings of the 6-th Annual Workshop on Cyber Security and Information Intelligence Research. – New York (USA), April 21-23, 2010. – Pp. 214-230.

192. Park Y. Towards Effective Virtualization of Intrusion Detection Systems / Y. Park, N. Zhang, H. Li, H. Hu // Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization.- Scottsdale, Arizona (USA). - March 24, 2017. - Pp. 47-50.

193. Patanaik C. K. Obfuscated malware detection using API call dependency / C. K. Patanaik, F.A. Barbhuiya, S. Nandi // Proceedings of the 1-st International Conference on Security of Internet of Things. – Kollam (India), August 17-19, 2012. – Pp. 185-193.

194. Pek G. nEther: In-guest Detection of Out-of-the-guest Malware Analyzers / G. Pek, B. Bencsáth, L. Buttyán // Proceedings of the 4-th European Workshop on System Security. – Salzburg (Austria), No. 3, April 10, 2011. – Pp. 85-91.

195. Perdisci R. Early detection of malicious flux networks via large-scale passive DNS analysis / R. Perdisci, I. Corona, G. Giacinto // IEEE Transactions on Dependable and Secure Computing. – 2012. – Vol. 9. – Issue 5. – Pp. 714-726.

196. Perdisci R. Maxs: Scaling malware execution with sequential multi-hypothesis testing / R. Perdisci, P. Vadrevu // Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security. - Xi'an (China). - May 30-June 03, 2016. - Pp. 771-782.

197. Perriot F. Principles and practise of xraying / F. Perriot, P. Ferrie // Proceedings of the 14-th Virus Bulletin International Conference. – September 12, 2004. – Pp. 51-56.

198. Plohmann D. Botnets: Detection, Measurement, Disinfection & Defence / D. Plohmann, E. Gerhards-Padilla, F. Leder // European Network and Information Security Agency. – 2011. – 153 p.

199. Plohmann D. A Comprehensive Measurement Study of Domain Generating Malware/ D Plohmann, K Yakdan, M Klatt, J Bader, E. Gerhards-Padilla, F. Fkie // 25th {USENIX} Security Symposium ({USENIX} Security 16). – Austin, TX (USA). – August 10–12, 2016. – Pp. 263-278.

200. Pomorova O. A Technique for detection of bots which are using polymorphic code / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, A. Nicheporuk // Communications in Computer and Information Science. – 2014. – Vol. 431. – Pp. 265-276.

201. Pomorova O. A Technique for the Botnet Detection Based on DNS-Traffic Analysis / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, K. Bobrovnikova // Proceedings of the 22-nd International Conference Computer Networks. – Brunów (Poland), June 16-19, 2015, Vol. 522. – Pp. 127-138.

202. Pomorova O. Multi-Agent Based Approach for Botnet Detection in a Corporate Area Network Using Fuzzy Logic / O. Pomorova, O. Savenko, S. Lysenko,

A. Kryshchuk // Communications in Computer and Information Science. – 2013. – Vol. 370. – Pp. 243-254.

203. Pomorova O. Anti-evasion Technique for the Botnets Detection Based on the Passive DNS Monitoring and Active DNS Probing / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, K. Bobrovnikova // Proceedings of the 23-nd International Conference Computer Networks. – Brunów (Poland), June 14-17, 2016, Vol. 608. – Pp. 83-95.

204. Porras P. A multi-perspective analysis of the Storm (Peacomm) worm / P. Porras, H. Saidi, V. Yegneswaran – Technical report. Computer Science Laboratory: SRI International. – 2007. – 18 p.

205. Porras P. An Analysis of Conficker's Logic and Rendezvous Points / P. Porras, H. Saidi, V. Yegneswaran // SRI International report. – 2009.

206. Pronoza A. Visual Analysis of Information Dissemination Channels in Social Network for Protection Against Inappropriate Content / A. Pronoza, L. Vitkova, A. Chechulin, I. Kotenko // Proceedings of the Third International Scientific Conference Intelligent Information Technologies for Industry (IITI'18). – Sochi (Russia), September 17-21, 2019, Vol. 2. – Pp. 95-105.

207. Rad B.B. Camouflage in Malware: From Encryption to Metamorphism / B.B. Rad, M. Masrom, S. Ibrahim // International Journal of Computer Science and Network Security. – 2012. – Vol. 12. – Pp. 74-83.

208. Rad B.B. Evolution of Computer Virus Concealment and Anti-Virus Techniques: A Short Survey / B.B. Rad, M. Masrom, S. Ibrahim // International Journal of Computer Science Issues. – 2011. – Vol. 8. – No. 1 – Pp. 113-121.

209. Rad B.B. Metamorphic Virus Variants Classification Using Opcode Frequency Histogram / B.B. Rad, M. Masroom // Proceedings of the 14-th WSEAS international conference of computers. – Corfu Island (Greece), July 23-25, 2010. – Pp. 147-155.

210. Raffetseder T. Detecting System Emulators / T. Raffetseder, C. Kruege, E. Kirda // Proceedings of the 10-th international conference on Information Security. – Valparaíso (Chile), October 10-12, 2007. – Pp. 1-18.

211. Raiyn J. A survey of Cyber Attack Detection Strategies / J. Raiyn // International Journal of Security and Its Applications. – 2014. – Vol.8. – No.1. – Pp. 247-256.

212. Raiyn J. Introduction to Big Data Management Based on Agent Oriented Cyber Security / J. Raiyn // Journal of Telecommunications and Information Technology. – 2017. – Pp. 65-70.

213. Rajarajan M. Detection and prevention of botnets and malware in an enterprise network / M. Rajarajan, F. Piper, H. Wang et al // International Journal of Wireless and Mobile Computing. – 2012. – Vol. 5. – Issue 2. – Pp. 144-153.

214. Ramachandran A. Revealing botnet membership using DNSBL counter-intelligence / A. Ramachandran, N. Feamster, D. Dagon // Second Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI 2006). – San Jose, CA (USA), July 7, 2006. – Pp. 8-13.

215. Raphel J. Heterogeneous Opcode Space for Metamorphic Malware Detection / J. Raphel, P. Vinod // Arabian Journal for Science and Engineering. – 2017. – Vol. 42. – Issue 2. – Pp. 537-558.

216. Roshna R.S. Botnet Detection Using Adaptive Neuro Fuzzy Inference System / R.S. Roshna, E. Vinodh // International Journal of Engineering Research and Applications. – 2013. – Vol. 3. – Issue 2. – Pp. 1440-1445.

217. Rostami M. R. Botnet evolution: Network traffic indicators / M. R. Rostami, M. Eslahi, B. Shanmugam, Z. Ismail // Biometrics and Security Technologies (ISBAST). – Kuala Lumpur (Malaysia), August 26-27, 2014. – Pp. 274-279.

218. Runwal N. Opcode graph similarity and metamorphic detection / N. Runwal, R.M. Low, M. Stamp // Journal in Computer Virology. – 2012. – Vol. 8. – Issue 1. – Pp. 37-52.

219. Sami A. Malware detection based on mining API calls / A. Sami, B.Yadegari, H. Rahimi, N. Peiravian, S. Hashemi, A. Hamze // Proceedings of the 10-th Symposium on Applied Computing. – Sierre (Switzerland), March 22 - 26, 2010. – Pp. 1020-1025.

220. Sami A. SIDS: State-based intrusion detection for stage-based cyber physical systems / A. Khalili, A. Sami, A. Khozaei, S. Pouresmaeeli// International Journal of Critical Infrastructure Protection.- September, 2018 . – Vol. 22. – Pp. 113-124.

221. Santos I. Opcode sequences as representation of executables for data-mining-based unknown malware detection / I. Santos, F. Brezo, X. Ugarte-Pedrero, P.G. Bringas // Information Sciences. – 2013. – Vol. 231. – Pp. 64-82.

222. Sathyanarayan V. Sai Signature Generation and Detection of Malware Families / V. Sai Sathyanarayan, P. Kohli, B. Bruhadeshwar // Proceedings of the 13th Australasian conference on Information Security and Privacy. – Wollongong, NSW (Australia), July 7-9, 2008. – Pp. 336-349.

223. Savenko O. Botnet detection technique for corporate area network / O. Savenko, S. Lysenko, A. Kryshchuk, Y. Klots / Proceedings of the 7-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. – Berlin (Germany), September 12-14, 2013. – Pp. 363-368.

224. Savenko O. Approach for the Unknown Metamorphic Virus Detection / O. Savenko, S. Lysenko, A. Nicheporuk, B. Savenko // Proceedings of the 8-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. – Bucharest (Romania), September 21-23, 2017. – Pp. 71-76.

225. Savenko O. Intelligent method of the search of the trojan program in computer systems / O. Savenko, S. Lysenko // Праці IV міжнародної конференції “Сучасні комп'ютерні системи та мережі: розробка та використання” (ACSN'2009). – Львів, 17-19 грудня, 2009. – С.154-158.

226. Savenko O. Metamorphic Viruses’ Detection Technique Based on the Equivalent Functional Block Search / O. Savenko, S. Lysenko, A. Nicheporuk, B. Savenko // CEUR-WS. – 2017. – Vol. 1844. – Pp. 555-569.

227. Savenko O. Multi-agent based approach of botnet detection in computer systems / O. Savenko, S. Lysenko, A. Kryschuk // Communications in Computer and Information Science. – 2012. – Vol. 291. – Pp. 171-180.

228. Savenko O. Software for computer systems trojans diagnosing as a safety-case tool / O. Savenko, S. Lysenko // Proceedings of the 1-st International Workshop “Critical infrastructure safety and security (CrlSS-DESSERT'11)”. – Kirovograd, May 11-13, 2011. – Pp. 353-361.

229. Savenko O. The Technique for Computer Systems Trojan Diagnosis in the Monitor Mode / O. Savenko S. Lysenko // Proceedings of the 6-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. – Prague (Czech Republic), September 15-17, 2011. – Pp. 770-774.

230. Schiller C. Botnets the killer web app / C. Schiller, J. Binkley, G. Evron [et al]. – Amsterdam: Syngress. – 2007. – 480 p.

231. Schomp K. Towards a model of DNS client behavior / K. Schomp, M. Rabinovich, and M. Allman // International Conference on Passive and Active Network Measurement, Vol. 9631. – Heraklion, (Greece), 31 March. - 1 April, 2016. – Pp. 263-275.

232. Schultz M. Data mining methods for detection of new malicious executables / M. Schultz, E. Eskin, E. Zadok, S. J. Stolfo // Proceedings of Symposium on Security and Privacy. – Oakland (USA), May 14-16, 2000. – Pp. 38-49.

233. Securelist. FAQ: Disabling the new Hlux/Kelihos Botnet [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://securelist.com/blog/research/32634/faq-disabling-the-new-hluxkelihos-botnet-13/ (Viewed on April 2, 2019). – Title from the screen.

234. Shafiq M. Z. Comparative Study of Fuzzy Inference Systems, Neural Networks and Adaptive Neuro Fuzzy Infer-ence Systems for Portscan Detection / M. Z. Shafiq, M. Farooq, S. A Khayam // Applications of Evolutionary Computing. – 2008. – Pp. 52-61.

235. Shafiq M.Z. PE-Miner: Mining Structural Information to Detect Malicious Executables in Realtime / M.Z. Shafiq, S.M. Tabish, F. Mirza, M. Farooq // International Workshop on Recent Advances in Intrusion Detection. – Saint-Malo (France), 23-25 September, 2009. – Pp.121-141.

236. Sheng L. A Distributed Botnet Detecting Approach Based on Traffic Flow Analysis / L. Sheng, L. Zhiming, H. Jin, D. Gaoming, and H. Wen // Proceedings of the 2-nd International Conference Instrumentation, Measurement, Computer, Communication and Control (IMCCC). – Heilongjiang (China), 8-10 December 2012. – Pp. 124-128.

237. Shoham Y. Multiagent Systems Algorithmic, Game-Theoretic, and Logical Foundations / Y. Shoham, K. Leyton-Brown. – Cambridge University Press. – 2009. – 504 p.

238. Sikorski M. Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software: 1st Edition / M. Sikorski, A.Honig. – No Starch Press, 2012. – 802 p.

239. Sinclair G. The waledac protocol: The how and why / G. Sinclair, C. Nunnery, B.B.-H. Kang // Proceedings of the 4-th International Conference on Malicious and Unwanted Software, (MALWARE). – Montreal, QC, (Canada), October 13-14, 2009. – Pp.69-77.

240. Singh J. A Survey on Machine Learning Techniques for Intrusion Detection Systems / J. Singh, M. J. Nene // International Journal of Advanced Research in Computer and Communication Engineering. – 2013. – Vol. 2. – Issue 11. – Pp. 4349-4355.

241. Sochor T. Attractiveness Study of Honeypots and Honeynets in Internet Threat Detection / T. Sochor, M. Zuzcak // Proceedings of the 22-nd International Conference Computer Networks. – Brunów (Poland), June 16-19, 2015, Vol. 522. – Pp. 69-81.

242. Sochor T. High-Interaction Linux Honeypot Architecture in Recent Perspective / T. Sochor, M. Zuzcak // Proceedings of the 21-st International Conference Computer Networks. – Brunów (Poland), June 23-27, 2014, Vol. 431. – Pp. 118-131.

243. Sochor T. Behavioral Analysis of Bot Activity in Infected Systems Using Honeypots/ M. Zuzcak, T. Sochor// Proceedings of the 24-st International Conference on Computer Networks. – Springer (Cham), May 30, 2017, Vol. 718. – Pp. 118-133.

244. Sochor T. Analysis of attackers against windows emulating honeypots in various types of networks and regions/ T. Sochor, M. Zuzcak, P. Bujok// 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN). – Vienna (Austria), July 5-8, 2016. – Pp. 863-868.

245. Statistics [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.av-test.org/ (Viewed on April 2, 2019). – Title from the screen.

246. Stevanovic M. An analysis of network traffic classification for botnet detection / M. Stevanovic, J. M. Pedersen // Cyber Situational Awareness, Data Analytics and Assessment (CyberSA). – London (UK), June 8-9, 2015. – Pp. 1-8.

247. Stevanovic M. An approach for detection and family classification of malware based on behavioral analysis/ S. S. Hansen, T. M. T. Larsen, , M. Stevanovic// 2016 International Conference on Computing, Networking and Communications (ICNC). – Kauai, HI (USA), February 15-18, 2016. – Pp. 1-5.

248. Stone-Gross B. Your Botnet is My Botnet: Analysis of a Botnet Takeover / B. Stone-Gross, M. Cova, L. Cavallaro [et al] // Proceedings of the 16th ACM conference on Computer and communications security. – Chicago, Illinois (USA), November 09-13, 2009. – Pp. 635-647.

249. Sun M. Tracking you through DNS traffic: Linking user sessions by clustering with Dirichlet mixture model / M. Sun, G. Xu, J. Zhang, D. Kim // Proceedings of 20-th ACM International Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems. – Miami (US), November 24-26, 2017. – Pp. 303-310.

250. Sugak, A. Models of the Mutation and Immunity in Test Behavioral Evolution / A. Sugak, O. Martynyuk, O. Drozd, // Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2015. – 2015. – Vol.2 - Pp. 790-795.

251. Symantec [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.symantec.com (Viewed on April 2, 2019). – Title from the screen.

252. Symantec Endpoint Protection [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://www.anti-malware.ru/reviews/ Symantec_Endpoint_Protection (Viewed on April 2, 2019). – Title from the screen.

253. Symantec Global Internet Security. Internet security threat report 2014 (Volume XIX) [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.symantec.com/security_response/publications/ threatreport.jsp (Viewed on April 2, 2019). – Title from the screen.

254. Symantec Official Blog. Morto worm sets a (DNS) record [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: http://www.symantec.com/connect /blogs/morto-worm-sets-dns-record (Viewed on April 2, 2019). – Title from the screen.

255. Szor P. The Art of Computer Virus Research and Defense / P. Szor. – Addison-Wesley Professional, 2005. – 744 p.

256. Szor, P. Hunting for Metamorphic / P. Szor, P. Ferrie. – Symantec Security Response, 2001. – 24 p.

257. Tamboli T. Metamorphic code generation from LLVM bytecode / T. Tamboli, T.H. Austin, M. Stamp // Journal of Computer Virology and Hacking Techniques. – 2013. – Vol. 10. – Issue.3. – Pp. 177-187.

258. The Independent IT-Security Institute AV-TEST [Electronic resource] : [Web-site]. – Mode of access: http://www.av-test.org/. – Title from the screen.

259. Thunga, S.P. and Neelisetti, R.K. Identifying Metamorphic Virus Using N-Grams and Hidden Markov Model / S. P. Thunga, R.K. Neelisetti // Proceedings of International Conference on Advances in Computing, Communications and Informatics. – Kochi (India), August 10-13, 2015. – Pp. 2016-2022.

260. Toderici A.H. Chi-squared distance and metamorphic virus detection / A.H. Toderici, M. Stamp // Journal in Computer Virology. – 2013. – Vol. 9. – Issue 1. – Pp. 1-14.

261. Villamarın-Salomon R. Identifying Botnets Using Anomaly Detection Techniques Applied to DNS Traffic / R. Villamarın-Salomon, J.C. Brustoloni //

Proceedings of the Consumer Communications and Networking Conference. – Las Vegas, NV (USA), January 10-12, 2008. – Pp. 476-481.

262. Vinod P. Scattered Feature Space for Malware Analysis / P. Vinod, V. Laxmi, M.S. Gaur // Proceedings of the 1-st Advances in Computing and Communications. – Kochi (India), Vol. 190, July 22-24, 2011. – Pp. 562-571.

263. Vinod P. A machine learning based approach to detect malicious android apps using discriminant system calls / P. Vinod, A. Zemmari, M. Conti // Future Generation Computer Systems. – May 2019. – Vol. 94. – Pp. 333-350.

264. Vinod P. Identification of malicious android app using manifest and opcode features/ M.V. Varsha, P. Vinod, K.A. Dhanya// Journal of Computer Virology and Hacking Techniques. – July 1, 2017. – Vol 13. – Issue 2. – Pp. 125-138.

265. Virus Bulletin [Electronic resource]: [Web-site]. – Electronic data. – Mode of access: https://www.virusbulletin.com/testing/ (Viewed on March 25, 2019). – Title from the screen.

266. Vlam S. Current Trends and the Future of Metamorphic Malware Detection / S. Alam, I. Traore, I. Sogukpinar // Proceedings of the 7th International Conference on Security of Information and Networks. – Glasgow, (Scotland, UK), 09-11 September, 2014. – Pp. 4-11.

267. Vokorokos L. The obfuscation efficiency measuring schemes / L. Vokorokos, M. Uchnár, J. Hurtuk // Proceedings of the 20-th IEEE Jubilee International Conference. – Budapest (Hungary), 30 June. – 2 July, 2016. – Pp. 125-130.

268. Wang H. Real-time fast-flux identification via localized spatial geolocation detection / H. Wang, C. Mao, K. Wu, H. Lee // Proceedings of the Computer Software and Applications Conference (COMPSAC). – Izmir (Turkey), July 16-20, 2012. – Pp. 244-252.

269. Wang P. An advanced hybrid peer-to-peer botnet / P. Wang, S. Sparks, C. Zou // IEEE Transactions on Dependable and Secure Computing. – 2010. – №7. – Issue 2. – Pp. 113-127.

270. Wanga G. A new approach to intrusion detection using Artificial Neural Networks and Expert Systems with Applications / G. Wanga, J. Hao, J. Ma, L. Huang // An International Journal. – 2010. – Vol. 37. – Issue 9. – Pp. 6225-6232.

