УДК 004.422:004.056.5

Г. В. Олійник[1], С. В. Грибков[1]
[1]Національний університет харчових технологій,
кафедра інформаційних систем

ВИКОРИСТАННЯ JWT-МАРКЕРІВ ДЛЯ АУТЕНТИФІКАЦІЇ ТА АВТОРИЗАЦІЇ КОРИСТУВАЧІВ У WEB-ДОДАТКАХ

Розглянуто проблеми реалізації механізмів аутентифікації та авторизації у web-орієнтованих системах. Здійснено огляд та порівняння підходів до аутентифікації та авторизації користувачів, розглянуто особливості та обґрунтовано переваги використання JWT-маркерів.

Ключові слова: інформаційна безпека, аутентифікація та авторизація, web-орієнтована система, JWT-маркер.

The problems of implementing of authentication and authorization mechanisms in web-oriented systems were considered. Analysis and comparison of users' authentication and authorization approaches were performed. Features of using of JWT tokens were examined and their advantages were proved.

Keywords: information protection, authentication and authorization, web-based information system, JWT token.

Вступ

Сучасний стрімкий розвиток інформаційних технологій спонукає до все більшого поширення web-орієнтованих інформаційних систем, які мають набагато більше переваг порівняно з традиційними. Автори розробляють web-орієнтовану систему підтримки прийняття рішень при плануванні виконання договорів, яка повинна забезпечувати можливість розв'язання основних задач прийняття рішень, а також їх підтримку необхідною інформацією у будь-якому місці, де є доступ до мережі Інтернет. Основним призначенням створюваної системи є підтримка планування виконання договорів з оцінкою виробничих можливостей підприємства та формування плану виконання нового договору таким чином, щоб узгодити та інтегрувати його з іншими здійснюваними на підприємстві роботами із забезпеченням мінімального часу простоїв та затримок для усіх договорів календарного періоду. Важливо зазначити, що економічна конкуренція на сучасному ринку вимагає швидкого реагування, оскільки більша частина клієнтів уже під час обговорення хоче знати строки виконання свого замовлення, а також мати можливість вносити необхідні корективи та відслідковувати усі зміни в оперативному режимі [1].

Враховуючи, що створювана система підтримки прийняття рішень при плануванні виконання договорів є web-орієнтованою, необхідно уважно поставитися до завдань, пов'язаних з організацією захисту даних, централізованим управлінням інформаційними ресурсами, розмежуванням доступу до ресурсів системи, управлінням сеансами доступу тощо. Розроблення сучасних web-орієнтованих систем потребує, з одного боку, відстеження змін в інформаційних технологіях, що постійно з'являються і швидко змінюються, а з іншого боку – врахування реальних характеристик апаратного й програмного забезпечення корпоративних мереж і систем.

Саме тому виникла задача обрання способу реалізації механізму аутентифікації та авторизації користувачів для web-орієнтованої системи підтримки прийняття рішень під час планування виконання договорів.

Аналіз досліджень та публікацій

З поширенням web-орієнтованих систем все гострішими стають проблеми їх безпеки, адже допущена під час проектування та створення елементів захисту навіть найменша помилка може призвести до таких небажаних наслідків, як несанкціонований доступ та модифікація корпоративних даних кіберзлочинцями, які постійно вдосконалюють способи та підходи для пошуку вразливостей та здійснення зламу. У роботі [2] розглянуто основні особливості захисту інформаційних ресурсів у корпоративних мережах і системах, а також запропоновано підхід до оцінювання його ефективності. Автори роботи [3] детально описали підходи до забезпечення доступу до інформаційних ресурсів, а також способи забезпечення і реалізації аутентифікації та авторизації у web-орієнтованих системах. Публікація [4] присвячена створенню модуля безпеки web-орієнтованої системи підтримки прийняття рішень при плануванні виконання договорів. У цій роботі проведено порівняльний аналіз програмних платформ для реалізації такого модуля. За результатами порівняння обрано програмну платформу Spring Security, що дає змогу реалізувати багаторівневий механізм захисту системи, а також надає можливості для забезпечення захищеності від поширених типів мережевих атак. У роботі [5] розглянуто використання JSON Web Token (JWT) маркерів, які в зашифрованому вигляді вміщують необхідну для аутентифікації та авторизації користувача інформацію.

