Способи виявлення віртуалізації систем приманок

Назар Тимошик

Кафедра «Захист інформації», Національний університет "Львівська політехніка", УКРАЇНА, м.Львів, вул.С.Бандери, 12, E-mail: root.nt@gmail.com

Abstract – The general belief is that system emulators (such as Qemu) are more difficult to detect than traditional virtual machines (such as VMware) because they handle all instructions in software. In this paper, we seek to answer the question whether this belief is justified. In particular, we analyze a number of possibilities to detect system emulators. Our results shows that emulation can be successfully detected, mainly because the task of perfectly emulating real hardware is complex. This paper describes known attacks against the most widely used virtual machine emulators (VMware and Qemu).

Ключові слова – honeypots, qemu, detecting virtual machines.

I. Вступ

Використання віртуалізації має багато застосувань у сучасній мережній інфраструктурі, наприклад у центрах обробки даних, центрах тестувань програмного коду, системах хостингу та атомарних обчислень. Зокрема в галузі захисту інформації віртуальні машини використовуються для розгортання систем та мереж віртуальних приманок, та для аналізу та відлагодження коду вірусів, хробаків та іншого шкідливого мережного коду. Віртуальні машини реалізуються двома шляхами. Повна віртуалізація (емуляція) – чисто програмна реалізація. Паравіртуалізація – це передача базових інструкцій гостьової ОС напряму до апаратною частини під жорстким контролем гіпервізора.

II. Проблематика

Сучасні зразки шкідливого програмного забезпечення (анг. malware, укр. ШПЗ) оснащуються додатковою функціональністю для виявлення стану віртуалізованості операційної системи. Якщо таке ПЗ виявляє, що воно працює в віртуальній ОС, воно може змінювати свою поведінку, приховуватись або взагалі не запускатись, з метою ускладнення аналізу та обдурення вірусного аналітика.

Це ж саме стосується зловмисника, який через нову вразливість потрапляє на приманку. У випадку виявлення віртуалізованості операційної системи, він може здійснити ідентифікацію гіпервізора та провести атаку.

Найсерйознішими атаками проти віртуалізації, які можуть бути реалізовані шкідливим програмним забезпеченням та зловмисниками, є атаки відмови в обслуговуванні (DoS), тобто крах системи віртуалізації або віртуалізованої ОС.

Найцікавішою формою атаки може бути прорив ізоляції віртуалізованого середовища, та отримання доступу або контролю над хостовою ОС або гіпервізором. Досі випадків реалізації останньої не було офіційно представлено.

Відомі також спроби злочинного використання віртуальних машин. Subvirt – реалізація концепції Virtual Machine Based Rootkit (VMBR). ПЗ, яке розробили Microsoft [1] та як доведення концепції можливості розгортання в операційній системі гіпервізора, завдяки якому шкідливе ПЗ може взагалі не проявляти себе в цільовій ОС, оскільки вся його діяльність відбуватиметься з віртуалізованого середовища. Такий тип ШПЗ не виявляється системами виявлення втручань та антивірусним ПЗ.

Найбільша небезпека криється в паравіртуалізації з використанням новітніх процесорних підсистем, де гостьові системи мають прямий доступ до певних апаратних ресурсів. Наприклад, можливість переповнення буфера в драйвері мережного пристрою хостової ОС, коли гостьова ОС посилає спеціально сформований пакет, який обробляється драйвером хостової ОС.

III. Способи ідентифікації середовища

Проблема емуляторів в тому, що різні процесори по різному реагують на деякі специфічні інструкції. Наприклад, для архітектури Intel 80x86 інструкція AAA встановлює прапорці одним з трьох шляхів в залежності від того, чи це Pentium 2, Pentium 3, Pentium 4 чи старші версії центрального процесора.

Виконання циклічного повторення деяких інструкцій може тривати набагато довше в віртуалізованому середовищі, ніж на справжній апаратурі.

Найбільш наочною атакою проти більшості гіпервізорів є перевірка джерела локального часу, наприклад регістру "Time Stamp Counter" (TSC) [2]. Це стосується обох базових архітектур *Intel ma AMD*. Результатом є "TSCDelta" в полі "Virtual Machine Control Block" (VMCB), який може використовуватися, щоб вносити відхилення значення гостьового TSC, щоб приховати затримку, викликану затримками гіпервізора.

Системи віртуалізації руткітів, такі як BluePill і Vitriol [3] використовують специфіку апаратної віртуалізації, реалізованої для процесорів Intel VT і AMD Pasific.

Виявлення середовища віртуалізації на базі VMware базується на особливостях розміщення важливих структур даних, таких як Interrupt Descriptor Table (IDT) і Global Descriptor Table (GDT). Також в Windows версії використовується Local Descriptor Table (LDT). Простим способом виявити VMware є перевірка на ненульове значення LDT в Windows версії. Більш загальним є метод "RedPill" запропонований Рутковською [4] шляхом перевірки значення IDT на перевищення певного допустимого значення, що підштовхує до підозри на наявність гіпервізора. Однак цей метод виявляється не ефективним для багатоядерних машин. Метод "Scooby Doo" [5] схожий з методом "RedPill", але порівнює значення IDT з специфічним, зашитим в апаратну частину значенням.