271. Wanga K. A fuzzy pattern-based filtering algorithm for botnet detection / K. Wanga, C.-Y. Huangb, S.-J. Lina, Y.-D. Lina // Computer Networks. 2011. – Vol. 55. – Issue 15. – Pp. 275-286.

272. Weimer F. Passive DNS replication / F/ Weimer // 17th Annual First Conference on Computer Security Incident Handling (FIRST 2005). – 2005. – P. 98.

273. Wong W. Hunting for metamorphic engines / W. Wong, M. Stamp // Journal in Computer Virology. – 2006. – Vol. 2. – Issue 3. – Pp. 211-229.

274. Wu W. Bot detection using unsupervised machine learning // W. Wu, J. Alvarez, C. Liu, H. M. Sun // Microsystem Technologies. – 2018. – Vol. 24. – No.1. – Pp. 209-217.

275. Xufang L. Mechanisms of Polymorphic and Metamorphic Viruses / L. Xufang, P. K. K. Loh // Proceedings of the 11-th Intelligence and Security Informatics Conference. – Athens (Greece), September 12-14, 2011. – Pp. 149-154.

276. Yadav S. Winning with DNS failures: Strategies for faster botnet detection / S.Yadav, A.L.N. Reddy // Proceedings of the 7th International ICST Conference on Security and Privacy in Communication Networks. – London (UK), September 7-9, 2011. – Pp. 446-459.

277. Yong Jin. Design of Detecting Botnet Communication by Monitoring Direct Outbound DNS Queries / Jin Yong, H. Ichise, K. Iida // Proceedings of the 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing. – New York, NY (USA), November 3-5, 2015. – Pp. 37-41.

278. You I. Malware Obfuscation Techniques: A Brief Survey / I. You, K. Yim // Proceedings of the 15-th International Conference on Broadband, Wireless Computing, Communication and Applications. – Fukuoka (Japan), November 4-6, 2010. – Pp. 297-300.

279. Zashcholkin K. The detection method of probable areas of hardware Trojans location in FPGA-based components of safety-critical systems /

K. Zashcholkin, O. Drozd, // Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies, DESSERT 2018. – 2018. – Pp. 212-217.

280. Zang X. Botnet Detection Through Fine Flow Classification / X. Zang, A. Tangpong, A. Kesidis, D. J. Miller // CSE Dept Technical Report No. CSE11-001. – 2011. – 17 p.

281. Zeidanloo H.R. Botnet Detection by Monitoring Similar Communication Patterns // H.R. Zeidanloo, A.B.A. Manaf // International Journal of Computer Science and Information Security. – 2010. – Vol. 7. – No. 3. – Pp. 36-45.

282. Zhang J. Detecting stealthy P2P botnets using statistical traffic fingerprints / J. Zhang, R. Perdisci, W. Lee, U. Sarfraz, and X. Luo // Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops. – Washington DC (USA), 27-30 June, 2011. – Pp. 121-132.

283. Zhang Q. MetaAware: Identifying Metamorphic Malware / Q. Zhang, D.S. Reeves // Proceedings of the 23-rd IEEE Annual Computer Security Applications Conference. – Florida (USA), December 10-14, 2007. – Pp. 411–420.

284. Zhaoy Y. BotGraph: Large Scale Spamming Botnet Detection / Y. Zhaoy, Y. Xie, F. Yu [et al] // Networked Systems Design and Implementation. – 2009. – Pp. 321-334.

285. Zheng J. Realtime DDoS Defense Using COTS SDN Switches via Adaptive Correlation Analysis / J. Zheng, Q. Li, G. Gu, J. Cao, D.K. Yau, J. Wu // IEEE Transactions on Information Forensics and Security. – 2018. – Vol. 13. – Issue 7. – Pp. 1838-1853.

286. Zhengbing Hu. Stochastic RA-Network for the nodes functioning analysis in the distributed computer systems / Zhengbing Hu, V. Mukhin, H. Loutskii, Y. Kornaga // International Journal of Computer Network and Information Security (IJCNIS). - Vol.8, № 6, June 2016. – Pp. 1-8.

287. Zhengbing Hu. The scheduler for the grid-system based on the parameters monitoring of the computer components / Zhengbing Hu, V. Mukhin, Y. Kornaga,

O. Herasymenko, Y. Bazaka // Eastern-European Journal of Enterprise Technologies. – Vol. 1, № 2 (85). – 2017. – Pp. 31- 39.

288. Zhengbing Hu. Analytical Assessment of Security Level of Distributed and Scalable Computer Systems / Zhengbing Hu, V. Mukhin, O. Barabash, Y. Kornaga, O. Herasymenko, Y. Lavrenko // International Journal of Intelligent Systems and Applications. – Vol. 8, № 12. – 2016. – Pp. 57 – 64.

289. Zhou C. A. Self-Healing, Self-Protecting Collaborative Intrusion Detection Architecture to Trace-Back Fast-Flux Phishing Domains / C. Zhou, C. Leckie, S. Karunasekera, T. Peng // Proceedings of the Network Operations and Management Symposium Workshops. – Salvador Da Bahia (Brazil) April 7-11, 2008. – Pp. 321-327.

290. Zuo Z. Some further theoretical results about computer viruses / Z. Zuo, M. Zhou // Computer Journal. – 2004. – Vol. 47, No. 6. – Pp. 627-633.

291. Anycoin.news. Новый троян крадет данные криптобиржевых аккаунтов Mac-пользователей [Электронный ресурс]: [Веб-сайт]. – Электронные данные – Режим доступа: https://anycoin.news/2019/02/ 01/cookieminer/ (дата обращения 25.03.2019). – Название с экрана.

292. CHIP. TPM 2.0 — крипточип для Windows [Электронный ресурс]: [Веб-сайт]. – Электронные данные – Режим доступа: https://ichip.ru/tpm-2-0-kriptochip-dlya-windows.html (дата обращения 25.03.2019). – Название с экрана.

293. Internetua. Red Hat, CentOS и Debian десять лет разрешали нелегально стать администратором [Электронный ресурс]: [Веб-сайт]. – Электронные данные – Режим доступа: http://internetua.com/red-hat-centos-i-debian-desyat-let-razreshali-nelegalno-stat-administratorom (дата обращения 25.03.2019). – Название с экрана.

294. Internetua. В Сети обнаружен ботнет из более чем 40 тыс. серверов, модемов и IoT-устройств [Электронный ресурс]: [Веб-сайт]. – Электронные данные – Режим доступа: http://internetua.com/v-seti-obnarujen-botnet-iz-bolee-csem-40-ts-serverov-modemov-i-iot-ustroistv (дата обращения 25.03.2019). – Название с экрана.

295. Internetua. Мощный ботнет перехватывает трафик, предназначенный для бразильских банков [Электронный ресурс]: [Веб-сайт]. – Электронные данные – Режим доступа: http://internetua.com/mosxnyi-botnet-perehvatyvaet-trafik-prednaznacsennyi-dlya-brazilskih-bankov (дата обращения 25.03.2019). – Название с экрана.

296. Internetua. На уязвимые маршрутизаторы GPON ведут охоту сразу 5 ботнетов [Электронный ресурс]: [Веб-сайт]. – Электронные данные – Режим доступа: http://internetua.com/na-uyazvime-marshrutizator-gpon-vedut-ohotu-srazu-5-botnetov (дата обращения 25.03.2019). – Название с экрана.

297. Internetua. Новый троян использует антивирусы для кражи данных [Электронный ресурс]: [Веб-сайт]. – Электронные данные – Режим доступа: http://internetua.com/novyi-troyan-ispolzuet-antivirusy-dlya-kraji-dannyh (дата обращения 25.03.2019). – Название с экрана.

298. Internetua. Обнаружен новый вирус для взлома криптокошельков [Электронный ресурс]: [Веб-сайт]. – Электронные данные – Режим доступа: http://internetua.com/obnarujen-novyi-virus-dlya-vzloma-kriptokoshelkov (дата обращения 25.03.2019). – Название с экрана.

299. Internetua. Обнаружен новый опасный вирус на Android [Электронный ресурс]: [Веб-сайт]. – Электронные данные – Режим доступа: http://internetua.com/obnarujen-novyi-opasnyi-virus-na-android (дата обращения 25.03.2019). – Название с экрана.

300. Internetua. Уязвимость в Facebook позволяла скомпрометировать учетные записи пользователей [Электронный ресурс]: [Веб-сайт]. – Электронные данные – Режим доступа: http://internetua.com/uyazvimost-v-facebook-pozvolyala-skomprometirovat-ucsetnye-zapisi-polzovatelei (дата обращения 25.03.2019). – Название с экрана.

301. Mediasat. ТВ, радиовещание и телекоммуникации. Зафиксирована первая атака сети ботов, имитирующих запросы от смарт-телевизоров [Электронный ресурс]: [Веб-сайт]. – Электронные данные – Режим доступа:

http://mediasat.info/2018/11/20/first-botnet-attack-on-smart-tv/ (дата обращения 25.03.2019). – Название с экрана.

302. Znaj.ua. Видалити не можна залишити: путінські хакери народили небезпечний вірус [Електронний ресурс]: [Веб-сайт]. – Електронні дані – Режим доступу: https://znaj.ua/world/177015-vidaliti-ne-mozhna-zalishiti-putinski-hakeri-narodili-nebezpechniy-virus (дата звернення 25.03.2019). – Назва з екрану.

303. А. с. 80223 Україна. Комп'ютерна програма пошуку та визначення еквівалентних функціональних блоків у виконуваних файлах для ідентифікації ознак метаморфних вірусів в локальних комп'ютерних мережах / А. О. Нічепорук, О. С. Савенко, С. М. Лисенко. 2018.

304. А. с. 80445 Україна. Розподілена комп'ютерна програма для виявлення зловмисного програмного забезпечення в локальних обчислювальних мережах на основі аналізу поведінкових сигнатур / О. С. Савенко. 2018.

305. Берников А. Р. Поиск вредоносных программ в распределенных тренажерах с использованием технологии нечеткой логики / А. Р. Берников, Р. П. Графов, С. Н. Лысенко, О. С. Савенко // Информационные технологии. – 2011. – № 10. – С. 42-47.

306. Бёрнс Б. Распределенные системы. Паттерны проектирования / Б. Бёрнс. – СПб.: Питер. – 2016. – 224 с.

307. Браницкий А. А. Обнаружение сетевых атак на основе комплексирования нейронных, иммунных и нейронечетких классификаторов / А.А. Браницкий, И. В. Котенко // Информационно-управляющие системы. – 2015. – № 4 (77). – С. 69-77.

308. Буквы. Киберполиция разоблачила мужчин, совершавших DDoS-атаки на украинские сайты [Электронный ресурс]: [Веб-сайт]. – Электронные данные – Режим доступа: https://bykvu.com/bukvy/108105-kiberpolitsiya-raz-oblachila-muzhchin-sovershavshikh-ddos-ataki-na-ukrainskie-sajty (дата обращения 25.03.2019). – Название с экрана.

309. Буквы. Эксперты ESET обнаружили в Украине новый вирус [Электронный ресурс]: [Веб-сайт]. – Электронные данные – Режим доступа:

https://bykvu.com/bukvy/101469-eksperty-eset-obnaruzhili-v-ukraine-novyj-virus (дата обращения 25.03.2019). – Название с экрана.

310. Вавренюк А. Б. Разрушающие программные воздействия: Учебно-методическое пособие / А. Б. Вавренюк, Н. П. Васильев, Е. В. Вельмякина и др.; под ред. М.А. Иванова. – М.: НИЯУ "МИФИ". – 2011. – 328 с.

311. Вікіпедія. Ботнет [Електронний ресурс]: [Веб-сайт]. – Електронні дані – Режим доступу: https://uk.wikipedia.org/wiki/%D0%91%D0%BE%D1%82%D0%BD%D0%B5%D1%82 (дата звернення 25.03.2019). – Назва з екрану.

312. Гатчин Ю. А. Основы информационной безопасности: учебное пособие / Ю. А. Гатчин, Е. В. Климова. – СПб.: СПбГУИТМО. – 2009. – 84 с.

313. Графов Р. П. Забезпечення надійності розподільних мереж на основі динамічної структурної реконфігурації / Графов Р. П., Савенко О. С. // Інформаційні технології та комп'ютерна інженерія. – 2005. – № 3. – С. 241-246.

314. Графов Р. П. Моделирование и тестирование распределенных динамических систем / Р. П. Графов, В. М. Локазюк, О. С. Савенко // Вісник Хмельницького національного університету. Серія: Технічні науки. – 2005. – № 4. – Т. 1. – С.175-181.

315. Дойникова Е. В. Методика выбора защитных мер для реагирования на инциденты безопасности в компьютерных сетях на основе показателей защищенности / Е. В. Дойникова, И. В. Котенко // Материалы 9-й конференции "Информационные технологии в управлении" (ИТУ-2016). – Санкт-Петербург, 4-6 октября 2016. – С. 700-705.

316. Дудикевич В. Б. Дослідження моделі оцінювання живучості систем захисту інформації корпоративних мереж зв'язку за допомогою мереж Петрі / В. Б. Дудикевич, Ю. Р. Гарасим // Вісник Національного університету "Львівська політехніка". – 2011. – № 699 : Інформаційні системи та мережі. – С. 82-94.

317. Дудикевич В. Б. Квінтесенція інформаційної безпеки кіберфізичної системи / В. Б. Дудикевич, Г. В. Микитин, А. І. Ребець // Вісник Національного університету «Львівська політехніка». Інформаційні системи та мережі. — Львів: Видавництво Львівської політехніки, 2018. – № 887. – С. 58–68.

318. Дудикевич В. Б. Парадигма та концепція побудови багаторівневої комплексної системи безпеки кіберфізичних систем / В. Б. Дудикевич, В. М. Максимович, Г. В. Микитин // Вісник Національного університету «Львівська політехніка». Автоматика, вимірювання та керування. – 2015. – № 821. – С. 3–7.

319. Економічна правда. Кібернапади на фінансові системи стануть більш руйнівними – експерт [Електронний ресурс]: [Веб-сайт]. – Електронні дані – Режим доступу: https://www.epravda.com.ua/news/2019/02/10/645137/ (дата звернення 25.03.2019). – Назва з екрану.

320. Загородна Н.В. Аналіз анти-бот стратегій, що грунтуються на інтерактивному підтвердженні участі людини / Н. Загородна, О. Маєвський, Р. Козак // Матеріали III-ої міжнародної науково-технічної конференції "Захист інформації і безпека інформаційних систем". – Львів, 05-06 червня, 2014. – С. 42-43.

321. Закон України «Про державну таємницю»: за станом на 22 січ. 2019 р. / Верховна Рада України. – Офіц. вид. – Київ: Парлам. вид-во, 1994. – 11 с.

322. Закон України «Про захист інформації в автоматизованих системах»: за станом на 22 січ. 2019 р. / Верховна Рада України. – Офіц. вид. – Київ: Парлам. вид-во, 1994. – 9 с.

323. Закон України «Про інформацію»: за станом на 22 січ. 2019 р. / Верховна Рада України. – Офіц. вид. – Київ: Парлам. вид-во, 1992. – 19 с.

324. Закон України «Про науково-технічну інформацію»: за станом на 22 січ. 2019 р. / Верховна Рада України. – Офіц. вид. – Київ: Парлам. вид-во, 1994. – 9 с.

325. Закон України «Про основні засади забезпечення кібербезпеки України»: за станом на 22 січ. 2019 р. / Верховна Рада України. – Офіц. вид. – Київ: Парлам. вид-во, 2017. – 31 с.

326. Захист інформації. Технічний захист інформації. Основні положення: ДСТУ 3396.0-96. [Чинний від 01.01.1997] – Київ: Держспоживстандарт України, 1997. – 37 с. – (Національний стандарт України).

327. Інформаційне агентство Уніан. Bloomberg: Китай міг вбудувати шпигунські чіпи у сервери Apple і Amazon [Електронний ресурс]: [Веб-сайт]. – Електронні дані – Режим доступу: https://www.unian.ua/world/10286736-bloomberg-kitay-mig-vbuduvati-shpigunski-chipi-u-serveri-apple-i-amazon.html (дата звернення 25.03.2019). – Назва з екрану.

328. Інформаційні технології. Методи захисту. Звід правил для управління інформаційною безпекою (ISO/IEC 27002:2005, MOD): ДСТУ СУІБ 2.0/ISO/IEC 27002:2010. – К.: Національний банк України, 2010. – 163 с.

329. Інформаційні технології. Методи захисту. Система управління інформаційною безпекою (ISO/IEC 27001:2013; Cor 1:2014, IDT): ДСТУ ISO/IEC 27001:2015. – [Чинний від 2015–18–12]. – К.: Держспоживстандарт України, 2016. – 28 с. – (Національний стандарт України).

330. Касперський Е.В. Компьютерное зловредство / Е.В. Касперський. – СПб.: Питер. – 2009. – 208 с.

331. Кирсанов Э. А. Обработка информации в пространственно-распределенных системах радиомониторинга. Статистический и нейросетевой подходы / Э. А. Кирсанов, А. А. Сирота. – М.: ФИЗМАТЛИТ. – 2013. – 344 с.

332. Котенко И. В. Исследовательское моделирование бот-сетей и механизмов защиты от них / И. В. Котенко, А. М. Коновалов, А. В. Шоров // Приложение к журналу «Информационные технологии». – М.: Издательство Новые технологии, 2012. – № 1. – 32 с.

333. Котенко И. В. Многоагентное моделирование механизмов защиты от распределенных компьютерных атак / И. В. Котенко, А. В. Уланов // Информационные технологии. – 2009. – № 2. – С. 38-44.

334. Котенко И. В. Система сбора, хранения и обработки информации и событий безопасности на основе средств Elastic Stack / И. В. Котенко, А. А. Кулешов, И. А. Ушаков // Труды СПИИРАН. – 2017. – № 5 (54). – С. 5-34.

335. Котенко И. В. Оценка киберустойчивости компьютерных сетей на основе моделирования кибератак методом преобразования стохастических сетей

/ И. В. Котенко, И. Б. Саенко, М. А. Коцыняк, О. С. Лаута // Труды СПИИРАН. – 2017. – Вып. 55. – С. 160-184.

336. Левшун Д. С. Архитектура комплексной системы безопасности / Д. С. Левшун, А. А. Чечулин, И. В. Котенко // Материалы 25-й научно-технической конференции “Методы и технические средства обеспечения безопасности информации”. – Санкт-Петербург, 4-7 июля, 2016. – С. 53-54.

337. Локазюк В. М. Модель прогнозування стану взаємоблокування процесів комп’ютерної системи / В. М. Локазюк, О. С. Савенко, С. В. Мостовий // Вісник Вінницького політехнічного інституту. – 2011. – № 4. – С. 130-133.

338. Мартинюк А. Н. Поведенческий рабочий контроль сетевых компьютерных систем / А. Н. Мартынюк, Ахмеш Тамим, Д. А. Мартынюк, А. В. Дрозд // Електротехнічні та комп’ютерні системи. – 2018. – № 28 (104). – С. 201-207.

339. Маевский Д. А. Метрики «оттенков зеленого» в программном обеспечении / Д. А. Маевский, Е. Д. Маевская, А. В. Дрозд // Радиоэлектронные и компьютерные системы. – № 6 (70). – Харьков: ХАИ. – 2014. – С. 120 – 124.

340. Мельник А. О. Персональні суперкомп’ютери: архітектура, проектування, застосування: монографія / А. О. Мельник, В. А. Мельник. — Львів: Видавництво Львівської політехніки, 2013. — 516 с.

341. НД ТЗІ 1.1-002-99. Загальні положення щодо захисту інформації в комп'ютерних системах від несанкціонованого доступу.