Розглянуті публікації недостатньо висвітлюють сучасні альтернативні способи для реалізації механізмів аутентифікації та авторизації, оскільки за останні роки розробники web-систем все частіше відмовляються від використання файлів cookies і серверної сесії для аутентифікації користувачів та переходять до використання сучасних та продуктивніших щодо швидкодії системи підходів.

Формулювання цілі статті

Необхідно дослідити та проаналізувати особливості аутентифікації та авторизації користувачів у web-додатках, а також розглянути можливість використання JWT-маркерів для підвищення рівня захисту від поширених типів мережевих атак при створенні web-орієнтованої системи підтримки прийняття рішень при плануванні виконання договорів.

Особливості аутентифікації та авторизації користувачів у web-додатках

Основним методом аутентифікації в сучасних web-додатках є використання файлів cookie, що містять ідентифікатор сесії на сервері. При цьому сесія має термін дії, який автоматично збільшується у разі звернення користувача до сервера [5].

Аутентифікація на основі серверної сесії та cookie належить до групи підходів, які зберігають стан взаємодії між клієнтом і сервером, а також дані, які між ними передаються. За таким підходом дані про сесію зберігаються як на клієнтській, так і на серверній частині. Перелік активних сесій зберігається на сервері (наприклад, у базі даних), а на клієнтській частині створюється файл cookie, що містить ідентифікатор активної сесії.

У web-орієнтованих системах алгоритм взаємодії зареєстрованого користувача з використанням традиційного підходу складається з таких кроків.

Користувач вводить свої ім’я та пароль на формі для введення клієнтської частини. Введені дані передаються на обробку до модуля аутентифікації та авторизації серверної частини, де їх перевіряють, звертаючись до репозиторію даних та шукаючи користувача, який повністю відповідає отриманим даним. Якщо користувача з таким іменем та паролем знайдено, створюють та зберігають сесію, ідентифікатор якої передають назад до клієнтської частини. Дані про створену сесію зберігають на сервері у вигляді, наприклад, окремої таблиці в базі даних з необхідним набором полів, серед яких ідентифікатор сесії, термін її дії, відповідний ідентифікатор користувача. У браузері клієнта зберігається файл cookie з ідентифікатором створеної на сервері сесії. Зазвичай використовується стандартне ім’я такого файла – JSESSIONID.

Під час усіх наступних запитів користувача ідентифікатор сесії з файла cookie відправляється у заголовку запиту до серверної частини, де запит спочатку надходить до фільтра аутентифікації, а потім передається до модуля авторизації та аутентифікації. Тут обробляється отриманий ідентифікатор сесії, а саме здійснюється звернення до репозиторію даних з метою пошуку необхідних для визначення можливості доступу до певного ресурсу даних та безпосередньо перевіряється можливість доступу.

Якщо перевірка успішна, дані із запиту передають до модулів бізнес-логіки системи, де здійснюється їх подальша обробка з метою надання користувачу необхідної інформації.

Після закінчення користувачем роботи з web-додатком дані про сесію видаляються як на серверній, так і на клієнтській частинах.

Алгоритм взаємодії зареєстрованого користувача з web-орієнтованою системою при використанні традиційного підходу у вигляді діаграми послідовності зображено на рис. 1.