Ще одним способом ідентифікації наявності VMware в системі є комунікаційні схеми хост-гість та гість-гість. Найбільш загальною формою виявлення цим методом є [6]:

```
mov eax, 564d5868h ;'VMXh'
mov ecx, 0ah ;отримання версії VMware
mov dx, 5658h ;'VX'
in eax, dx
cmp ebx, 564d5868h ;'VMXh'
je detected
```

При запуску на рівні привілеїв 3 в операційній системі з захистом операцій, такій як Linux чи Windows, виконання інструкції IN викликає генерацію виняткової ситуації, допоки рівень переривання вводу-виводу не зміниться. Це зумовлено тим, що переривання IN є привілейованою інструкцією. Ще один цікавий спосіб виявлення наявності VMware був реалізований в вірусі W32/Polip [7]

Також в гостьовій ОС Windows версії дуже багато ключів реєстру, які сигналізують про використання VMware. Класичними ознаками для ідентифікації гостьових систем на базі VMware є MAC-адреси з діапазону, виділеного для компанії VMware, назви блокових та дискових пристроїв які є константами.

Деякі інструкції та задачі в роботі ОС в віртуалізованому середовищі вимагають підвищених привілеїв для коректного виконання.

У 2005 році через некоректну перевірку запитів 'EPRT' та 'PORT' FTP сервісом VMnat стало можливим «переповнення купи» (heap buffer overflow) в середовищі хостової ОС, та можливість виходу за периметр віртуалізації та доступу до виконання коду в середовищі хостової ОС [8].

Іншим важливим емулятором, який би ми хотіли розглянути, є Qemu – повністю програмний емулятор віртуальних машин, який ізолює віртуальну машину від хостової, а також підтримує динамічну трансляцію інструкцій від процесора гостьової ОС хостовій ОС. Оскільки Qemu повністю програний емулятор, його суттєвою перевагою є незалежність архітектур процесора гостьової та хостової систем. У зв'язку з повною емуляцією апаратної частини він дещо програє решті відомих систем віртуалізацій в продуктивності, проте при використанні спеціального модулю KQEMU стає можливим передавати усі непривілейовані інструкції від гостьової ОС напряму процесору хостової системи, що суттєво підвищує швидкодію.

Використання динамічної трансляції є ризиком для ідентифікації у випадку само модифікованого коду, наприклад, само перезаписів REP послідовностей [9]]. Майже класичним способом ідентифікації Qemu є CPUID інструкція, яка повертає некоректне значення для процесорів AMD - це значення буде «QEMU Virtual CPU version x..x..x». В документації на процесори Intel є обмеження в 15 байт на довжину інструкції. В Qemu дане обмеження не реалізовано[]. За рахунок динамічної трансляції, блок інструкцій які виконуються перший раз виконується повільніше ніж наступного, за рахунок кешування трансльованих блоків Qemu. Також некоректна реалізації певних кодів операцій (н.п.00FFh) та розширення для відлагодження (з допомогою #DB та при спробі ділення на нуль) дозволяє легко виявити Qemu. Ще одним способом виявлення можна вважати реалізацію специфічних для кожної моделі процесору регістри.

Висновок

В даній статті наведено аналіз відомих методів виявлення факту віртуалізації операційних систем, що може бути оцінено зловмисником як потенційне використання даної системи в якості приманки, та може спровокувати реалізацію DoS атаки або іншої атаки на обхід ізольованості середовища. Рекомендаціями для боротьби з ідентифікацією віртуалізованості середовища, для прикладу в Qemu може бути заміна поточних значень ідентифікації процесора на будь-яку приблизно правдоподібну. Це ж саме стосується блокових пристроїв, віртуальних інтерфейсів і.т.д. Можливо в дана проблема зникне з часом, в силу того що віртуалізація проникає у всі сфери IT.

[1] Samuel T. King, Peter M. Chen, Yi-Min Wang, Chad Verbowski, Helen J. Wang, and Jacob R. Lorch http://www.eecs.umich.edu/virtual/papers/king06.pdf

[2] Danny Quist and Val Smith "Detecting the Presence of Virtual Machines Using the Local Data Table"

[3] VMware Bugcheck "Detecting hardware assisted hypervisors" http://www.rootkit.com/newsread.php?newsid=548

[4] Joanna Rutkowska "Red Pill" http://invisiblethings.org/papers/redpill.html

[5] Tobias Klein "Scooby Doo - VMware Fingerprint Suite" http://www.trapkit.de/research/vmm/scoopydoo/

[6] Peter Ferrie "Detecting hardware-assisted hypervisors without external timing"

[7] Peter Ferrie "Tumours and Polips" http://pferrie.tripod.com/vb/polip.pdf http://www.symantec.com/enterprise/security_

[8] Garfinkel, T., Adams, K., Warfield, A., Franklin, J.: Compatibility is Not Transparency: VMM Detection Myths and Realities. In: Proceedings of the 11th Workshop on Hot Topics in Operating Systems (HotOS-XI). (May 2007).

[9] Fabrice Bellard http://fabrice.bellard.free.fr/qemu