342. НД ТЗІ 2.5-004-99. Критерії оцінки захищеності інформації в комп’ютерних системах від несанкціонованого доступу.

343. НД ТЗІ 2.5-005-99. Класифікація автоматизованих систем і стандартні функціональні профілі захищеності оброблюваної інформації від несанкціонованого доступу.

344. НД ТЗІ 2.5-008-02. Вимоги із захисту конфіденційної інформації від несанкціонованого доступу під час оброблення в автоматизованих системах класу 2.

345. НД ТЗІ 2.5-010-03. Вимоги до захисту інформації WEB-сторінки від несанкціонованого доступу.

346. НД ТЗІ 3.6-001-2000. Технічний захист інформації. Комп'ютерні системи. Порядок створення, впровадження, супроводження та модернізації засобів технічного захисту інформації від несанкціонованого доступу.

347. Нейман Дж. Теория самовоспроизводящихся автоматов / Дж. Фон Нейман. – М.: Мир. – 1971. – 281 с.

348. Нічепорук А. О. Моделі життєвого циклу поліморфних вірусів / А. О. Нічепорук, О. С. Савенко // Комп'ютерно-інтегровані технології: освіта, наука, виробництво. – 2013. – № 11. – С. 64-71.

349. Нормативні документи в галузі технічного захисту інформації (НД ТЗІ) та державні стандарти України (ДСТУ) стосовно створення і функціонування КСЗІ: НД ТЗІ 3.7-003-05. Порядок проведення робіт із створення комплексної системи захисту інформації в інформаційно-телекомунікаційній системі.

350. Пат. на корисну модель 108238 Україна, МПК G06F 21/55 Мультиагентний спосіб локалізації бот-мереж у корпоративних комп'ютерних мережах / О. В. Поморова, О. С. Савенко, А. Ф. Крищук, С. М. Лисенко, К. Ю. Бобровнікова, А. О. Нічепорук; заявник і патентовласник Хмельницький національний університет. – № u201600127; заявл. 04.01.2016; опубл. 11.07.2016, Бюл. № 13/2016.

351. Пат. на корисну модель 118456 Україна, МПК G06F 21/55 Спосіб виявлення метаморфних вірусів на основі статистичних метрик для визначення еквівалентних функціональних програмних блоків / О. С. Савенко, С. М. Лисенко, К. Ю. Бобровнікова, А. О. Нічепорук, Б. О. Савенко; заявник і патентовласник Хмельницький національний університет. – № u201701743; заявл. 23.02.2017; опубл. 10.08.2017, Бюл. № 15/2017.

352. Пат. на корисну модель 118663 Україна, МПК G06F 21/55 Спосіб ідентифікації бот-мереж у корпоративних комп'ютерних мережах на основі аналізу DNS-трафіку / О. С. Савенко, С. М. Лисенко, К. Ю. Бобровнікова, А. О. Нічепорук, Б. О. Савенко; заявник і патентовласник Хмельницький

національний університет. – № u201612041; заявл. 28.11.2016; опубл. 28.08.2017, Бюл. № 16/2017.

353. Пат. на корисну модель Україна, МПК G06F 21/55 Спосіб організації взаємодії компонентів децентралізованих розподілених систем виявлення зловмисного програмного забезпечення на основі рівнів їх безпеки в локальних комп'ютерних мережах / О. С. Савенко; заявник Хмельницький національний університет. – № u201612041; заявл. 26.12.2018; позитивне рішення 19.03.2019.

354. Погребенник В. Д. Активні методи виявлення ботнет-мереж / В. Д. Погребенник, П. Т. Хромчак // Вісник Національного університету «Львівська політехніка». – 2013. – № 774. – С. 3-9.

355. Погребенник В. Д. Пасивні методи виявлення ботнет-мереж / В. Д. Погребенник, П. Т. Хромчак // Вісник Національного університету «Львівська політехніка». – 2012. – № 741. – С. 97-104.

356. Погребенник В. Д. Розроблення моделі системи виявлення центрів управління ботнет-мережами / В. Д. Погребенник, П. Т. Хромчак // Вісник Національного університету «Львівська політехніка». – 2009. – № 639. – С. 117-123.

357. Подловченко Р. И. Использование алгебраических моделей программ для обнаружения метаморфного вредоносного кода / Р. И. Подловченко, Н. Н. Кузюрин, В. С. Щербина, В. А. Захаров // Фундаментальная и прикладная математика. – 2009. –№ 5. – С. 181-198.

358. Подпружников Ю. Классификация методов обнаружения неизвестного вредоносного программного обеспечения / Ю. Подпружников // Сборник трудов Современные тенденции технических наук. Уфа (РФ), октябрь, 2011. – С. 22-25.

359. Про затвердження Правил забезпечення захисту інформації в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах : Постанова Кабінету Міністрів України від 29 бер. 2006 р. № 373 // Урядовий кур'єр. – 2006. – 18 квітня. – С. 73–74.

360. Рувинская В. М. Эвристические методы детектирования вредоносных программ на основе сценариев / В. М. Рувинская, Е. Л. Беркович, А. А. Лотоцкий // Штучний інтелект. – 2008. – № 3. – С. 197-207.

361. Савенко О. С. CAN-архітектура в промислових мережах / О. С. Савенко, А. Ф. Крищук // Вісник Хмельницького національного університету. Технічні науки. – 2011. - № 4. - С. 219-223.

362. Савенко О. С. Адаптивна інформаційна технологія виявлення троянських програм в комп'ютерних системах / О. С. Савенко, С. М. Лисенко // Комп'ютинг. – 2011. – Т. 10, № 3. – С. 85-92.

363. Савенко О. С. Архітектура багаторівневої програмної системи виявлення шкідливого програмного забезпечення в локальних комп'ютерних мережах. / О. С. Савенко, В. І. Грибинчук, М. О. Кульчицький // Комп'ютерно-інтегровані технології: освіта, наука, виробництво. – 2018. – № 30-31. – С. 132-140.

364. Савенко О. С. Архітектура розподіленої багаторівневої системи виявлення шкідливого програмного забезпечення в локальних комп'ютерних мережах / О. С. Савенко // Вчені записки Таврійського національного університету. Технічні науки. – 2018. – Т. 29 (68). – № 2. – С. 172-181.

365. Savenko O. Interoperability of distributed multiple system for malware detection based on components levels of safety / O. Savenko // Проблеми інформаційних технологій. – 2018. – № 24. – С. 78-92.

366. Савенко О. С. Генерація моделей комп'ютерних вірусних програм в системі оцінки достовірності результатів роботи антивірусних засобів / О. С. Савенко, С. В. Мостовий // Вісник Хмельницького національного університету. Технічні науки. – 2005. – № 4, т. 1. – С. 198-200.

367. Савенко О. С. Діагностування комп'ютерних систем на наявність шкідливого програмного забезпечення на основі антивірусної мультиагентної системи / О. С. Савенко, С. М. Лисенко, А. Ф. Крищук // Вісник Національного університету «Львівська політехніка». – 2011. – № 5. – С. 99-105.

368. Савенко О. С. Дослідження антивірусних технологій діагностування комп’ютерних систем на наявність шкідливого програмного забезпечення / О. С. Савенко, С. М. Лисенко, А. Ф. Крищук // Праці XII міжнародної науково-практичної конференції «Сучасні інформаційні та електронні технології» (СІЕТ-2011), 27-31 травня 2011. – Одеса. – С. 165-166.

369. Савенко О. С. Дослідження методів антивірусного діагностування комп’ютерних мереж / О. С. Савенко, С. М. Лисенко // Вісник Хмельницького національного університету. Технічні науки. – 2007. – № 2, т. 2. – С. 120-126.

370. Савенко О. С. Дослідження та аналіз блокування процесів в комп'ютерній системі / О. С. Савенко, Ю. П. Кльоц, С. В. Мостовий // Вісник Хмельницького національного університету. Технічні науки. – 2007. – № 3, т. 1. – С. 248-251.

371. Савенко О. С. Життєвий цикл процесів комп'ютерної системи / О. С. Савенко, С. В. Мостовий // Радіоелектронні і комп'ютерні системи. – 2010. – № 7 (48). – С. 35-38.

372. Савенко О. С. Критерії класифікації методів виявлення шкідливого програмного забезпечення / О. С. Савенко // Вісник Хмельницького національного університету. Технічні науки. – 2018. – № 1. – С. 23-27.

373. Савенко О. С. Метод антивірусного діагностування персональних комп’ютерів на основі матриць інцидентності / О. С. Савенко, В. М. Джулій, В. М. Стецюк // Вісник Технологічного університету Поділля. Технічні науки. – 2000. – № 3. - С. 136-139.

374. Савенко О. С. Метод взаємодії компонентів розподіленої системи виявлення зловмисного програмного забезпечення в локальних обчислювальних мережах / О. С. Савенко, А. О. Нічепорук // Контроль і управління в складних системах (КУСС-2018): збірник тез доповіді XIV Міжнародної конференції. – Вінниця, 15-17 жовтня, 2018. – С. 37.

375. Савенко О. С. Методи та засоби антивірусного комбінованого діагностування персональних комп'ютерів: автореф. дис. на здобуття наук.

ступеня канд. техн. наук : спец. 05.13.13 – обчислювальні машини, системи і мережі / О. С. Савенко. – Вінниця, 1999. – 20 с.

376. Савенко О. С. Методи лексичного аналізу технічного завдання на розробку програмного забезпечення / О. С. Савенко, Ю. П. Кльоц, В. С. Шевцов // Вісник Хмельницького національного університету. Технічні науки. – 2011. - № 5 - С. 167-172.

377. Савенко О.С. Методика генерації матриць інцидентності для використання в антивірусних програмних засобах // Вісник Технологічного університету Поділля. Технічні науки. – 2003. – № 3, т. 2. – С. 48-56.

378. Савенко О.С. Моделі рівнів поліморфних комп'ютерних вірусів // О. С. Савенко, С. М. Лисенко, А. О. Нічепорук / Вісник Вінницького політехнічного інституту. – Вінниця: ВНТУ. – 2015. – № 2. – С. 75-83.

379. Савенко О. С. Метод виявлення бот-мереж розподіленими системами на основі самоорганізації / О. С. Савенко // Штучний інтелект. – 2018. – № 4 (82). – С. 58-72.

380. Савенко О. С. Модель та архітектура розподіленої багаторівневої системи виявлення шкідливого програмного забезпечення в локальних комп'ютерних мережах / О. С. Савенко // Вісник Хмельницького національного університету. Технічні науки. – 2018. – № 2. – С. 153-163.

381. Савенко О. С. Розподілена апаратно-програмна система та методи захисту інформації в комп'ютерних системах локальних мереж / О. С. Савенко // Наукові праці Чорноморського національного університету ім. П. Могили. Комп'ютерні технології. – 2018. – Т. 320. Вип. 308. – С. 72-75.

382. Савенко О.С. Побудова адаптивної інформаційної технології діагностування комп'ютерних систем на наявність троянських програм / О. С. Савенко, С. М. Лисенко // Вісник Вінницького політехнічного інституту. - 2011.- № 5. - С. 93-99.

383. Савенко О. С. Прогнозування потрапляння процесів у стан взаємоблокування / О. С. Савенко, С. В. Мостовий // Вісник Вінницького політехнічного інституту. – 2013. – № 2. – С. 81-86.

384. Савенко О. С. Програмне забезпечення інформаційної технології моделювання поширення вірусних кодів в гетерогенних мережах / О. С. Савенко // Вісник Хмельницького національного університету. Технічні науки. – 2017. – № 1. – С. 144-148.

385. Савенко О. С. Распределенная многоуровневая сетевая система обнаружения метаморфных вирусов в локальных компьютерных сетях / О. С. Савенко // Вестник Брестского государственного технического университета (физика, математика, информатика). – 2017. – № 5 (107). – С. 40-44.

386. Савенко О. С. Розподілена система виявлення зловмисного програмного забезпечення та метод взаємодії її компонент в локальних мережах / О. С. Савенко // Інформаційні технології та взаємодії: матеріали доповідей V Міжнародної науково-практичної конференції. – Київ, 20-21 листопада 2018. – С. 308-309.

387. Савенко О. С. Формалізація шкідливого програмного забезпечення в локальних комп'ютерних мережах / О. С. Савенко // Вісник Хмельницького національного університету. Серія: Технічні науки. – 2018. – № 3. – С. 145-154.

388. Савенко О. С. Формалізоване структурування шкідливого програмного забезпечення на основі алгебраїчних систем / О. С. Савенко // Вимірювальна та обчислювальна техніка в технологічних процесах. – 2018. – № 1. – С. 67-72.

389. Савенко О. С. Формування сигнатури поведінки програм на основі трасування API викликів / О. С. Савенко, А. О. Нічепорук, А. А. Нічепорук, Ю. О. Нічепорук // Електротехнічні та комп'ютерні системи. – 2018. – № 29 (105). – С. 67-77.

390. Савенко О. С. Функційна модель бота як складової ботнет-мережі / О. С. Савенко, С. М. Лисенко, А. Ф. Крищук // Тези доповідей І Міжнародна науково-технічна конференція: Захист інформації і безпека інформаційних систем. – Львів, 2012. – С. 123-124.

391. Сиротинський О. І. Захист інформаційних систем від мережевих DDOS атак на основі марківської моделі поведінки ботнету // О. І. Сиротинський, І. С. Беляєв, Т. А. Максимюк, М. І. Олексін // Вісник Національного університету «Львівська політехніка». – 2012. – № 738. – С. 192-197.

392. Собейкис В. Г. Азбука хакера. Компьютерная вирусология / В. Г. Собейкис. – М.: Майор. – 2006. – 512 с.

393. Ткаченко В. А. Комп'ютерні мережі та телекомунікації: навч. посіб. / В. А. Ткаченко, О. В. Касілов, В. А. Рябик. – Харків: НТУ "ХПІ", 2011. – 224 с.

394. Українська правда. Затримали хакерів, які "чистили" банківські рахунки і переводили кошти в криптовалюту [Електронний ресурс]: [Веб-сайт]. – Електронні дані – Режим доступу: https://www.pravda.com.ua /news/2019/01/10/7203471/ (дата звернення 25.03.2019). – Назва з екрану.

395. Українська правда. Найвідомішим у Darknet ресурсом заправляли українці – Кіберполіція [Електронний ресурс]: [Веб-сайт]. – Електронні дані – Режим доступу: https://www.pravda.com.ua/news/2019/01/28/7205116/ (дата звернення 25.03.2019). – Назва з екрану.

396. Фокус. Хакеры через WordPress пытались атаковать сайт ЦИК, - СБУ [Электронный ресурс]: [Веб-сайт]. – Электронные данные – Режим доступа: https://focus.ua/ukraine/422005-xakery-cherez-wordpress-pytalis-atakovat- sajt-cik--sbu.html (дата обращения 25.03.2019). – Название с экрана.

397. Чиж В. М. Класифікація атак на бездротові сенсорні мережі і шляхи їх візуалізації / В. М. Чиж, М. П. Карпінський, С. М. Балабан // Вісник Тернопільського національного технічного університету. – 2012. – № 2. – С. 191-197.

398. Широчин В. П. Управління ризиками безпеки корпоративних інформаційних систем на основі Balanced Scorecard / В. П. Широчин, В. Є. Мухін, А. М. Волокита // Зб. праць Міжнародної науково-практичної конференції «Розподілені комп’ютерні системи РКС-2010». – Київ, 6-8 квітня 2010. – С. 156-157.

399. Чженбин Ху. Управление ресурсами распределенной компьютерной системы с учетом уровня доверия к вычислительным компонентам / Ху Чженбин, В. Е. Мухин, Я. И. Корнага, О. Ю. Герасименко // Кибернетика и системный анализ. – № 2, т. 53. – 2017. – С. 168–180.

ДОДАТКИ

Додаток А

Таблиці

Таблиця А.1

Значення функції F_{A_i}

Елементи множини R_i	Кодований перехід	Стан
$r_{i,0}$	«1000j», 0000i	$s_{i,0}$
$r_{i,1}$	0000i, 0001i	$s_{i,1}$
$r_{i,2}$	0000i, 0011i	$s_{i,3}$
$r_{i,3}$	0000i, 0101i	$s_{i,5}$
$r_{i,4}$	0000i, 1000i	$s_{i,8}$
$r_{i,5}$	0001i, 0000i	$s_{i,0}$
$r_{i,6}$	0011, 0000i	$s_{i,0}$
$r_{i,7}$	0101i, 0000i	$s_{i,0}$
$r_{i,8}$	1000i, 0000i	$s_{i,0}$
$r_{i,9}$	0001i, 0010i	$s_{i,2}$
$r_{i,10}$	0010i, 0001i	$s_{i,1}$
$r_{i,11}$	0010i, 1000i	$s_{i,8}$
$r_{i,12}$	1000i, 0010i	$s_{i,2}$
$r_{i,13}$	0011i, 0100i	$s_{i,4}$
$r_{i,14}$	0100i, 0011i	$s_{i,3}$
$r_{i,15}$	0100i, 1000i	$s_{i,8}$
$r_{i,16}$	1000i, 0100i	$s_{i,4}$
$r_{i,17}$	0001i, 0111i	$s_{i,7}$
$r_{i,18}$	0111i, 0001i	$s_{i,1}$
$r_{i,19}$	0111i, 1000i	$s_{i,8}$
$r_{i,20}$	1000i, 0111i	$s_{i,7}$
$r_{i,21}$	0001i, 0010i	$s_{i,1}$
$r_{i,22}$	0011i, 0110i	$s_{i,6}$
$r_{i,23}$	0110i, 0011i	$s_{i,5}$
$r_{i,24}$	1000i, 0110i	$s_{i,8}$
$r_{i,25}$	0110i, 1000i	$s_{i,6}$
$r_{i,26}$	1000i, «0000j»	$s_{i,0}$

Позначення:
i, j – номери програмних модулів РБС;
$r_{i,0}$ - $r_{i,26}$ – елементи множини R_i;
$s_{i,0}$ - $s_{i,8}$ – стани програмного модуля.