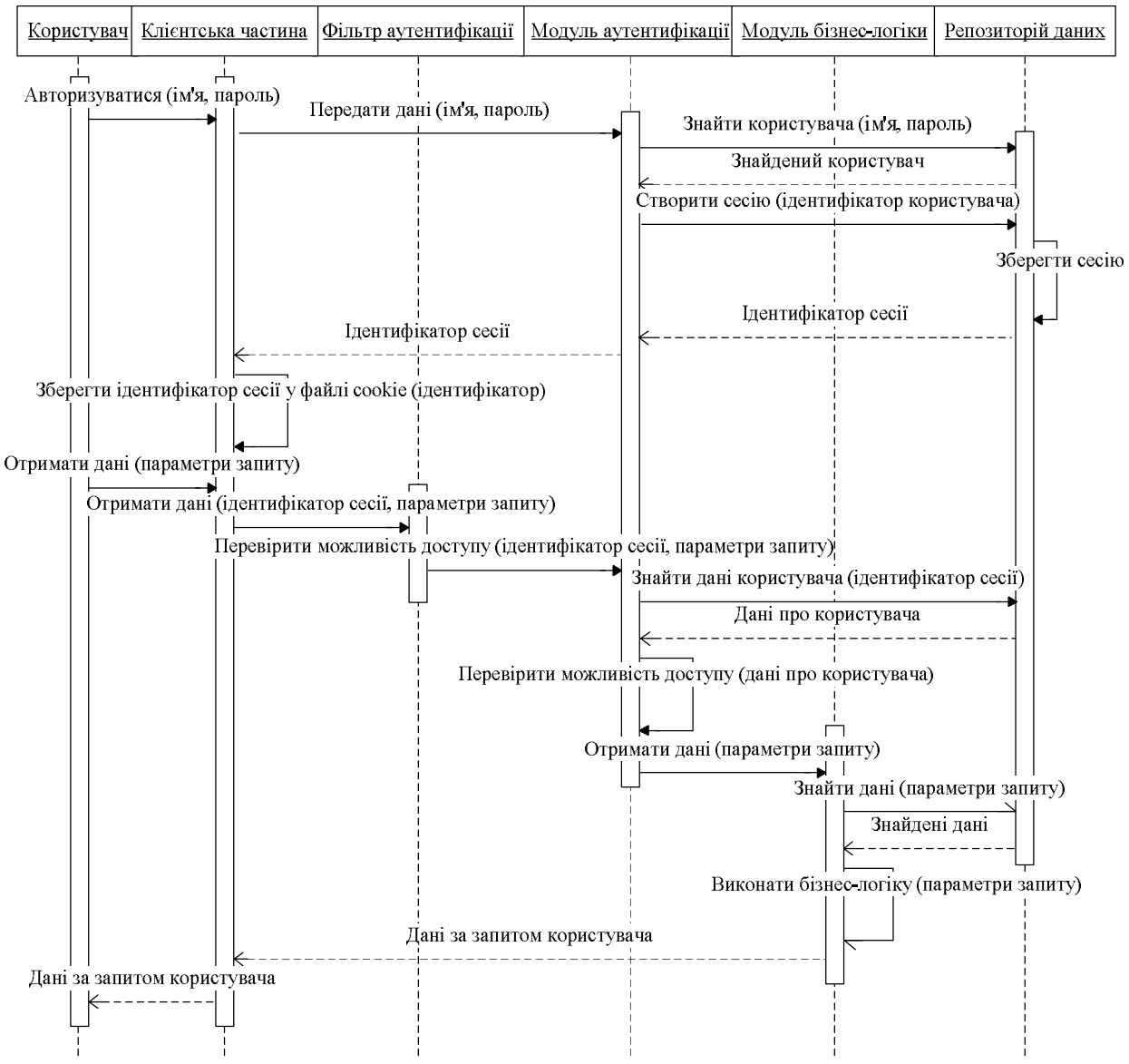

Рис. 1. Алгоритм взаємодії зареєстрованого користувача з web-орієнтованою системою при використанні традиційного підходу на основі серверної сесії

Важливо зазначити, що останнім часом виділяють три основні причини, що призводять до відмови від використання файлів cookie та серверної сесії.

Перша причина полягає у тому, що все частіше розробники реалізують взаємодію з серверною частиною за допомогою запитів через окремий інтерфейс, яким визначено усі функціональні можливості системи. Цей самий інтерфейс використовують і для мобільних додатків. Уніфікація підходу до аутентифікації користувачів при використанні традиційного підходу є доволі складним завданням, оскільки використання однакових файлів cookie у настільних додатках та на мобільних платформах є доволі проблематичним.

Друга причина визначається необхідністю у постійному зверненні до репозиторію даних під час кожного запиту для пошуку даних про користувача за отриманим ідентифікатором сесії, що негативно впливає на продуктивність та швидкодію всієї системи.

Третя причина має місце у випадку горизонтального масштабування web-системи, коли виникає проблема синхронізації стану сесій між серверами з метою підтримки актуального стану даних. Для вирішення цієї проблеми існують окремі підходи, проте створення додатка, який не потребує використання сесії взагалі, є значно простішим для його підтримки в майбутньому, а також дає змогу значно скоротити обсяг робіт під час вдосконалення та розширення.