Таблиця А.2

Стратегії для подій у КС локальної мережі

№ з/п	Встановлення в КС до (1)/після(0) встановлення ПМ або ПЗ вузла бот-мережі		Стартовий контроль в КС: так(1) / ні(0)		Атака з цієї КС: так(1) / ні(0)	Атака з цієї КС на КС цієї ж мережі: так(1) / ні(0)	Атака на КС: так(1) / ні(0)	Атака з КС цієї ж мережі: так(1) / ні(0)	Наявність файлового ЗПЗ, встановленого до(1) / після(0) встановлення ПМ	Стратегії
	ПМ	Б	ПМ	Б						
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	1	0	0	1	1	1	1	1	1	1
2	1	0	0	1	1	1	0	1	1	1
3	1	0	0	1	1	1	1	0	1	1
4	1	0	0	1	1	1	0	0	1	1
5	1	0	0	1	1	1	1	1	0	1
6	1	0	0	1	1	1	1	0	0	1
7	1	0	0	1	1	1	0	1	0	1
8	1	0	0	1	1	1	0	0	0	1
9	1	0	0	1	1	0	1	1	1	1
10	1	0	0	1	1	0	0	1	1	1
11	1	0	0	1	1	0	1	0	1	1
12	1	0	0	1	1	0	0	0	1	1
13	1	0	0	1	1	0	1	1	0	1
14	1	0	0	1	1	0	1	0	0	1
15	1	0	0	1	1	0	0	1	0	1
16	1	0	0	1	1	0	0	0	0	1
17	1	0	0	1	0	1	1	1	1	1
18	1	0	0	1	0	1	0	1	1	1
19	1	0	0	1	0	1	1	0	1	1
20	1	0	0	1	0	1	0	0	1	1
21	1	0	0	1	0	1	1	1	0	1
22	1	0	0	1	0	1	1	0	0	1
23	1	0	0	1	0	1	0	1	0	1
24	1	0	0	1	0	1	0	0	0	1
25	1	0	0	1	0	0	1	1	1	1
26	1	0	0	1	0	0	0	1	1	1
27	1	0	0	1	0	0	1	0	1	1
28	1	0	0	1	0	0	0	0	1	1
29	1	0	0	1	0	0	1	1	0	1
30	1	0	0	1	0	0	1	0	0	1
31	1	0	0	1	0	0	0	1	0	1
32	1	0	0	1	0	0	0	0	0	1

Продовження таблиці А.2

1	2	3	4	5	6	7	8	9	10	11
33	1	0	0	0	1	1	1	1	1	1
34	1	0	0	0	1	1	0	1	1	1
35	1	0	0	0	1	1	1	0	1	1
36	1	0	0	0	1	1	0	0	1	1
37	1	0	0	0	1	1	1	1	0	1
38	1	0	0	0	1	1	1	0	0	1
39	1	0	0	0	1	1	0	1	0	1
40	1	0	0	0	1	1	0	0	0	1
41	1	0	0	0	1	0	1	1	1	1
42	1	0	0	0	1	0	0	1	1	1
43	1	0	0	0	1	0	1	0	1	1
44	1	0	0	0	1	0	0	0	1	1
45	1	0	0	0	1	0	1	1	0	1
46	1	0	0	0	1	0	1	0	0	1
47	1	0	0	0	1	0	0	1	0	1
48	1	0	0	0	1	0	0	0	0	1
49	1	0	0	0	0	1	1	1	1	1
50	1	0	0	0	0	1	0	1	1	1
51	1	0	0	0	0	1	1	0	1	1
52	1	0	0	0	0	1	0	0	1	1
53	1	0	0	0	0	1	1	1	0	1
54	1	0	0	0	0	1	1	0	0	1
55	1	0	0	0	0	1	0	1	0	1
56	1	0	0	0	0	1	0	0	0	1
57	1	0	0	0	0	0	1	1	1	1
58	1	0	0	0	0	0	0	1	1	1
59	1	0	0	0	0	0	1	0	1	1
60	1	0	0	0	0	0	0	0	1	1
61	1	0	0	0	0	0	1	1	0	1
62	1	0	0	0	0	0	1	0	0	1
63	1	0	0	0	0	0	0	1	0	1
64	1	0	0	0	0	0	0	0	0	1
65	1	0	1	0	1	1	1	1	1	1
66	1	0	1	0	1	1	1	0	1	1
67	1	0	1	0	1	1	1	1	0	1
68	1	0	1	0	1	1	1	0	0	1
69	1	0	1	0	1	1	1	1	1	1
70	1	0	1	0	1	1	1	1	0	1
71	1	0	1	0	1	1	1	0	1	1
72	1	0	1	0	1	1	1	0	0	1
73	1	0	1	0	1	1	0	1	1	1
74	1	0	1	0	1	1	0	0	1	1
75	1	0	1	0	1	1	0	1	0	1
76	1	0	1	0	1	1	0	0	0	1
77	1	0	1	0	1	1	0	1	1	1
78	1	0	1	0	1	1	0	1	0	1
79	1	0	1	0	1	1	0	0	1	1
80	1	0	1	0	1	1	0	0	0	1
81	1	0	1	0	1	0	1	1	1	1
82	1	0	1	0	1	0	1	0	1	1
83	1	0	1	0	1	0	1	1	0	1
84	1	0	1	0	1	0	1	0	0	1
85	1	0	1	0	1	0	1	1	1	1
86	1	0	1	0	1	0	1	1	0	1
87	1	0	1	0	1	0	1	0	1	1

Продовження таблиці А.2

1	2	3	4	5	6	7	8	9	10	11
88	1	0	1	0	1	0	1	0	0	1
89	1	0	1	0	1	0	0	1	1	1
90	1	0	1	0	1	0	0	0	1	1
91	1	0	1	0	1	0	0	1	0	1
92	1	0	1	0	1	0	0	0	0	1
93	1	0	1	0	1	0	0	1	1	1
94	1	0	1	0	1	0	0	1	0	1
95	1	0	1	0	1	0	0	0	1	1
96	1	0	1	0	1	0	0	0	0	1
97	1	0	1	0	0	1	1	1	1	1
98	1	0	1	0	0	1	1	0	1	1
99	1	0	1	0	0	1	1	1	0	1
100	1	0	1	0	0	1	1	0	0	1
101	1	0	1	0	0	1	1	1	1	1
102	1	0	1	0	0	1	1	1	0	1
103	1	0	1	0	0	1	1	0	1	1
104	1	0	1	0	0	1	1	0	0	1
105	1	0	1	0	0	1	0	1	1	1
106	1	0	1	0	0	1	0	0	1	1
107	1	0	1	0	0	1	0	1	0	1
108	1	0	1	0	0	1	0	0	0	1
109	1	0	1	0	0	1	0	1	1	1
110	1	0	1	0	0	1	0	1	0	1
111	1	0	1	0	0	1	0	0	1	1
112	1	0	1	0	0	1	0	0	0	1
113	1	0	1	0	0	0	1	1	1	1
114	1	0	1	0	0	0	1	0	1	1
115	1	0	1	0	0	0	1	1	0	1
116	1	0	1	0	0	0	1	0	0	1
117	1	0	1	0	0	0	1	1	1	1
118	1	0	1	0	0	0	1	1	0	1
119	1	0	1	0	0	0	1	0	1	1
120	1	0	1	0	0	0	1	0	0	1
121	1	0	1	0	0	0	0	1	1	1
122	1	0	1	0	0	0	0	0	1	1
123	1	0	1	0	0	0	0	1	0	1
124	1	0	1	0	0	0	0	0	0	1
125	1	0	1	0	0	0	0	1	1	1
126	1	0	1	0	0	0	0	1	0	1
127	1	0	1	0	0	0	0	0	1	1
128	1	0	1	0	0	0	0	0	0	1
129	1	1	0	1	1	1	1	1	1	1
130	1	1	0	1	1	1	0	1	1	1
131	1	1	0	1	1	1	1	0	1	1
132	1	1	0	1	1	1	0	0	1	1
133	1	1	0	1	1	1	1	1	0	1
134	1	1	0	1	1	1	1	0	0	1
135	1	1	0	1	1	1	0	1	0	1
136	1	1	0	1	1	1	0	0	0	1
137	1	1	0	1	1	0	1	1	1	1
138	1	1	0	1	1	0	0	1	1	1
139	1	1	0	1	1	0	1	0	1	1
140	1	1	0	1	1	0	0	0	1	1
141	1	1	0	1	1	0	1	1	0	1
142	1	1	0	1	1	0	1	0	0	1

Продовження таблиці А.2

1	2	3	4	5	6	7	8	9	10	11
143	1	1	0	1	1	0	0	1	0	1
144	1	1	0	1	1	0	0	0	0	1
145	1	1	0	1	0	1	1	1	1	1
146	1	1	0	1	0	1	0	1	1	1
147	1	1	0	1	0	1	1	0	1	1
148	1	1	0	1	0	1	0	0	1	1
149	1	1	0	1	0	1	1	1	0	1
150	1	1	0	1	0	1	1	0	0	1
151	1	1	0	1	0	1	0	1	0	1
152	1	1	0	1	0	1	0	0	0	1
153	1	1	0	1	0	0	1	1	1	1
154	1	1	0	1	0	0	0	1	1	1
155	1	1	0	1	0	0	1	0	1	1
156	1	1	0	1	0	0	0	0	1	1
157	1	1	0	1	0	0	1	1	0	1
158	1	1	0	1	0	0	1	0	0	1
159	1	1	0	1	0	0	0	1	0	1
160	1	1	0	1	0	0	0	0	0	1
161	1	1	0	0	1	1	1	1	1	1
162	1	1	0	0	1	1	0	1	1	1
163	1	1	0	0	1	1	1	0	1	1
164	1	1	0	0	1	1	0	0	1	1
165	1	1	0	0	1	1	1	1	0	1
166	1	1	0	0	1	1	1	0	0	1
167	1	1	0	0	1	1	0	1	0	1
168	1	1	0	0	1	1	0	0	0	1
169	1	1	0	0	1	0	1	1	1	1
170	1	1	0	0	1	0	0	1	1	1
171	1	1	0	0	1	0	1	0	1	1
172	1	1	0	0	1	0	0	0	1	1
173	1	1	0	0	1	0	1	1	0	1
174	1	1	0	0	1	0	1	0	0	1
175	1	1	0	0	1	0	0	1	0	1
176	1	1	0	0	1	0	0	0	0	1
177	1	1	0	0	0	1	1	1	1	1
178	1	1	0	0	0	1	0	1	1	1
179	1	1	0	0	0	1	1	0	1	1
180	1	1	0	0	0	1	0	0	1	1
181	1	1	0	0	0	1	1	1	0	1
182	1	1	0	0	0	1	1	0	0	1
183	1	1	0	0	0	1	0	1	0	1
184	1	1	0	0	0	1	0	0	0	1
185	1	1	0	0	0	0	1	1	1	1
186	1	1	0	0	0	0	0	1	1	1
187	1	1	0	0	0	0	1	0	1	1
188	1	1	0	0	0	0	0	0	1	1
189	1	1	0	0	0	0	1	1	0	1
190	1	1	0	0	0	0	1	0	0	1
191	1	1	0	0	0	0	0	1	0	1
192	1	1	0	0	0	0	0	0	0	1
193	1	1	1	0	1	1	1	1	1	1
194	1	1	1	0	1	1	1	0	1	1
195	1	1	1	0	1	1	1	1	0	1
196	1	1	1	0	1	1	1	0	0	1
197	1	1	1	0	1	1	1	1	1	1

Продовження таблиці А.2

1	2	3	4	5	6	7	8	9	10	11
198	1	1	1	0	1	1	1	1	0	1
199	1	1	1	0	1	1	1	0	1	1
200	1	1	1	0	1	1	1	0	0	1
201	1	1	1	0	1	1	0	1	1	1
202	1	1	1	0	1	1	0	0	1	1
203	1	1	1	0	1	1	0	1	0	1
204	1	1	1	0	1	1	0	0	0	1
205	1	1	1	0	1	1	0	1	1	1
206	1	1	1	0	1	1	0	1	0	1
207	1	1	1	0	1	1	0	0	1	1
208	1	1	1	0	1	1	0	0	0	1
209	1	1	1	0	1	0	1	1	1	1
210	1	1	1	0	1	0	1	0	1	1
211	1	1	1	0	1	0	1	1	0	1
212	1	1	1	0	1	0	1	0	0	1
213	1	1	1	0	1	0	1	1	1	1
214	1	1	1	0	1	0	1	1	0	1
215	1	1	1	0	1	0	1	0	1	1
216	1	1	1	0	1	0	1	0	0	1
217	1	1	1	0	1	0	0	1	1	1
218	1	1	1	0	1	0	0	0	1	1
219	1	1	1	0	1	0	0	1	0	1
220	1	1	1	0	1	0	0	0	0	1
221	1	1	1	0	1	0	0	1	1	1
222	1	1	1	0	1	0	0	1	0	1
223	1	1	1	0	1	0	0	0	1	1
224	1	1	1	0	1	0	0	0	0	1
225	1	1	1	0	0	1	1	1	1	1
226	1	1	1	0	0	1	1	0	1	1
227	1	1	1	0	0	1	1	1	0	1
228	1	1	1	0	0	1	1	0	0	1
229	1	1	1	0	0	1	1	1	1	1
230	1	1	1	0	0	1	1	1	0	1
231	1	1	1	0	0	1	1	0	1	1
232	1	1	1	0	0	1	1	0	0	1
233	1	1	1	0	0	1	0	1	1	1
234	1	1	1	0	0	1	0	0	1	1
235	1	1	1	0	0	1	0	1	0	1
236	1	1	1	0	0	1	0	0	0	1
237	1	1	1	0	0	1	0	1	1	1
238	1	1	1	0	0	1	0	1	0	1
239	1	1	1	0	0	1	0	0	1	1
240	1	1	1	0	0	1	0	0	0	1
241	1	1	1	0	0	0	1	1	1	1
242	1	1	1	0	0	0	1	0	1	1
243	1	1	1	0	0	0	1	1	0	1
244	1	1	1	0	0	0	1	0	0	1
245	1	1	1	0	0	0	1	1	1	1
246	1	1	1	0	0	0	1	1	0	1
247	1	1	1	0	0	0	1	0	1	1
248	1	1	1	0	0	0	1	0	0	1
249	1	1	1	0	0	0	0	1	1	1
250	1	1	1	0	0	0	0	0	1	1
251	1	1	1	0	0	0	0	1	0	1
252	1	1	1	0	0	0	0	0	0	1

Продовження таблиці А.2

1	2	3	4	5	6	7	8	9	10	11
253	1	1	1	0	0	0	0	1	1	1
254	1	1	1	0	0	0	0	1	0	1
255	1	1	1	0	0	0	0	0	1	1
256	1	1	1	0	0	0	0	0	0	1
257	0	0	0	1	1	1	1	1	1	2
258	0	0	0	1	1	1	0	1	1	2
259	0	0	0	1	1	1	1	0	1	2
260	0	0	0	1	1	1	0	0	1	2
261	0	0	0	1	1	1	1	1	0	2
262	0	0	0	1	1	1	1	0	0	2
263	0	0	0	1	1	1	0	1	0	2
264	0	0	0	1	1	1	0	0	0	2
265	0	0	0	1	1	0	1	1	1	2
266	0	0	0	1	1	0	0	1	1	2
267	0	0	0	1	1	0	1	0	1	2
268	0	0	0	1	1	0	0	0	1	2
269	0	0	0	1	1	0	1	1	0	2
270	0	0	0	1	1	0	1	0	0	2
271	0	0	0	1	1	0	0	1	0	2
272	0	0	0	1	1	0	0	0	0	2
273	0	0	0	1	0	1	1	1	1	2
274	0	0	0	1	0	1	0	1	1	2
275	0	0	0	1	0	1	1	0	1	2
276	0	0	0	1	0	1	0	0	1	2
277	0	0	0	1	0	1	1	1	0	2
278	0	0	0	1	0	1	1	0	0	2
279	0	0	0	1	0	1	0	1	0	2
280	0	0	0	1	0	1	0	0	0	2
281	0	0	0	1	0	0	1	1	1	2
282	0	0	0	1	0	0	0	1	1	2
283	0	0	0	1	0	0	1	0	1	2
284	0	0	0	1	0	0	0	0	1	2
285	0	0	0	1	0	0	1	1	0	2
286	0	0	0	1	0	0	1	0	0	2
287	0	0	0	1	0	0	0	1	0	2
288	0	0	0	1	0	0	0	0	0	2
289	0	0	0	0	1	1	1	1	1	2
290	0	0	0	0	1	1	0	1	1	2
291	0	0	0	0	1	1	1	0	1	2
292	0	0	0	0	1	1	0	0	1	2
293	0	0	0	0	1	1	1	1	0	2
294	0	0	0	0	1	1	1	0	0	2
295	0	0	0	0	1	1	0	1	0	2
296	0	0	0	0	1	1	0	0	0	2
297	0	0	0	0	1	0	1	1	1	2
298	0	0	0	0	1	0	0	1	1	2
299	0	0	0	0	1	0	1	0	1	2
300	0	0	0	0	1	0	0	0	1	2
301	0	0	0	0	1	0	1	1	0	2
302	0	0	0	0	1	0	1	0	0	2
303	0	0	0	0	1	0	0	1	0	2
304	0	0	0	0	1	0	0	0	0	2
305	0	0	0	0	0	1	1	1	1	2
306	0	0	0	0	0	1	0	1	1	2
307	0	0	0	0	0	1	1	0	1	2

Продовження таблиці А.2

1	2	3	4	5	6	7	8	9	10	11
308	0	0	0	0	0	1	0	0	1	2
309	0	0	0	0	0	1	1	1	0	2
310	0	0	0	0	0	1	1	0	0	2
311	0	0	0	0	0	1	0	1	0	2
312	0	0	0	0	0	1	0	0	0	2
313	0	0	0	0	0	0	1	1	1	2
314	0	0	0	0	0	0	0	1	1	2
315	0	0	0	0	0	0	1	0	1	2
316	0	0	0	0	0	0	0	0	1	2
317	0	0	0	0	0	0	1	1	0	2
318	0	0	0	0	0	0	1	0	0	2
319	0	0	0	0	0	0	0	1	0	2
320	0	0	0	0	0	0	0	0	0	2
321	0	0	1	0	1	1	1	1	1	3
322	0	0	1	0	1	1	1	0	1	3
323	0	0	1	0	1	1	1	1	0	3
324	0	0	1	0	1	1	1	0	0	3
325	0	0	1	0	1	1	1	1	1	3
326	0	0	1	0	1	1	1	1	0	3
327	0	0	1	0	1	1	1	0	1	3
328	0	0	1	0	1	1	1	0	0	3
329	0	0	1	0	1	1	0	1	1	3
330	0	0	1	0	1	1	0	0	1	3
331	0	0	1	0	1	1	0	1	0	3
332	0	0	1	0	1	1	0	0	0	3
333	0	0	1	0	1	1	0	1	1	3
334	0	0	1	0	1	1	0	1	0	3
335	0	0	1	0	1	1	0	0	1	3
336	0	0	1	0	1	1	0	0	0	3
337	0	0	1	0	1	0	1	1	1	3
338	0	0	1	0	1	0	1	0	1	3
339	0	0	1	0	1	0	1	1	0	3
340	0	0	1	0	1	0	1	0	0	3
341	0	0	1	0	1	0	1	1	1	4
342	0	0	1	0	1	0	1	1	0	4
343	0	0	1	0	1	0	1	0	1	4
344	0	0	1	0	1	0	1	0	0	4
345	0	0	1	0	1	0	0	1	1	4
346	0	0	1	0	1	0	0	0	1	4
347	0	0	1	0	1	0	0	1	0	4
348	0	0	1	0	1	0	0	0	0	4
349	0	0	1	0	1	0	0	1	1	4
350	0	0	1	0	1	0	0	1	0	4
351	0	0	1	0	1	0	0	0	1	4
352	0	0	1	0	1	0	0	0	0	4
353	0	0	1	0	0	1	1	1	1	4
354	0	0	1	0	0	1	1	0	1	4
355	0	0	1	0	0	1	1	1	0	4
356	0	0	1	0	0	1	1	0	0	4
357	0	0	1	0	0	1	1	1	1	4
358	0	0	1	0	0	1	1	1	0	4
359	0	0	1	0	0	1	1	0	1	4
360	0	0	1	0	0	1	1	0	0	4
361	0	0	1	0	0	1	0	1	1	4
362	0	0	1	0	0	1	0	0	1	4