З урахуванням усіх наведених причин актуальним завданням є дослідження альтернативних підходів реалізації аутентифікації та авторизації користувачів з усуненням вищезазначених недоліків.

Одним із сучасних підходів при реалізації механізмів аутентифікації та авторизації є використання спеціальних web-маркерів – JWT (JSON Web Token). JWT-маркер являє собою текстовий рядок, що містить у зашифрованому вигляді необхідні для аутентифікації та авторизації користувача дані.

Особливості використання JWT-маркера

JWT-маркер у текстовому вигляді складається з трьох частин, розділених між собою крапкою, а саме: заголовка (header), інформаційної частини (payload) та підпису (signature). Тобто, маркер має вигляд “header.payload.signature”. Кожна частина маркера кодується за допомогою методу кодування Base64.

Сформований JWT-маркер наведено на такому прикладі:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikhhbm5hIE9saWlueWsiLCJjb21wYW55IjoiTXkgQ29tcGFueSIsImF1dGgiOjQsImV4cCI6MTQ5Mjg4MzM0MX0.KRe-HIQFaQzwn1U-5e_tJkor9F-v-WHM3bG2ZPjgC9A

Заголовок містить тип маркера та назву алгоритму шифрування (наприклад, HS256) [6, 7].

JWT-маркери використовують криптографічні алгоритми для підпису вмісту заголовку та інформаційної частини. Відповідно до JWA (JSON Web Algorithms) специфікацією виділяють групи алгоритмів шифрування з використання симетричних та асиметричних алгоритмів.

До симетричних алгоритмів належать HS256, HS384, HS512, кожен з яких використовує відповідні алгоритми хешування HMAC SHA-256, HMAC SHA-384, або HMAC SHA-512. HMAC (hash-based message authentication code – хеш-код аутентифікації повідомлення) являє собою механізм перевірки цілісності інформації з використанням секретного рядка для встановлення автентичності даних, що передаються, у поєднанні з використанням хеш-функції [6].

Асиметричними алгоритмами є RS256, RS384, RS512 з використанням RSASSA-PKCS1-v1_5 – асиметричного криптографічного алгоритму. При цьому використовують пару, яка включає приватний і публічний ключі. Приватний ключ застосовуються при генеруванні підпису маркера, а публічний для перевірки автентичності отриманого маркера на серверній частині.

Окрім цього, до використовуваних алгоритмів належать ES256, ES384, ES512 на основі ECDSA – алгоритму з відкритим ключем, визначеним над групою точок еліптичної кривої.

Типовий заголовок у форматі обміну даних JSON (JavaScript Object Notation) має вигляд:

```
{
  "typ":"JWT",
   "alg":"HS256"
}
```

Інформаційна частина складається з наборів пар ключ-значення (JWT Claims Set), що в загальному випадку містять мінімально необхідні для аутентифікації та авторизації в системі дані про користувача, якому даний маркер належить. Інформаційна частина з чотирьох полів, що містять певні значення, має наведений на прикладі вигляд:

```
{
  "sub": "1234567890",
  "name": "Hanna Oliinyk",
  "company": "My Company",
  "auth": 4,
  "exp": 1492883341
}
```

Інформаційна частина може містити як стандартні поля, описані у специфікації (RFC 7519), так і довільні користувальницькі поля. У наведеному прикладі використані стандартні поля: sub – ідентифікатор користувача, name – ім'я користувача, exp – час закінчення дії маркера. Крім цього, використано два додаткові поля: auth, значення якого у числовому вигляді відповідає рівню доступу користувача у системі; company – містить назву підприємства, співробітником якого є власник облікового запису даного користувача.

Підпис створюється з використанням секретного рядка, заголовка та інформаційної частини. Він призначений для верифікації маркера, а також унеможливлює його підробку генерацією сторонніми системами.