Продовження таблиці А.2

1	2	3	4	5	6	7	8	9	10	11
363	0	0	1	0	0	1	0	1	0	4
364	0	0	1	0	0	1	0	0	0	4
365	0	0	1	0	0	1	0	1	1	4
366	0	0	1	0	0	1	0	1	0	4
367	0	0	1	0	0	1	0	0	1	4
368	0	0	1	0	0	1	0	0	0	4
369	0	0	1	0	0	0	1	1	1	4
370	0	0	1	0	0	0	1	0	1	4
371	0	0	1	0	0	0	1	1	0	4
372	0	0	1	0	0	0	1	0	0	4
373	0	0	1	0	0	0	1	1	1	4
374	0	0	1	0	0	0	1	1	0	4
375	0	0	1	0	0	0	1	0	1	4
376	0	0	1	0	0	0	1	0	0	4
377	0	0	1	0	0	0	0	1	1	4
378	0	0	1	0	0	0	0	0	1	4
379	0	0	1	0	0	0	0	1	0	4
380	0	0	1	0	0	0	0	0	0	4
381	0	0	1	0	0	0	0	1	1	4
382	0	0	1	0	0	0	0	1	0	4
383	0	0	1	0	0	0	0	0	1	4
384	0	0	1	0	0	0	0	0	0	4
385	0	1	0	1	1	1	1	1	1	4
386	0	1	0	1	1	1	0	1	1	4
387	0	1	0	1	1	1	1	0	1	4
388	0	1	0	1	1	1	0	0	1	4
389	0	1	0	1	1	1	1	1	0	4
390	0	1	0	1	1	1	1	0	0	4
391	0	1	0	1	1	1	0	1	0	4
392	0	1	0	1	1	1	0	0	0	4
393	0	1	0	1	1	0	1	1	1	4
394	0	1	0	1	1	0	0	1	1	4
395	0	1	0	1	1	0	1	0	1	4
396	0	1	0	1	1	0	0	0	1	4
397	0	1	0	1	1	0	1	1	0	4
398	0	1	0	1	1	0	1	0	0	4
399	0	1	0	1	1	0	0	1	0	4
400	0	1	0	1	1	0	0	0	0	4
401	0	1	0	1	0	1	1	1	1	4
402	0	1	0	1	0	1	0	1	1	4
403	0	1	0	1	0	1	1	0	1	4
404	0	1	0	1	0	1	0	0	1	4
405	0	1	0	1	0	1	1	1	0	4
406	0	1	0	1	0	1	1	0	0	4
407	0	1	0	1	0	1	0	1	0	4
408	0	1	0	1	0	1	0	0	0	4
409	0	1	0	1	0	0	1	1	1	4
410	0	1	0	1	0	0	0	1	1	4
411	0	1	0	1	0	0	1	0	1	4
412	0	1	0	1	0	0	0	0	1	4
413	0	1	0	1	0	0	1	1	0	4
414	0	1	0	1	0	0	1	0	0	4
415	0	1	0	1	0	0	0	1	0	4
416	0	1	0	1	0	0	0	0	0	4
417	0	1	0	0	1	1	1	1	1	4

Продовження таблиці А.2

1	2	3	4	5	6	7	8	9	10	11
418	0	1	0	0	1	1	0	1	1	4
419	0	1	0	0	1	1	1	0	1	4
420	0	1	0	0	1	1	0	0	1	4
421	0	1	0	0	1	1	1	1	0	4
422	0	1	0	0	1	1	1	0	0	4
423	0	1	0	0	1	1	0	1	0	4
424	0	1	0	0	1	1	0	0	0	4
425	0	1	0	0	1	0	1	1	1	4
426	0	1	0	0	1	0	0	1	1	4
427	0	1	0	0	1	0	1	0	1	4
428	0	1	0	0	1	0	0	0	1	4
429	0	1	0	0	1	0	1	1	0	4
430	0	1	0	0	1	0	1	0	0	4
431	0	1	0	0	1	0	0	1	0	4
432	0	1	0	0	1	0	0	0	0	4
433	0	1	0	0	0	1	1	1	1	4
434	0	1	0	0	0	1	0	1	1	4
435	0	1	0	0	0	1	1	0	1	4
436	0	1	0	0	0	1	0	0	1	4
437	0	1	0	0	0	1	1	1	0	4
438	0	1	0	0	0	1	1	0	0	4
439	0	1	0	0	0	1	0	1	0	4
440	0	1	0	0	0	1	0	0	0	4
441	0	1	0	0	0	0	1	1	1	4
442	0	1	0	0	0	0	0	1	1	4
443	0	1	0	0	0	0	1	0	1	4
444	0	1	0	0	0	0	0	0	1	4
445	0	1	0	0	0	0	1	1	0	4
446	0	1	0	0	0	0	1	0	0	4
447	0	1	0	0	0	0	0	1	0	4
448	0	1	0	0	0	0	0	0	0	4
449	0	1	1	0	1	1	1	1	1	4
450	0	1	1	0	1	1	1	0	1	4
451	0	1	1	0	1	1	1	1	0	4
452	0	1	1	0	1	1	1	0	0	4
453	0	1	1	0	1	1	1	1	1	4
454	0	1	1	0	1	1	1	1	0	4
455	0	1	1	0	1	1	1	0	1	4
456	0	1	1	0	1	1	1	0	0	4
457	0	1	1	0	1	1	0	1	1	4
458	0	1	1	0	1	1	0	0	1	4
459	0	1	1	0	1	1	0	1	0	4
460	0	1	1	0	1	1	0	0	0	4
461	0	1	1	0	1	1	0	1	1	4
462	0	1	1	0	1	1	0	1	0	4
463	0	1	1	0	1	1	0	0	1	4
464	0	1	1	0	1	1	0	0	0	4
465	0	1	1	0	1	0	1	1	1	4
466	0	1	1	0	1	0	1	0	1	4
467	0	1	1	0	1	0	1	1	0	4
468	0	1	1	0	1	0	1	0	0	4
469	0	1	1	0	1	0	1	1	1	4
470	0	1	1	0	1	0	1	1	0	4
471	0	1	1	0	1	0	1	0	1	4
472	0	1	1	0	1	0	1	0	0	4

Продовження таблиці А.2

1	2	3	4	5	6	7	8	9	10	11
473	0	1	1	0	1	0	0	1	1	4
474	0	1	1	0	1	0	0	0	1	4
475	0	1	1	0	1	0	0	1	0	4
476	0	1	1	0	1	0	0	0	0	4
477	0	1	1	0	1	0	0	1	1	4
478	0	1	1	0	1	0	0	1	0	4
479	0	1	1	0	1	0	0	0	1	4
480	0	1	1	0	1	0	0	0	0	4
481	0	1	1	0	0	1	1	1	1	4
482	0	1	1	0	0	1	1	0	1	4
483	0	1	1	0	0	1	1	1	0	4
484	0	1	1	0	0	1	1	0	0	5
485	0	1	1	0	0	1	1	1	1	5
486	0	1	1	0	0	1	1	1	0	5
487	0	1	1	0	0	1	1	0	1	5
488	0	1	1	0	0	1	1	0	0	5
489	0	1	1	0	0	1	0	1	1	5
490	0	1	1	0	0	1	0	0	1	5
491	0	1	1	0	0	1	0	1	0	5
492	0	1	1	0	0	1	0	0	0	5
493	0	1	1	0	0	1	0	1	1	5
494	0	1	1	0	0	1	0	1	0	5
495	0	1	1	0	0	1	0	0	1	5
496	0	1	1	0	0	1	0	0	0	5
497	0	1	1	0	0	0	1	1	1	5
498	0	1	1	0	0	0	1	0	1	5
499	0	1	1	0	0	0	1	1	0	5
500	0	1	1	0	0	0	1	0	0	5
501	0	1	1	0	0	0	1	1	1	5
502	0	1	1	0	0	0	1	1	0	5
503	0	1	1	0	0	0	1	0	1	6
504	0	1	1	0	0	0	1	0	0	6
505	0	1	1	0	0	0	0	1	1	6
506	0	1	1	0	0	0	0	0	1	6
507	0	1	1	0	0	0	0	1	0	6
508	0	1	1	0	0	0	0	0	0	6
509	0	1	1	0	0	0	0	1	1	6
510	0	1	1	0	0	0	0	1	0	6
511	0	1	1	0	0	0	0	0	1	6
512	0	1	1	0	0	0	0	0	0	6

Таблиця А.3

Зв’язок шляхів поширення файлового ЗПЗ та команд

l	Шлях l	Port	Int	Ah	Al	Дія	Опис
1	2	3	4	5	6	7	8
1	3,10,11		13h	00h		D	Скидання пристрою
2	3,10,11		13h	01h		D	Запитати статус помилки диску
3	3,8-11		13h	02h		R/D	Читати сектор
4	3,8-11		13h	03h		R/D	Писати в сектор
5	3,8-11		13h	04h		D	Верифікація сектору
6	3,8-11		13h	05h		D	Форматувати трек
7	3,10,11		13h	08h		D	Отримати параметри диску
8	3,10,11		13h	09h		D	Ініціалізація таблиці параметрів диску
9	3,8-11		13h	0ah		R/D	Читати сектор
10	3,8-11		13h	0bh		R/D	Записати сектор
11	3,8-11		13h	0ch		R/D	Перейти на циліндр
12	3,8-11		13h	0dh		D	Альтернативне скидання диску
13	3,8-11		13h	10h		R/D	Тест готовності диску
14	3,8-11		13h	11h		R/D	Рекалібровка диску
15	3,8-11		13h	14h		D	Тест контролеру накопичувача
16	3,8-11		13h	15h		D	Встановити тип диску
17	3,10,11		13h	16h		R/D	Визначити зміни середовища
18	3,10,11		13h	17h		R/D	Встановити частоту передачі
19	3,10,11		13h	18h		R/D	Встановити тип середовища для форматування
20	3,10,11		15h	80h		R/D	Відкрити пристрій
21	3,10,11		15h	81h		R/D	Закрити пристрій
22	0,1		15h	87h		D	Перемістити блок пам’яті
23	3,10,11		21h	0dh		R/D	Вибрати довільний диск
24	3,10,11		21h	0eh		R/D	Відкрити файл
25	3,10,11		21h	10h		R/D	Закрити файл
26	3,10,11		21h	11h		R	Знайти довільний файл
27	3,10,11		21h	12h		R	Знайти наступний файл
28	3,10,11		21h	13h		D	Стерти файл
29	3,10,11		21h	14h		R/D	Читати запис з файлу

Продовження таблиці А.3

1	2	3	4	5	6	7	8
30	3,10,11		21h	15h		R/D	Записати запис у файл
31	3,10,11		21h	16h		R	Створити файл
32	3,10,11		21h	17h		D	Перейменувати файл
33	3,10,11		21h	19h		R/D	Опитати довільний диск
34	3,10,11		21h	1ah		R/D	Установити DTA
35	3,10,11		21h	1bh		R/D	Отримати інформацію про довільний диск
36	3,10,11		21h	21h		R/D	Читати довільний запис з файлу
37	3,10,11		21h	22h		R/D	Записати довільний запис у файл
38	3,10,11		21h	23h		R	Визначити розмір файлу
39	3,10,11		21h	26h		R	Створити PSP
40	3,10,11		21h	27h		R/D	Читати довільний блок файлів
41	3,10,11		21h	28h		R/D	Записати довільний блок файлів
42	3,10,11		21h	29h		D	Змінити ім’я файлу
43	3,10,11		21h	2fh		R	Опитати DTA
44	0,1,4,5		21h	31h		R	Стати резидентом
45	3,10,11		21h	3ch		R/D	Відкрити файл за файловим числом
46	3,10,11		21h	3dh		R/D	Створити файл за файловим числом
47	3,10,11		21h	3eh		R/D	Закрити файл за файловим числом
48	3,10,11		21h	3fh		R/D	Читати з файлу за файловим числом
49	3,10,11		21h	40h		R/D	Записати в файл за файловим числом
50	3,10,11		21h	41h		D	Стерти файл за файловим числом
51	3,10,11		21h	42h		R/D	Встановити точку входу
52	3,10,11		21h	43h	01 h	D	Встановити атрибути
53	3,10,11		21h	45h		D	Створити дублікат файлового числа
54	3,10,11		21h	46h		D	Знищити дублікат файлового числа

Продовження таблиці А.3

1	2	3	4	5	6	7	8
55	3,10,11		21h	47h		R/D	Визначити довільний каталог
56	0,1,4,5		21h	49h		R	Виділити блок пам’яті
57			21h	4eh		R	Знайти перший машинний файл
58	3,10,11		21h	4fh		R	Знайти наступний файл
59	3,10,11		21h	50h		R	Установити PSP
60	3,10,11		21h	51h		R/D	Визначити поточне PSP
61	3,10,11		21h	52h		R/D	Установити зміні оточення
62	3,10,11		21h	56h		D	Перейменувати перемістити файли
63	0,1,4,5		21h	58h	xx h	R	Управління пам’яттю
64			21h	Інші		D	Інші функції 21 переривання
65				Інші		D	Інші переривання
66	0,1,4,5	000-00f				R/D	Управління пам’яттю
67	0,1,4,5	080-083				R/D	Управління пам’яттю
68	3,8-11	320-32f				R/D	Доступ до жорсткого диску
69	3,8-11	070-177				R/D	Доступ до жорсткого диску
70		Інші				D	Інші порти КС

Додаток Б

Програмне забезпечення РБС Distributed Multilevel System

Опис методів класів реалізованих у розробленій розподіленій багаторівневій системі DefenderApp представлено у табл. Б.1 – Б.5.

Таблиця Б.1

Інтерфейс класу DefenderEngine

Назва методу	Призначення та реалізовані функції
1	2
DefenderEngine()	Конструктор класу, який ініціалізує усі поля значеннями за замовчуванням
~DefenderEngine()	Деструктор класу, який виконує очищення виділеної пам’яті та закриття каналу передачі повідомлень через сокети
void init(int port)	Ініціалізація сокетів на передачу даних через вибраний порт
void scheduleTask()	Запуск планування наступної задачі
void stopAnyRunningTask()	Зупинка поточного завдання
void setPort(int port)	Встановити порт для обміну повідомленнями в мережі
int getPort()	Отримати порт для обміну повідомленнями в мережі
void sendStartUpMessage()	Надіслати повідомлення про активацію усім решті програмним модулям
void setTimeout(int ms)	Встановити тайм-аут оновлення списку запущених процесів
int getTimeout()	Отримати тайм-аут оновлення списку запущених процесів
ProcessesList getAllRunningProcesses()	Отримати список запущених процесів
void killProcess(long)	Завершити виконання процесу за вказаним ідентифікатором
void sendCurrentStateMessage (const QString&)	Надіслати повідомлення про поточний стан виконання завдання

Продовження таблиці Б.1

1	2
FilesInfo checkFilesInDir (const QString&)	Перевірити файли у директорії
ProcessesList checkRunningProcesses()	Перевірити запущені процеси
void scanFiles(const FilesInfo &)	Сканувати файли у списку
void scanProcesses (const ProcessesList&)	Сканувати процеси у списку
void optimizeLog()	Оптимізувати файл-журнал, видаливши застарілі записи
int getWorkingPeriod()	Отримати встановлений робочий період
void setWorkingPeriod(int period)	Встановити робочий період
EngineStates getCurrentState()	Отримати поточний стан виконання завдання
void setCurrentState (EngineStates state)	Встановити стан виконання завдання
void sendShutDownMessage()	Надіслати повідомлення про деактивацію модуля
void sendStateCompleteInfo (StatesMessageData*msg)	Надіслати повідомлення про завершення виконання завдання

Таблиця Б.2

Інтерфейс класу Message

Назва методу	Призначення та реалізовані функції
1	2
Message ()	Конструктор класу, який ініціалізує усі поля значеннями за замовчуванням
void append(const QByteArray& data)	Записати дані у повідомлення
void append(const char* data, size_t len)	Перевантажений метод запису даних у повідомлення
void setFrom(StatesMessageData* msg)	Створити повідомлення із даних, які містяться в об’єкті класу StatesMessageData

Продовження таблиці Б.2

1	2
std::unique_ptr<StatesMessageData> getStatesMsg()	Отримати із повідомлення об’єкт StatesMessageData
MessageType type()	Отримати тип повідомлення
const QByteArray& body()	Отримати тіло повідомлення
bool valid()	Перевірити чи повідомлення у валідному стані

Таблиця Б.3

Інтерфейс класу MessageReceiver

Назва методу	Призначення та реалізовані функції
MessageReceiver ()	Конструктор класу, який ініціалізує усі поля значеннями за замовчуванням
void recieveMessage(const Message&, const QString&)	Отримати повідомлення
bool isLocalIp(const QString&);	Перевірити чи передана IP адреса є локальною IP адресою

Таблиця Б.4

Інтерфейс класу MessageSender

Назва методу	Призначення та реалізовані функції
MessageSender ()	Конструктор класу, який ініціалізує усі поля значеннями за замовчуванням
void broadcast(const Message& msg)	Розіслати передане повідомлення усім іншим активним програмним модулям у мережі
void send(const Message& msg, const QString&)	Надіслати повідомлення конкретному програмному модулю, який встановлений в комп’ютерній системі з переданою IP адресою

Таблиця Б.5

Інтерфейс класу MainWindow

Назва методу	Призначення та реалізовані функції
MainWindow(QWidget *parent = 0)	Конструктор класу, який ініціалізує усі поля значеннями за замовчуванням
~MainWindow()	Деструктор класу, який здійснює очищення виділеної пам’яті
void closeEvent(QCloseEvent *event)	Callback на подію закриття головного вікна інтерфейсу
void createEngine()	Створити об’єкт класу DefenderEngine
void createUI()	Створити елементи графічного інтерфейсу
void setupEngine()	Налаштувати об’єкт класу DefenderEngine
void setupUI()	Налаштувати роботи на початкові значення графічних елементів інтерфейсу

Додаток В

Апаратно-програмний пристрій РБС

1. Опис структурної схеми USB-обчислювача

USB-обчислювач є апаратно-програмною частиною компоненти РБС виявлення зловмисного програмного забезпечення (ЗПЗ) та призначений для виконання розрахунків. В складі його структурної схеми, зображеної на рис. В.1 міститься обчислювальний блок, енергонезалежна пам’ять програм та даних (ППД), оперативний запам’ятовуючий пристрій (ОЗП) та контролер інтерфейсу USB, об’єднані в єдину схему відповідними шинами.

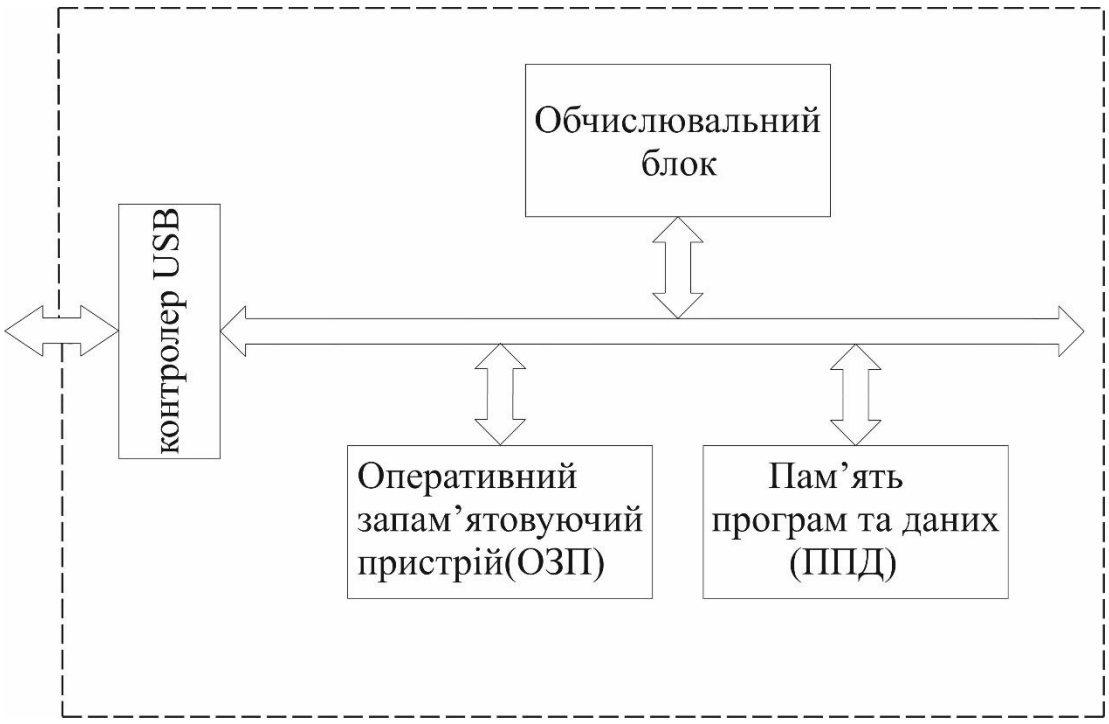


Рис. В.1. Структурна схема USB-обчислювача

Обчислювальний пристрій призначається для безпосереднього виконання розрахунків визначення стану безпеки РБС за командами, які отримані від хоста, в складі якого знаходиться USB-обчислювач, збереження результатів

розрахунків у пам’яті даних з наступною передачею їх за запитом хосту.