Формування підпису можна описати за допомогою такого псевдокоду:

```
data = base64urlEncode(header) + "." + base64urlEncode(payload);
signature = hash(data, secret).
```

Отже, закодовані за допомогою Base64 заголовок та інформаційна частина об'єднуються через крапку в єдиний текстовий рядок. У наведеному псевдокоді цей рядок має назву data. Для отримання підпису виконується хешування цього рядка з секретним ключем та використанням алгоритму, вказаного у заголовку.

Основні кроки, виконувані під час роботи користувача з web-орієнтованою системою за умови використання JWT маркерів, можна описати так.

Як і за підходом із серверною сесією, користувач вводить свої ім'я та пароль на формі для введення клієнтської частини. Аналогічно з традиційним підходом введені дані передаються на обробку до модуля аутентифікації та авторизації серверної частини, де за допомогою звернення до репозиторію даних для пошуку користувача з відповідними даними їх перевіряють. У разі успіху формується JWT-маркер, до інформаційної частини якого вносяться усі необхідні дані (ідентифікатор користувача, роль у системі, час закінчення дії маркера тощо). Інакше кажучи, маркер містить дані про користувача та дії, які він може виконувати у системі, що є необхідним для визначення можливостей доступу під час усіх наступних запитів. Сформований маркер надсилається у відповідь та зберігається клієнтською частиною.

Під час кожного наступного запиту користувача отриманий маркер включається до заголовка авторизації (Authorization header) у форматі: Authorization: Bearer {маркер} та разом із параметрами запиту надсилається до серверної частини. Після отримання такого запиту фільтр аутентифікації передає його до модуля аутентифікації та авторизації, де відбувається перевірка коректності та дійсності маркера на основі його підпису та значення окремих полів, наприклад, поля з терміном дії. Правильність підпису маркера визначається з використанням алгоритму, вказаного у заголовку, за умови повної відповідності алгоритму, використовуваному серверною частиною. Якщо підпис дійсний, тобто, наприклад, сформований з використанням певного секретного рядка, або за допомогою використовуваного на сервері відкритого ключа, встановлюється автентичність маркера, відбуваються додаткові перевірки, такі як актуальність терміну дії, відповідність ролі користувача можливості звернення до певного ресурсу тощо. Інакше кажучи, успішне проходження перевірки дозволяє подальшу обробку запиту. Модулі, які реалізують бізнес-логіку системи,

отримують дані із запиту та виконують необхідні дії. Отримані в результаті дані повертаються у відповідь до клієнтської частини, де в деякому вигляді відображаються для користувача.

На рис. 2 на діаграмі послідовності відображено порядок взаємодії зареєстрованого користувача з web-орієнтованою системою у разі використання JWT-маркера.