Пам’ять програм та даних (ППД) використовується для зберігання програм, які керують роботою USB-обчислювача, збереження результатів розрахунків стану РБС.

Оперативна пам’ять призначена для зберігання поточних локальних даних, які використовуються під час здійснення заданого обчислювального процесу.

Контролер інтерфейсу USB використовується для забезпечення обміну даних між USB-обчислювачем та хостом.

2. Опис принципової схеми USB-обчислювача

Розроблена електрична принципова схема, зображена на рис. В.2 є варіантом реалізації описаної структурної схеми USB-обчислювача. Її особливістю є реалізація у вигляді мікромодуля з габаритними розмірами, що відповідають флеш-накопичувача.

Основою схеми є мікросхема DD1 - мікроконтролер компанії Atmel AT90USB1287-16MV. USB-контролер містить всі необхідні компоненти для з’єднання USB-каналу із вбудованим двох портовим ОЗП (DPRAM).

3. Ініціювання USB-обчислювача зі сторони хоста

USB-обчислювач, як і будь який інший USB - пристрій, підтримує "гаряче" (plug’n’play) з’єднання із динамічним завантаженням та вивантаженням драйверів. Після того, як користувач вставить пристрій в USB-порт, хост знаходить це приєднання, опитує тільки встановлений пристрій та завантажує відповідний драйвер. Даний USB-обчислювач визначається в ОС Windows як пристрій AT90USB1287.

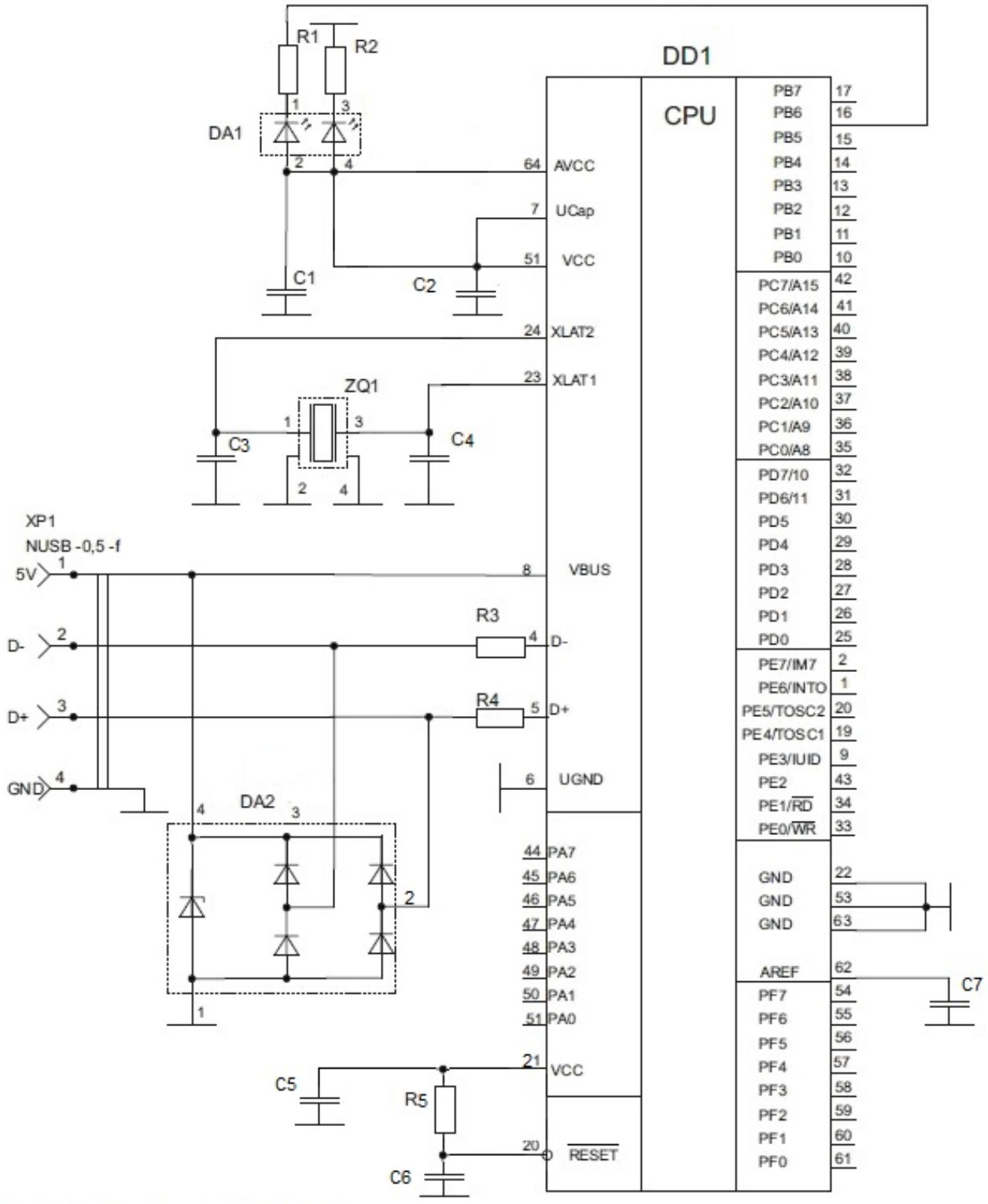


Рис. В.2. Схема електрична принципова USB-обчислювача

4. Ініціювання USB-обчислювача

Ініціалізація USB-обчислювачу виконується шляхом завантаження прикладного програмного забезпечення (firmware) в пам’ять програм мікроконтролера AT90USB1287 при його підготовці до роботи в складі РБС.

Таблиця В.1.

Перелік елементів схеми електричної-принципової USB-обчислювача

Позначення	Назва	Кількість
	Мікросхеми	
DD1	AT90USB1287-16MU	1
DA1	SML-020MLT	1
DA2	PRTR5V0U2X	1
ZQ1	8Mhz, GSX-752	1
	Резистори	
R1,R2	SMD 0201 0,05Вт 220 Ом +-5%	2
R3,R4	SMD 0201 0,05Вт 22 Ом +-5%	2
R5	SMD 0201 0,05Вт 100 кОм +-5%	1
	Конденсатори	
C1,C5-C7	SMD 1206 X7R 50V 0,1мкФ +-10%	3
C2	SMD 1210 X7R 50V 1,0 мкФ +-10%	1
C3,C4	SMD 1206 NPO 50V 18 пФ +-5%	2
	Роз’єми	
XP1	NUSB-0,5-f	1

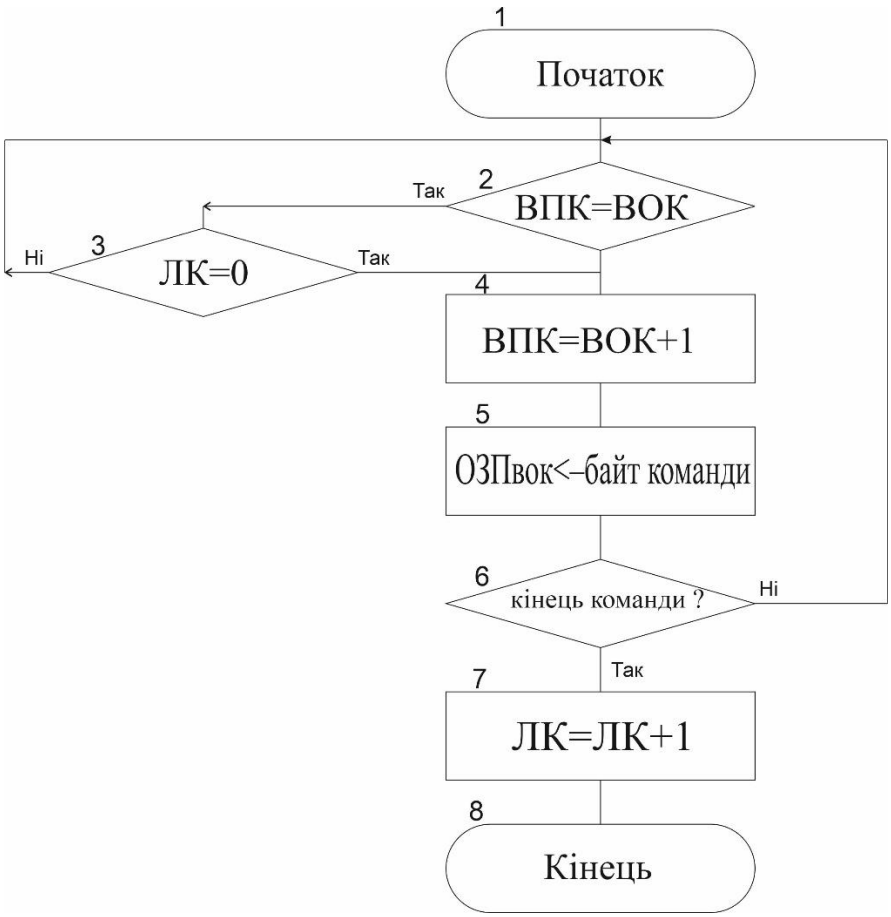
5. Опис алгоритмів роботи програмного забезпечення (ПЗ) USB-обчислювача

Алгоритми роботи ПЗ USB-обчислювача побудовані виходячи з правила, що ініціатором транзакції може бути хост-машина, в складі якої працює USB-обчислювач.

Прикладне ПЗ USB-обчислювача складається з двох основних модулів, які взаємодіють через спільну частину ОЗП, використовуваної як кільцевий

програмний буфер команд (КПБК) розміром 256 байт.

Перший модуль (рис. В.3) реалізується як підпрограма обробки переривань процесора мікроконтролера і відповідає за прийом від хоста, що знаходиться під контролем РБС, команд на виконання розрахунків стану її безпеки, зберігання результатів цих розрахунків та їх видачу за запитом компонентів РБС.



Позначення:
ВОК - вказівник останньої команди;
ВПК – вказівник першої команди;
ЛК – лічильник команд.

Рис. В.3. Алгоритм роботи першого модуля

Даний модуль активується сигналами переривань транзакцій USB-контролера. На нього покладено контроль КПБК USB-обчислювача. Якщо буфер не переповнений командами, то модуль розміщує прийняті з USB-інтерфейсу команди в буфер команд із одночасним модифікуванням вказівника останньої команди (ВОК).

Робота ПЗ USB-обчислювача організована таким чином, що при його підключені до хоста, виконується встановлення апаратних засобів в початковий стан, після чого здійснюється старт процесора, першою командою якого є команда, що є частиною другого модуля. Таким чином, управління USB-обчислювачем одразу ж після старту передається другому модулю.

Другий модуль є прикладною програмою USB-обчислювача (рис. В.4).

Наступним етапом є ініціалізація КПБК з виділенням під нього пам'яті (рис. В.4, блок 3), встановлення початкових значень вказівників першої та останньої команд, видача дозволу переривання процесорного пристрою і, тим самим, дозволу для роботи першого модуля.

Після виконання процедури ініціалізації другий модуль переходить в режим контролю за станом КПБК (рис. В.4).

Якщо КПБК не порожній, то модуль зчитує команду, використовуючи вказівник першої команди (ВПК) із наступною модифікацією ВПК.

На наступному кроці виконується дешифрування вибраної із КПБК команди РБК (блоки 12-14, 18, 19). Система команд РБК для USB-обчислювача включає п'ять базових команд К01 - К05 і може бути розширена.

Команда К01 призначається для очистки енергонезалежної пам'яті USB-обчислювача, відведеної під збереження результатів розрахунків стану безпеки РБС.

Команди К02 та К05 призначаються для виконання розрахунків станів безпеки РБС на 1-му та 2-му етапах відповідно. Команда К02 програмно реалізує алгоритм розрахунку стану РБС за формулою 2.12 розділу 2, а команда К05 - за формулою 2.39 розділу 2.

Команди К03 та К04 призначені для видачі РБС результатів розрахунку станів її безпеки на 1-му та 2-му етапах відповідно.

Після виконання поточної команди модуль повертається до точки контролю стану КПБК.

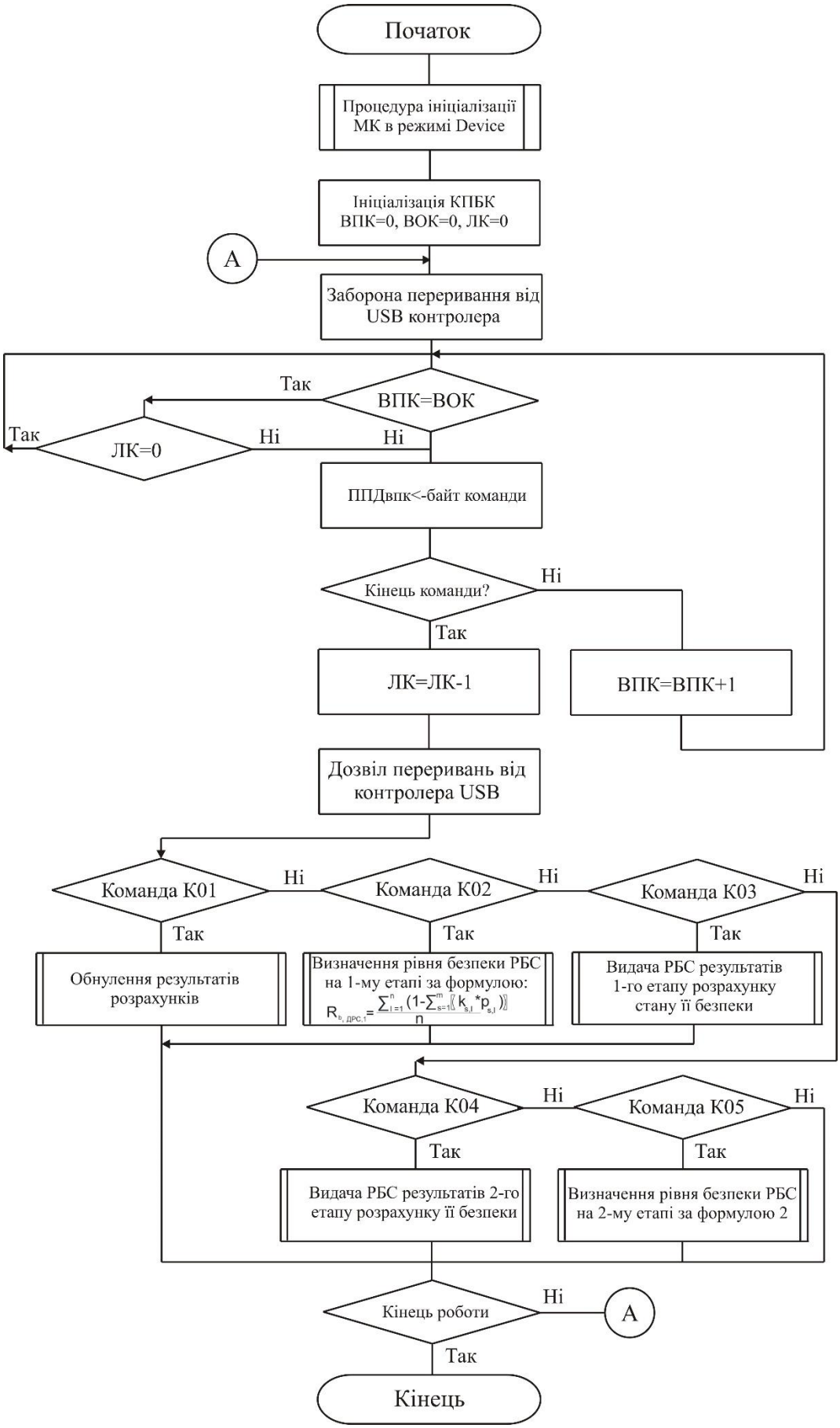


Рис. В.4. Алгоритм роботи другого модуля

Додаток Г

Фрагмент результатів роботи РБС Distributed Multilevel System

Записи з файлу-журналу Defender App, які відображають результати роботи системи за декілька годин.

WorkingState: 04.04.2018 10:17:05.064 Scheduling next task. Working period = 10 minutes.

WorkingState: 04.04.2018 10:17:05.065 Start working on task: Checking running processes

WorkingState: 04.04.2018 10:17:05.067 Complete work on task: Checking running processes, results: running processes = 5

WorkingState: 04.04.2018 10:27:05.104 Scheduling next task. Working period = 10 minutes.

WorkingState: 04.04.2018 10:27:05.104 Start working on task: Checking files in hard disk in C:/Users/8.1x64/AppData/Local

WorkingState: 04.04.2018 10:27:05.337 Complete work on task: Checking files in hard disk in C:/Users/8.1x64/AppData/Local, results: total dirs count = 802, total files count = 835, total exe files = 16

WorkingState: 04.04.2018 10:27:05.337 Start working on task: Scanning files in hard disk

WorkingState: 04.04.2018 10:27:05.337 Complete work on task: Scanning files in hard disk, results: scanning files = 16, infected files = 0

WorkingState: 04.04.2018 10:36:05.572 169.254.169.70 started working on: Optimizing stored data

WorkingState: 04.04.2018 10:36:05.572 169.254.169.70 completed working on task: Optimizing stored data, results: deleted entries = 11, file size before = 8418, file size after = 7335

WorkingState: 04.04.2018 10:36:05.588 169.254.169.70 started working on: Waiting next task

WorkingState: 04.04.2018 13:48:12.178 Scheduling next task. Working period = 116 minutes.

WorkingState: 04.04.2018 13:48:12.178 Start working on task: Checking running processes

WorkingState: 04.04.2018 13:48:12.193 Complete work on task: Checking running processes, results: running processes = 4

WorkingState: 04.04.2018 13:48:12.193 Start working on task: Scanning running processes

WorkingState: 04.04.2018 13:48:12.209 Complete work on task: Scanning running processes, results: scanned processes = 4, infected processes = 0

WorkingState: 04.04.2018 13:48:43.444 10.0.2.15 started working on: Checking files in hard disk

WorkingState: 04.04.2018 13:48:43.491 10.0.2.15 completed working on task: Checking files in hard disk in C:/Users/8.1x64, results: total dirs count = 31, total files count = 21, total exe files = 5

WorkingState: 04.04.2018 13:48:43.491 10.0.2.15 started working on: Waiting next task

WorkingState: 04.04.2018 13:49:36.445 Scheduling next task. Working period = 116 minutes.