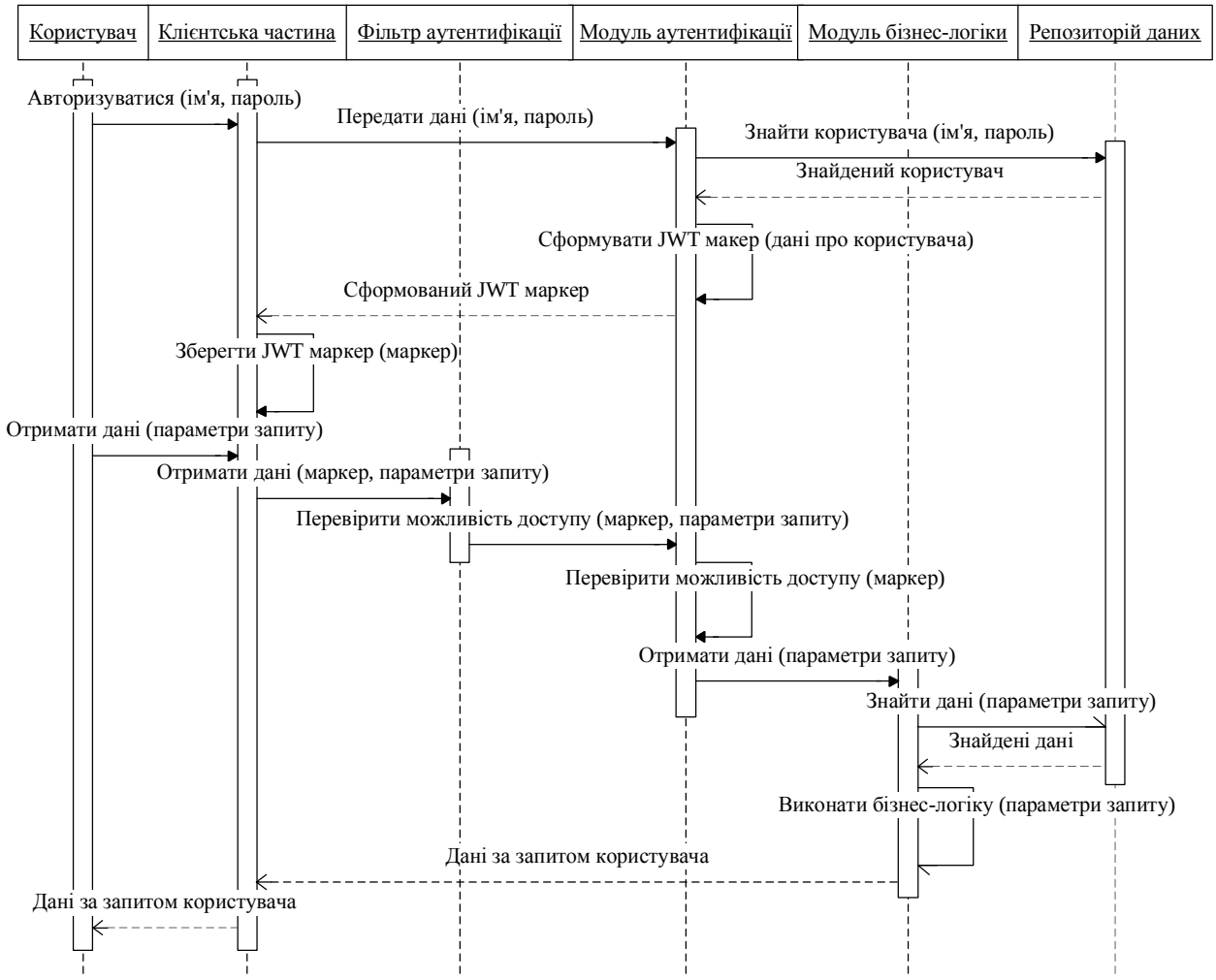

Рис. 2. Алгоритм взаємодії зареєстрованого користувача з web-орієнтованою системою при використанні JWT-маркера

Основною перевагою використання JWT-маркерів є відсутність необхідності зберігання будь-яких даних про маркери чи самі маркери на сервері. Кожен маркер містить всі необхідні дані для перевірки його достовірності, а також дає можливість передавати довільний набір даних. Серверна частина забезпечує формування, підпис та видачу JWT-маркерів, перевірку дійсності маркерів у вхідних запитах.

Недоліком використання маркерів є необхідність обмеження довжини заголовка у запиті для деяких серверів. У таких випадках до інформаційної частини маркера варто вводити лише мінімально необхідні дані. Крім цього, маркер містить термін дії, який відповідно до вимог безпеки повинен бути не дуже тривалим, а після його закінчення необхідно формувати новий маркер.

Реалізація використання JWT-маркерів

Сьогодні існує значна кількість готових реалізацій та бібліотек для роботи з JWT-маркерами з метою розроблення механізму аутентифікації та авторизації web-орієнтованої системи на їх основі.

У разі використання мови програмування Java для розроблення серверної частини системи, на прикладі системи підтримки прийняття рішень при плануванні виконання договорів, доволі зручною у використанні та доступною для розуміння є бібліотека Nimbus JOSE + JWT.

Розглянемо приклад практичної реалізації роботи з JWT-маркерами за умови використання алгоритмів з відкритим ключем.

Створюємо Java клас за назвою JwtConfiguration. Цей клас повинен містити текстове поле key, значення якого відповідає значенню секретного рядка, використовуваного як закритий ключ. Отже, викликаючи метод jwtConfiguration.getKey(), отримуємо його значення.