WorkingState: 04.04.2018 13:49:36.451 Start working on task: Checking files in hard disk in C:/Users/8.1x64

WorkingState: 04.04.2018 13:49:36.488 Complete work on task: Checking files in hard disk in C:/Users/8.1x64, results: total dirs count = 31, total files count = 21, total exe files = 5

WorkingState: 04.04.2018 13:58:05.527 169.254.169.70 started working on: Checking files in hard disk

WorkingState: 04.04.2018 13:58:05.527 169.254.169.70 completed working on task: Checking files in hard disk in C:/Users/8x64_/Music, results: total dirs count = 2, total files count = 0, total exe files = 0

WorkingState: 04.04.2018 13:58:05.541 169.254.169.70 started working on: Waiting next task

WorkingState: 04.04.2018 13:59:05.511 169.254.169.70 started working on: Checking files in hard disk

WorkingState: 04.04.2018 13:59:05.525 169.254.169.70 completed working on task: Checking files in hard disk in C:/Users/8x64_/Music, results: total dirs count = 2, total files count = 0, total exe files = 0

WorkingState: 04.04.2018 13:59:05.541 169.254.169.70 started working on: Waiting next task

WorkingState: 04.04.2018 14:00:05.025 Scheduling next task. Working period = 1 minutes.

WorkingState: 04.04.2018 14:00:05.025 Start working on task: Checking files in hard disk in C:/Users/8.1x64/AppData/Local

WorkingState: 04.04.2018 14:00:05.526 10.0.2.14 started working on: Checking running processes

WorkingState: 04.04.2018 14:00:05.526 10.0.2.14 completed working on task: Checking running processes, results: running processes = 40

WorkingState: 04.04.2018 14:00:05.526 10.0.2.14 started working on: Waiting next task

WorkingState: 04.04.2018 14:00:05.760 Complete work on task: Checking files in hard disk in C:/Users/8.1x64/AppData/Local, results: total dirs count = 802, total files count = 835, total exe files = 16

WorkingState: 04.04.2018 14:10:05.041 Scheduling next task. Working period = 10 minutes.

WorkingState: 04.04.2018 14:10:05.041 Start working on task: Checking running processes

WorkingState: 04.04.2018 14:10:05.041 Complete work on task: Checking running processes, results: running processes = 21

WorkingState: 04.04.2018 14:10:05.041 Start working on task: Scanning running processes

WorkingState: 04.04.2018 14:10:05.041 Complete work on task: Scanning running processes, results: scanned processes = 21, infected processes = 0

WorkingState: 04.04.2018 14:01:05.541 169.254.169.70 started working on: Checking running processes

WorkingState: 04.04.2018 14:01:05.557 169.254.169.70 completed working on task: Checking running processes, results: running processes = 4

WorkingState: 04.04.2018 14:01:05.557 169.254.169.70 started working on: Scanning running processes

WorkingState: 04.04.2018 14:01:05.557 169.254.169.70 completed working on task: Scanning files in hard disk, results: scanned processes = 4, infected processes = 0

WorkingState: 04.04.2018 14:01:05.572 169.254.169.70 started working on: Waiting next task

WorkingState: 04.04.2018 14:02:05.042 Scheduling next task. Working period = 1 minutes.

WorkingState: 04.04.2018 14:02:05.042 Start working on task: Checking running processes

WorkingState: 04.04.2018 14:02:05.042 Complete work on task: Checking running processes, results: running processes = 5

WorkingState: 04.04.2018 14:02:05.526 169.254.169.70 started working on: Checking running processes

WorkingState: 04.04.2018 14:02:05.542 169.254.169.70 completed working on task: Checking running processes, results: running processes = 4

WorkingState: 04.04.2018 14:02:05.557 169.254.169.70 started working on: Waiting next task

WorkingState: 04.04.2018 14:03:05.057 Scheduling next task. Working period = 1 minutes.

WorkingState: 04.04.2018 14:03:05.057 Start working on task: Optimizing stored data

WorkingState: 04.04.2018 14:03:05.072 Complete work on task: Optimizing stored data, results: deleted entries = 108, file size before = 13176, file size after = 4861

WorkingState: 04.04.2018 14:03:05.541 169.254.169.70 started working on: Optimizing stored data

WorkingState: 04.04.2018 14:03:05.588 169.254.169.70 completed working on task: Optimizing stored data, results: deleted entries = 43, file size before = 7297, file size after = 3742

WorkingState: 04.04.2018 14:03:05.588 169.254.169.70 started working on: Waiting next task

WorkingState: 04.04.2018 14:04:05.056 Scheduling next task. Working period = 1 minutes.

WorkingState: 04.04.2018 14:04:05.056 Start working on task: Optimizing stored data

WorkingState: 04.04.2018 14:04:05.056 Complete work on task: Optimizing stored data, results: deleted entries = 3, file size before = 5861, file size after = 5575

WorkingState: 04.04.2018 14:04:05.541 169.254.169.70 started working on: Optimizing stored data

WorkingState: 04.04.2018 14:04:05.588 169.254.169.70 completed working on task: Optimizing stored data, results: deleted entries = 3, file size before = 4724, file size after = 4438

WorkingState: 04.04.2018 14:04:05.588 169.254.169.70 started working on: Waiting next task

WorkingState: 04.04.2018 14:05:05.073 Scheduling next task. Working period = 1 minutes.

WorkingState: 04.04.2018 14:05:05.073 Start working on task: Checking running processes

WorkingState: 04.04.2018 14:05:05.073 Complete work on task: Checking running processes, results: running processes = 5

WorkingState: 04.04.2018 14:05:05.088 Start working on task: Scanning running processes

WorkingState: 04.04.2018 14:05:05.103 Complete work on task: Scanning running processes, results: scanned processes = 5, infected processes = 0

WorkingState: 04.04.2018 14:05:05.573 169.254.169.70 started working on: Checking files in hard disk

WorkingState: 04.04.2018 14:05:05.588 169.254.169.70 completed working on task: Checking files in hard disk in C:/Users/8x64_/AppData/Local/DefenderApp, results: total dirs count = 0, total files count = 0, total exe files = 0

WorkingState: 04.04.2018 14:05:05.588 169.254.169.70 started working on: Waiting next task

WorkingState: 04.04.2018 14:08:05.573 169.254.169.70 started working on: Optimizing stored data

WorkingState: 04.04.2018 14:08:05.619 169.254.169.70 completed working on task: Optimizing stored data, results: deleted entries = 3, file size before = 8611, file size after = 8325

WorkingState: 04.04.2018 14:08:05.619 169.254.169.70 started working on: Waiting next task

WorkingState: 04.04.2018 14:09:05.572 169.254.169.70 started working on: Optimizing stored data

WorkingState: 04.04.2018 14:09:05.604 169.254.169.70 completed working on task: Optimizing stored data, results: deleted entries = 3, file size before = 8949, file size after = 8663

WorkingState: 04.04.2018 14:09:05.618 169.254.169.70 started working on: Waiting next task

WorkingState: 04.04.2018 14:10:05.588 169.254.169.70 started working on: Checking files in hard disk

WorkingState: 04.04.2018 14:10:05.588 169.254.169.70 completed working on task: Checking files in hard disk in C:/Users/8x64_/Desktop, results: total dirs count = 2, total files count = 0, total exe files = 0

WorkingState: 04.04.2018 14:10:05.603 169.254.169.70 started working on: Waiting next task

WorkingState: 04.04.2018 14:11:05.118 Scheduling next task. Working period = 3 minutes.

WorkingState: 04.04.2018 14:11:05.118 Start working on task: Checking files in hard disk in C:/Users/8.1x64/AppData/Local/DefenderApp

WorkingState: 04.04.2018 14:11:05.118 Complete work on task: Checking files in hard disk in C:/Users/8.1x64/AppData/Local/DefenderApp, results: total dirs count = 0, total files count = 0, total exe files = 0

WorkingState: 04.04.2018 14:11:05.603 169.254.169.70 started working on: Checking running processes

WorkingState: 04.04.2018 14:11:05.618 169.254.169.70 completed working on task: Checking running processes, results: running processes = 4

WorkingState: 04.04.2018 14:11:05.618 169.254.169.70 started working on: Waiting next task

WorkingState: 04.04.2018 14:12:03.358 Scheduling next task. Working period = 3 minutes.

WorkingState: 04.04.2018 14:12:03.360 Start working on task: Checking running processes

WorkingState: 04.04.2018 14:12:03.364 Complete work on task: Checking running processes, results: running processes = 4

WorkingState: 04.04.2018 14:12:15.103 Scheduling next task. Working period = 3 minutes.

WorkingState: 04.04.2018 14:12:15.103 Start working on task: Optimizing stored data

Message: 04.04.2018 14:12:15.134 Send message about completing working on: Optimizing stored data

WorkingState: 04.04.2018 14:12:15.134 Complete work on task: Optimizing stored data, results: deleted entries = 37, file size before = 15350, file size after = 11985

Message: 04.04.2018 14:12:15.134 Send message about starting woking on: Waiting next task

Message: 04.04.2018 14:12:17.197 Receive startup message from 169.254.169.70

Message: 04.04.2018 14:12:17.197 Send greeting to 169.254.169.70

Message: 04.04.2018 14:12:17.212 Receive poll out message from 169.254.169.70

Message: 04.04.2018 14:12:17.212 Send current state to 169.254.169.70

WorkingState: 04.04.2018 14:12:19.212 169.254.169.70 started working on: Optimizing stored data

WorkingState: 04.04.2018 14:12:19.228 169.254.169.70 completed working on task: Optimizing stored data, results: deleted entries = 15, file size before = 11893, file size after = 10763

WorkingState: 04.04.2018 14:12:19.228 169.254.169.70 started working on: Waiting next task

Message: 04.04.2018 14:12:51.962 Send shutdown message

Message: 04.04.2018 14:12:51.962 Engine is deinitialized

Додаток Д

СПИСОК ПУБЛІКАЦІЙ

Наукові праці, в яких опубліковані основні наукові результати дисертації

1. Савенко О. С. Метод антивірусного діагностування персональних комп’ютерів на основі матриць інцидентності / О. С. Савенко, В. М. Джулій, В. М. Стецюк // Вісник Технологічного університету Поділля. Технічні науки. – 2000. – № 3. – С. 136–139.

2. Савенко О. С. Методика генерації матриць інцидентності для використання в антивірусних програмних засобах / О. С. Савенко // Вісник Технологічного університету Поділля. Технічні науки. – 2003. – № 3, т. 2. – С. 48–56.

3. Савенко О. С. Генерація моделей комп’ютерних вірусних програм в системі оцінки достовірності результатів роботи антивірусних засобів / О. С. Савенко, С. В. Мостовий // Вісник Хмельницького національного університету. Технічні науки. – 2005. – № 4, т. 1. – С. 198–200.

4. Савенко О. С. Дослідження методів антивірусного діагностування комп’ютерних мереж / О. С. Савенко, С. М. Лисенко // Вісник Хмельницького національного університету. Технічні науки. – 2007. – № 2, т. 2. – С. 120–126.

5. Савенко О. С. Дослідження та аналіз блокування процесів в комп'ютерній системі / О. С. Савенко, Ю. П. Кльоц, С. В. Мостовий // Вісник Хмельницького національного університету. Технічні науки. – 2007. – № 3, т. 1. – С. 248–251.

6. Савенко О. С. Життєвий цикл процесів комп'ютерної системи / О. С. Савенко, С. В. Мостовий // Радіоелектронні і комп'ютерні системи. – 2010. –№ 7 (48). – С. 35–38.

7. Локазюк В. М. Модель прогнозування стану взаємоблокування процесів комп’ютерної системи / В. М. Локазюк, О. С. Савенко, С. В. Мостовий // Вісник Вінницького політехнічного інституту. – 2011. – № 4. – С. 130–133.

8. Савенко О. С. Діагностування комп’ютерних систем на наявність

шкідливого програмного забезпечення на основі антивірусної мультиагентної системи / О. С. Савенко, С. М. Лисенко, А. Ф. Крищук // Вісник Національного університету «Львівська політехніка». – 2011. – № 5. – С. 99–105.

9. Савенко О. С. Адаптивна інформаційна технологія виявлення троянських програм в комп'ютерних системах / О. С. Савенко, С. М. Лисенко // Комп'ютинг. – 2011. – Т. 10, № 3. – С. 85–92.

10. Берников А. Р. Поиск вредоносных программ в распределенных тренажерах с использованием технологии нечеткой логики / А. Р. Берников, Р. П. Графов, С. Н. Лысенко, О. С. Савенко // Информационные технологии. *(РИНЦ)* – 2011. – № 10. – С. 42–47.

11. Савенко О. С. Прогнозування потрапляння процесів у стан взаємоблокування / О. С. Савенко, С. В. Мостовий // Вісник Вінницького політехнічного інституту. – 2013. – № 2. – С. 81–86.

12. Нічепорук А. О. Моделі життєвого циклу поліморфних вірусів / А. О. Нічепорук, О. С. Савенко // Комп'ютерно-інтегровані технології: освіта, наука, виробництво. – 2013. – № 11. – С. 64–71.

13. Савенко О. С. Моделі рівнів поліморфних комп'ютерних вірусів / О. С. Савенко, С. М. Лисенко, А. О. Нічепорук // Вісник Вінницького політехнічного інституту *(Index Copernicus)*. – 2015. – № 2. – С. 75–83.

14. Савенко О. С. Програмне забезпечення інформаційної технології моделювання поширення вірусних кодів в гетерогенних мережах / О. С. Савенко // Вісник Хмельницького національного університету. Технічні науки *(Index Copernicus)*. – 2017. – № 1. – С. 144–148.

15. Савенко О. С. Распределенная многоуровневая сетевая система обнаружения метаморфных вирусов в локальных компьютерных сетях / О. С. Савенко // Вестник Брестского государственного технического университета (физика, математика, информатика). – 2017. – № 5 (107). – С. 40–44.

16. Савенко О. С. Критерії класифікації методів виявлення шкідливого програмного забезпечення / О. С. Савенко // Вісник Хмельницького

національного університету. Технічні науки. – 2018. – № 1. – С. 23–27.

17. Савенко О. С. Модель та архітектура розподіленої багаторівневої системи виявлення шкідливого програмного забезпечення в локальних комп'ютерних мережах / О. С. Савенко // Вісник Хмельницького національного університету. Технічні науки. – 2018. – № 2. – С. 153–163.

18. Савенко О. С. Формалізація шкідливого програмного забезпечення в локальних комп'ютерних мережах / О. С. Савенко // Вісник Хмельницького національного університету. Технічні науки. – 2018. – № 3. – С. 145–154.

19. Савенко О. С. Архітектура багаторівневої програмної системи виявлення шкідливого програмного забезпечення в локальних комп'ютерних мережах / О. С. Савенко, В. І. Грибинчук, М. О. Кульчицький // Комп'ютерно-інтегровані технології: освіта, наука, виробництво. – 2018. – № 30–31. – С. 132–140.

20. Савенко О. С. Архітектура розподіленої багаторівневої системи виявлення шкідливого програмного забезпечення в локальних комп'ютерних мережах / О. С. Савенко // Вчені записки Таврійського національного університету. Технічні науки. – 2018. – Т. 29 (68), № 2. – С. 172–181.

21. Савенко О. С. Формування сигнатури поведінки програми на основі трасування API викликів / О. С. Савенко, А. О. Нічепорук, А. А. Нічепорук, Ю. О. Нічепорук // Електротехнічні та комп'ютерні системи. – 2018. – № 29 (105). – С. 67–77.

Праці, які засвідчують апробацію матеріалів дисертації

22. Savenko O. Multi-agent based approach of botnet detection in computer systems / O. Savenko, S. Lysenko, A. Kryschuk // Communications in Computer and Information Science, ISSN: 1865-0929 *(Scopus, Web of Science*). – 2012. – Vol. 291. – Pp. 171–180.

23. Pomorova O. Multi-Agent Based Approach for Botnet Detection in a Corporate Area Network Using Fuzzy Logic / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk // Communications in Computer and Information Science, ISSN: 1865–

0929 *(Scopus, Web of Science)*. – 2013. – Vol. 370. – Pp. 146–156.

24. Pomorova O. A Technique for detection of bots which are using polymorphic code / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, A. Nicheporuk // Communications in Computer and Information Science, ISSN: 1865–0929 *(Scopus, Web of Science)*. – 2014. – Vol. 431. – Pp. 265–276.

25. Savenko O. Metamorphic Viruses' Detection Technique Based on the Equivalent Functional Block Search / O. Savenko, S. Lysenko, A. Nicheporuk, B. Savenko // CEUR-WS, ISSN: 1613–0073 *(Scopus)*. – 2017. – Vol. 1844. – Pp. 555–569.

26. Lysenko S. Information technology for botnets detection based on their behaviour in the corporate area network / S. Lysenko, O. Savenko, K. Bobrovnikova, A. Kryshchuk, B. Savenko // Communications in Computer and Information Science, ISSN: 1865–0929 *(Scopus, Web of Science)*. – 2017. – Vol. 718. – Pp. 166–181.

27. Markowsky G. The technique for metamorphic viruses' detection based on its obfuscation features analysis / G. Markowsky, O. Savenko, S. Lysenko, A. Nicheporuk // CEUR-WS, ISSN: 1613–0073 *(Scopus)*. – 2018. – Vol. 2104. – Pp. 680–687.

28. Markowsky G. Distributed Malware Detection System Based on Decentralized Architecture in Local Area Networks / G. Markowsky, O. Savenko, A. Sachenko // Advances in Intelligent Systems and Computing, ISSN: 2194–5357 *(Scopus)*. – 2019. – Vol. 871. – Pp. 582–598.

29. Savenko O. The Technique for Computer Systems Trojan Diagnosis in the Monitor Mode / O. Savenko, S. Lysenko // Proceedings of the 6-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Prague (Czech Republic), September 15–17, 2011 *(Scopus)*. – Prague, 2011. – Pp. 770–774.

30. Savenko O. Botnet detection technique for corporate area network / O. Savenko, S. Lysenko, A. Kryshchuk, Y. Klots / Proceedings of the 7-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Berlin (Germany), September 12–14, 2013

(Scopus). – Berlin, 2013. – Pp. 363–368.

31. Lysenko S. DNS-based Anti-evasion Technique for Botnets Detection / S. Lysenko, O. Pomorova, O. Savenko, A. Kryshchuk and K. Bobrovnikova // Proceedings of the 8-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Warsaw (Poland), September 24–26, 2015 *(Scopus, Web of Science)*. – Warsaw, 2015. – Pp. 453–458.

32. Pomorova O. A Technique for the Botnet Detection Based on DNS-Traffic Analysis / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, K. Bobrovnikova // Proceedings of the 22-nd International Conference Computer Networks, Brunów (Poland), June 16–19, 2015 *(Scopus, Web of Science)*. – Brunów, 2015. – Vol. 522. – Pp. 127–138.

33. Pomorova O. Anti-evasion Technique for the Botnets Detection Based on the Passive DNS Monitoring and Active DNS Probing / O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, K. Bobrovnikova // Proceedings of the 23-nd International Conference Computer Networks, Brunów (Poland), June 14–17, 2016 *(Scopus, Web of Science)*. – Brunów, 2016 – Vol. 608. – Pp. 83–95.

34. Savenko O. Approach for the Unknown Metamorphic Virus Detection / O. Savenko, S. Lysenko, A. Nicheporuk, B. Savenko // Proceedings of the 8-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Bucharest (Romania), September 21–23, 2017 *(Scopus, Web of Science)*. – Bucharest, 2017. – Pp. 71–76.

35. Markowsky G. Distributed System for Detecting the Malware in LAN / G. Markowsky, O. Savenko, A. Sachenko // Proceedings of the 2018 IEEE 13th International Scientific and Technical Conference on Computer Science and Information Technologies, CSIT’2018, Lviv (Ukraine), September 11–14, 2018 *(Scopus, Web of Science)*. – Lviv, 2018. – Pp. 306–309.

36. Графов Р. П. Забезпечення надійності розподільних мереж на основі динамічної структурної реконфігурації / Р. П. Графов, О. С. Савенко // Інформаційні технології та комп’ютерна інженерія. – 2005. – №3. – С. 241–246.

37. Савенко О. С. Метод виявлення бот-мереж розподіленими системами

на основі самоорганізації / О. С. Савенко // Штучний інтелект. – 2018. – № 4 (82). – С. 58–72.

38. Savenko O. Intelligent method of the search of the trojan program in computer systems / O. Savenko, S. Lysenko // Праці IV міжнародної конференції «Сучасні комп'ютерні системи та мережі: розробка та використання», (ACSN'2009), (Львів, 17–19 грудня 2009). – Львів, 2009. – С. 154–158.

39. Savenko O. Software for computer systems trojans diagnosing as a safety-case tool / O. Savenko, S. Lysenko // Proceedings of the 1-st International Workshop on Critical infrastructure safety and security, CrlSS-DESSERT'11, Kirovograd, May 11–13, 2011. – Kirovograd, 2011. – Pp. 353–361.

40. Савенко О. С. Дослідження антивірусних технологій діагностування комп’ютерних систем на наявність шкідливого програмного забезпечення / О. С. Савенко, С. М. Лисенко, А. Ф. Крищук // Праці XII міжнародної науково-практичної конференції «Сучасні інформаційні та електронні технології», (СІЕТ-2011), (Одеса, 27–31 травня 2011). – Одеса, 2011. – С. 165–166.

41. Savenko O. Interoperability of distributed multiple system for malware detection based on components levels of safety / O. Savenko // Проблеми інформаційних технологій. – 2018. – № 24. – С. 78–92.

42. Савенко О. С. Функційна модель бота як складової ботнет-мережі / О. С. Савенко, С. М. Лисенко, А. Ф. Крищук // Тези доповідей І Міжнародної науково-технічної конференції «Захист інформації і безпека інформаційних систем», (Львів, 2012). – Львів, 2012. – С. 123–124.

43. Савенко О. С. Розподілена апаратно-програмна система та методи захисту інформації в комп’ютерних системах локальних мереж / О. С. Савенко // Наукові праці Чорноморського національного університету ім. П. Могили. Комп’ютерні технології. – 2018. – Т. 320. Вип. 308. – С. 72–75.

44. Савенко О. С. Метод взаємодії компонентів розподіленої системи виявлення зловмисного програмного забезпечення в локальних обчислювальних мережах / О. С. Савенко, А. О. Нічепорук // Збірник тез доповідей XIV Міжнародної конференції «Контроль і управління в складних системах», (КУСС-

2018), (Вінниця, 15–17 жовтня, 2018). – Вінниця, 2018. – С. 37.

45. Савенко О. С. Формалізоване структурування шкідливого програмного забезпечення на основі алгебраїчних систем / О. С. Савенко // Вимірювальна та обчислювальна техніка в технологічних процесах. – 2018. – №1. – С. 67–72.

46. Савенко О. С. Розподілена система виявлення зловмисного програмного забезпечення та метод взаємодії її компонент в локальних мережах / О. С. Савенко // Матеріали доповідей V Міжнародної науково-практичної конференції «Інформаційні технології та взаємодії», (Київ, КНУ ім. Т. Шевченка, 20–21 листопада 2018). – Київ, 2018. – С. 308–309.

Публікації, які додатково відображають наукові результати дисертації

47. Пат. на корисну модель 108238 Україна, МПК G06F 21/55 Мультиагентний спосіб локалізації бот-мереж у корпоративних комп'ютерних мережах / О. В. Поморова, О. С. Савенко, А. Ф. Крищук, С. М. Лисенко, К. Ю. Бобровнікова, А. О. Нічепорук; заявник і патентовласник Хмельницький національний університет. – № u201600127; заявл. 04.01.2016; опубл. 11.07.2016, Бюл. № 13/2016.

48. Пат. на корисну модель 118456 Україна, МПК G06F 21/55 Спосіб виявлення метаморфних вірусів на основі статистичних метрик для визначення еквівалентних функціональних програмних блоків / О. С. Савенко, С. М. Лисенко, К. Ю. Бобровнікова, А. О. Нічепорук, Б. О. Савенко; заявник і патентовласник Хмельницький національний університет. – № u201701743; заявл. 23.02.2017; опубл. 10.08.2017, Бюл. № 15/2017.

49. Пат. на корисну модель 118663 Україна, МПК G06F 21/55 Спосіб ідентифікації бот-мереж у корпоративних комп'ютерних мережах на основі аналізу DNS-трафіку / О. С. Савенко, С. М. Лисенко, К. Ю. Бобровнікова, А. О. Нічепорук, Б. О. Савенко; заявник і патентовласник Хмельницький національний університет. – № u201612041; заявл. 28.11.2016; опубл. 28.08.2017, Бюл. № 16/2017.

50. А. с. 80223 Україна. Комп’ютерна програма пошуку та визначення еквівалентних функціональних блоків у виконуваних файлах для ідентифікації ознак метаморфних вірусів в локальних комп’ютерних мережах / А. О. Нічепорук, О. С. Савенко, С. М. Лисенко. 2018.

51. А. с. 80445 Україна. Розподілена комп’ютерна програма для виявлення зловмисного програмного забезпечення в локальних обчислювальних мережах на основі аналізу поведінкових сигнатур / О. С. Савенко. 2018.

Додаток Е

Акти впровадження результатів дисертаційної роботи

«Затверджую»
Головний інженер ДП «Новатор»
Тітов Ю.О.
«28» 11 2019 р.

АКТ
про впровадження результатів дисертаційної роботи
Савенка Олега Станіславовича
«Теорія і практика створення розподілених систем виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах»

Результати дисертаційної роботи професора кафедри комп'ютерної інженерії та системного програмування Хмельницького національного університету Савенка О.С. пройшли апробацію на Державному підприємстві «Новатор» у відділі автоматизованих систем управління.

В процесі проходження апробації розподілених систем виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах були використані на ДП «Новатор» такі результати, які одержані Савенком О.С. особисто:

1) вперше розроблено модель архітектури розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах, особливістю якої є синтез в ній вимог розподіленості, децентралізованості, багаторівневості та самоорганізованості, що на відміну від відомих систем дає змогу її використовувати автономно для виконання закладених в ній функцій;

2) вперше розроблено мережний метод виявлення файлового ЗПЗ, який для організації функціонування програмних модулів РБС, здійснення вилучення ймовірно уражених ПМ з РБС, встановлення відношення до файлового ЗПЗ на основі обміну і обробки знань, сканування виконуваних файлів створенням для них окремих процесів, базується на двох розроблених методах, які здійснюють побудову поведінкових сигнатур та їх подальший аналіз на наявність файлового ЗПЗ, в тому числі поліморфного та метаморфного коду і для здійснення детальнішого аналізу програмного коду до процесу виявлення залучаються інші ПМ РБС;

3) вперше розроблено метод виявлення файлового зловмисного програмного забезпечення, який базується на основі динамічного формування поведінкової сигнатури шляхом відстеження викликів API і дає змогу виявляти інші різновиди вірусних програм, зокрема нові версії існуючих вірусів;

4) вперше розроблено метод виявлення поліморфних та метаморфних вірусів на основі аналізу функцій обфускації, особливістю якого є аналіз програмного об'єкту та його модифікованих версій, отриманих від різних ПМ РБС і подальший аналіз на основі пошуку еквівалентних функціональних блоків, що дозволяє здійснити детальніший аналіз коду програмного об'єкту на наявність поліморфних та метаморфних вірусів.

Отриманні результати дозволили підвищити достовірність виявлення файлового зловмисного програмного забезпечення до 10 % та зменшити кількість помилок першого роду до 5,2 % за рахунок організації узгодженої мережної взаємодії компонентів розробленої системи. Результати роботи дозволяють суттєво підвищувати зовнішню завадостійкість системи автоматизованого управління виробництва підприємства, що є досить актуальною проблемою на наш час.

Цей акт не є підставою для фінансових розрахунків.

Начальник відділу
автоматизованих систем управління Притисюк С.А.

«Затверджую»
Директор ТОВ «ITT - telecommunication company»
_____ В.С. Сімогук
«21» 02 2019 р.

АКТ
про впровадження результатів дисертаційної роботи
Савенка Олега Станіславовича
«Теорія і практика створення розподілених систем виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах»

Комісія в складі:

Технічний директор — Ладунець О.Г.

Начальник відділу продажу комп'ютерної техніки — Вінтер Ю.Г.

Адміністратор — Веремеєнко В.А.

склала акт про впровадження результатів дисертаційної роботи професора кафедри комп'ютерної інженерії та системного програмування Хмельницького національного університету Савенка О.С. на «ITT - telecommunication company», в тому, що він проводив роботу по впровадженню розподілених системи виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах.

В процесі вирішення науково-технічної проблеми розвитку теорії і практики створення розподілених систем в комп'ютерних системах локальних мереж для підвищення достовірності процесу виявлення зловмисного програмного забезпечення, яка ґрунтується на використанні основних положень теорії розподілених систем, теорії множин, теорії графів, методів класифікації та теорії комп'ютерних мереж. Савенком О.С. було особисто отримано і використано на «ITT - telecommunication company» такі результати:

1) вперше розроблено модель архітектури розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах, особливістю якої є синтез в ній вимог розподіленості, децентралізованості, багаторівневості та самоорганізованості, що на відміну від відомих систем дає змогу її використовувати автономно для виконання закладених в ній функцій;

2) вперше розроблено метод взаємодії компонентів розподіленої багаторівневої системи, який дозволяє організовувати підтримку цілісності системи та здійснення передачі знань, отриманих окремими структурними компонентами системи програмними модулями іншим компонентам і базується на основі аналітичних залежностей рівнів безпеки програмних

модулів і РБС, що дозволяє здійснювати зміну архітектури РБС динамічно та впливати на подальші її дії без втручання користувача, який на відміну від інших методів організації зв'язуючої компоненти програмного забезпечення створює можливості гнучкої зміни архітектури самої системи додаючи і видаляючи окремі компоненти системи; на основі розробленої моделі архітектури розподіленої багаторівневої системи та методу взаємодії компонентів було реалізовано програмне забезпечення, яке використовувалось в локальній мережі ТОВ «ITT - telecommunication company»;

3) вперше розроблено метод виявлення бот-мереж на основі класифікації в комп'ютерних системах локальних мереж на основі пасивного моніторингу системних подій та здійснення узгодження взаємодії програмних модулів РБС при прийнятті рішення для спільного досягнення поставленої мети.

Проведені експериментальні дослідження за участі Савенка О.С. надали можливість перевірити ефективність програмного забезпечення, розробленого на базі методів виявлення зловмисного програмного забезпечення;

Експериментальні дослідження показали, що розроблені методи виявлення зловмисного програмного забезпечення в локальних мережах дозволяють виявляти його існуючі та нові копії з вищим рівнем достовірності, ніж існуючі мережні антивірусні засоби.

Отримані результати дозволили підвищити достовірність виявлення зловмисного програмного забезпечення на 5-12% в порівнянні з існуючими мережними антивірусними програмними засобами.

Цей акт не є підставою для фінансових розрахунків.

Ладунець О.Г.
Вінтер Ю.Г.
Веремеєнко В.А.

«Затверджую»
Директор ТОВ «ЮКС++»
_____Ю.П. Кльоц
«_26_» ____02_____ 2019 р.

Акт
про впровадження результатів дисертаційної роботи
Савенка Олега Станіславовича
«Теорія і практика створення розподілених систем виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах»

Професор кафедри комп'ютерної інженерії та системного програмування Хмельницького національного університету Савенко О.С. дійсно впроваджував результати дисертаційної роботи в ТОВ «ЮКС++», зокрема, він проводив роботу по впровадженню розподілених системи виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах.

В процесі вирішення науково-технічної проблеми розвитку теорії і практики створення розподілених систем в комп'ютерних системах локальних мереж для підвищення достовірності процесу виявлення зловмисного програмного забезпечення, яка ґрунтується на використанні основних положень теорії розподілених систем, теорії множин, теорії графів, методів класифікації та теорії комп'ютерних мереж. Савенком О.С. було особисто отримано і використано в ТОВ «ЮКС++» такі результати:

1) вперше розроблено модель архітектури розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах, особливістю якої є синтез в ній вимог розподіленості, децентралізованості, багаторівневості та самоорганізованості, що на відміну від відомих систем дає змогу її використовувати автономно для виконання закладених в ній функцій; на основі розробленої моделі архітектури розподіленої багаторівневої системи було реалізовано програмне забезпечення, яке використовувалось в локальній мережі ТОВ «ЮКС++»;

2) удосконалено моделі зловмисного програмного забезпечення на основі їх подання алгебрами поведінок, що дозволило створити основу бази поведінкових сигнатур, і на відміну від відомих представлень врахувало особливості їх функціонування в локальних мережах та здійснити класифікацію за типами поведінок;

3) вперше розроблено мережний метод виявлення файлового ЗПЗ, який для організації функціонування програмних модулів РБС, здійснення вилучення ймовірно уражених ПМ з РБС, встановлення відношення до файлового ЗПЗ на основі обміну і обробки знань, сканування виконуваних файлів створенням для них окремих процесів, базується на двох розроблених методах, які здійснюють побудову поведінкових сигнатур та їх подальший аналіз на наявність файлового ЗПЗ, в тому числі поліморфного та

метаморфного коду і для здійснення детальнішого аналізу програмного коду до процесу виявлення залучаються інші ПМ РБС.

Проведені експериментальні дослідження за участі Савенка О.С. надали можливість перевірити ефективність програмного забезпечення, розробленого на базі методів виявлення зловмисного програмного забезпечення;

Експериментальні дослідження показали, що розроблені методи виявлення зловмисного програмного забезпечення в локальних мережах дозволяють виявляти його існуючі та нові копії з вищим рівнем достовірності, ніж існуючі мережні антивірусні засоби.

Отримані результати дозволили підвищити достовірність виявлення зловмисного програмного забезпечення на 5-12% в порівнянні з існуючими мережними системами виявлення.

Ця довідка не є підставою для фінансових розрахунків.

Кльоц Ю.П.

АКТ
впровадження результатів дисертаційної роботи
Савенка Олега Станіславовича
«Теорія і практика створення розподілених систем виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах», представленої на здобуття наукового ступеня доктора технічних наук

Даний акт складений в тому, результати дисертаційної роботи Савенка О.С. «Теорія і практика створення розподілених систем виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах» використані при розробці продуктів напрямку IoT (WiFi, BLE, мікроконтролери з низьким споживанням) фірми «Cypress Semiconductors» та сервісного мікро-програмного забезпечення, зокрема:

1) модель архітектури розподіленої багаторівневої системи на основі принципів децентралізації та самоорганізації, яка дозволяє здійснювати наповнення всієї системи різними функціоналами виявлення зловмисного програмного забезпечення та збільшення кількості рівнів системи без зміни її архітектури;

2) метод виявлення бот-мереж на основі класифікації, який дозволяє виявляти і локалізувати бот-мережі в комп'ютерних локальних мережах на основі активного моніторингу системних подій та здійснення узгодження взаємодії програмних модулів розподіленої багаторівневої системи;

3) мережний метод виявлення файлового зловмисного програмного забезпечення для організації функціонування програмних модулів розподіленої багаторівневої системи, який забезпечує вилучення ймовірно уражених програмних модулів з розподіленої багаторівневої системи, встановлення відношення до файлового зловмисного програмного забезпечення ЗПЗ на основі обміну і обробки знань і сканування виконуваних файлів.

Компанії «Cypress Semiconductors» передано розроблені алгоритми і програмне забезпечення для реалізації вищенаведених моделей і методів. Їх впровадження дозволило підвищити достовірність виявлення зловмисного програмного забезпечення в середньому на 12% в порівнянні з існуючими мережними антивірусними програмними засобами.

Цей акт не є підставою для фінансових розрахунків.

Консультант фірми
Cypress Semiconductor, к.т.н _____ Кремінь В.Т.
28.02.2019

«Затверджую»
Проректор з науково-педагогічної та
наукової роботи, д.е.н., професор
_____ Войнаренко М.П.
«05» 04 2019 р.

АКТ
про впровадження в навчальний процес Хмельницького національного університету результатів дисертаційної роботи професора кафедри комп'ютерної інженерії та системного програмування
Савенка Олега Станіславовича
«Теорія і практика створення розподілених систем виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах»

Ми, комісія в складі: завідувача кафедри комп'ютерної інженерії та системного програмування, д.т.н., професора Говорущенко Т.О., к.т.н., доцента кафедри комп'ютерної інженерії та системного програмування Лисенка С.М., к.т.н., старшого викладача кафедри комп'ютерної інженерії та системного програмування Нічепорука А.О., склала акт про те, що результати дисертаційної роботи Савенка О.С. впроваджені та використовуються в навчальному процесі на кафедрі комп'ютерної інженерії та системного програмування для спеціальності 123 Комп'ютерна інженерія, зокрема, при викладанні навчальних дисциплін «Безпека та захист комп'ютерних систем», «Технічна діагностика і надійність комп'ютерних пристроїв та систем», «Паралельні та розподілені обчислення» та «Системне програмне забезпечення».

При викладанні цих дисциплін автором використовувались наступні матеріали досліджень, отримані ним особисто:

1) удосконалена модель архітектури розподіленої багаторівневої системи виявлення зловмисного програмного забезпечення в локальних комп'ютерних мережах на основі розподіленості, децентралізованості, багаторівневості та самоорганізованості, що, на відміну від відомих систем, дає змогу використовувати її автономно;

2) вперше розроблена модель архітектури програмних модулів та метод їх взаємодії в розподіленій багаторівневій системі виявлення зловмисного програмного забезпечення на основі структур Кріпке, особливістю яких є така самоорганізація, яка дає змогу здійснювати обмін знаннями всередині системи, що, на відміну від відомих систем, дозволяє використовувати знання, отримані окремими компонентами системи в інших частинах;

3) вперше розроблений метод взаємодії компонентів розподіленої багаторівневої системи, який дозволяє організовувати підтримку цілісності системи;

4) удосконалені моделі зловмисного програмного забезпечення на основі їх подання алгебрами поведінок, що дозволило створити основу бази поведінкових

сигнатур і, на відміну від відомих представлень, врахувало особливості їх функціонування в локальних мережах;

5) вперше розроблений метод виявлення бот-мереж на основі класифікації в комп'ютерних системах локальних мереж на основі пасивного моніторингу системних подій та здійснення узгодження взаємодії програмних модулів розподіленої багаторівневої системи при прийнятті рішення для спільного досягнення поставленої мети;

6) вперше розроблений мережний метод виявлення файлового зловмисного програмного забезпечення, який для організації функціонування програмних модулів розподіленої багаторівневої системи, здійснення вилучення ймовірно уражених програмних модулів з розподіленої багаторівневої системи, встановлення відношення до файлового зловмисного програмного забезпечення на основі обміну і обробки знань, сканування виконуваних файлів створенням для них окремих процесів, базується на двох розроблених методах, які здійснюють побудову поведінкових сигнатур та їх подальший аналіз на наявність файлового зловмисного програмного забезпечення, в тому числі поліморфного та метаморфного коду, і для здійснення детальнішого аналізу програмного коду до процесу виявлення залучаються інші програмні модулі розподіленої багаторівневої системи.

Отримані матеріали досліджень дозволили розробити лекційні курси та лабораторні практикуми по використанню систем виявлення зловмисного програмного забезпечення в локальних мережах та застосувати теорію і практику створення розподілених систем.

Говорущенко Т.О.

Лисенко С.М.

Нічепорук А.О.