Для формування підпису маркера створюємо клас JwtMacSigner, що реалізує інтерфейс com.nimbusds.jose.JwtSigner, який пропонується використовуваною бібліотекою. При цьому необхідно описати логіку роботи єдиного методу sign(JWTClaimsSet claimSet). Тут аргумент claimSet – це набір полів інформаційної частини маркера. Дії для створення підписаного маркера показано на наступному прикладі коду.

```
JWSHeader header = new JWSHeader(JWSAlgorithm.HS256);
SignedJWT jwtObject = new SignedJWT(header, claimSet);
JWSSigner signer = new MACSigner(jwtConfiguration.getKey().getBytes());
return jwtObject.sign(signer);
```

У наведеному фрагменті коду спочатку створюється заголовок для маркера з використанням алгоритму HS256. Потім формується об’єкт маркера зі створеним заголовком. Для підпису маркера з використанням секретного рядка створюється і використовується екземпляр класу MACSigner. У результаті виконання описаних операції об’єкт jwtObject являтиме собою готовий підписаний JWT-маркер.

З метою отримання можливості перевірки маркера створюємо клас JwtMacVerifier. Для цього використовуємо інтерфейс com.nimbusds.jose.JwtVerifier та описуємо логіку методу verify(SignedJWT jwtObject). Цей метод повертає значення true, якщо перевірка маркера успішна, та false у протилежному випадку. Фрагмент коду, призначений для перевірки отриманого маркера, такий:

```
JWSVerifier verifier = new MACVerifier(jwtConfiguration.getKey().getBytes());
return jwsObject.verify(verifier);
```

У цьому прикладі створюємо об’єкт MACVerifier для перевірки і використовуємо його.

Якщо потрібно використати асиметричні алгоритми, практичну реалізацію можна здійснити так.

JwtConfiguration в такому випадку повинна надавати приватний та публічний ключі, які можуть зберігатися, наприклад, як окремі файли, або всередині сховища ключів (keystore) під деякими визначеними іменами.

Кроки для отримання підписаного маркера, використовуючи інтерфейс com.nimbusds.jose.JwtSigner, описуємо в класі JwtRSASigner, де реалізація методу sign(JWTClaimsSet claimSet) така:

```
JWSHeader header = new JWSHeader(JWSAlgorithm.RS256);
SignedJWT jwtObject = new SignedJWT(header, claimSet);
JWSSigner rsaSigner = new RSASSASigner(jwtConfiguration.getPrivateKey());
return jwtObject.sign(rsaSigner);
```

Використовуваний алгоритм у такому випадку – RS256, а для підпису маркера використовується екземпляр класу RSASSASigner зі вказаним приватним ключем.

Для перевірки маркера реалізуємо інтерфейс com.nimbusds.jose.JwtVerifier та створюємо клас JwtRSAVerifier, метод verify(SignedJWT jwtObject) якого містить такі кроки:

```
RSASSAVerifier rsaVerifier = new RSASSAVerifier(jwtConfiguration.getPublicKey());
return jwsObject.verify(rsaVerifier);
```

Замість класу MACVerifier використовуємо клас RSASSAVerifier відповідно до алгоритму шифрування.

Описані класи можуть бути застосовані модулем аутентифікації та авторизації системи та забезпечити основу роботи з JWT-маркерами. Отже, з використанням бібліотеки Nimbus JOSE +

JWT отримуємо можливість реалізації функціональності для формування та перевірки JWT-маркерів для web-орієнтованої системи, а також засіб для побудови механізму аутентифікації та авторизації на їх основі.

Висновок

У результаті проведених досліджень та аналізу особливостей використання JWT-маркерів для аутентифікації та авторизації користувачів було прийнято рішення про використання такого підходу для створення web-орієнтованої системи підтримки прийняття рішень під час планування виконання договорів, що підвищить рівень захисту інформації при роботі з web-орієнтованими системами та зменшить час обробки запитів. Окрім цього, під час розроблення механізму аутентифікації та авторизації користувачів для web-орієнтованої системи важливо запобігти низці концептуальних помилок, які часто допускають розробники, що дасть змогу спростити несанкціоноване отримання даних облікового запису. Отже, до основних складових забезпечення захисту системи, крім механізму аутентифікації та авторизації, також належать: контроль складності паролів; використання підходів до захисту від підбору паролів; передавача паролів тільки через захищене з'єднання; використання хеш-функції шифрування при зберіганні паролів; надання користувачам можливості зміни паролів та вчасне повідомлення про його зміну; вимога повторної аутентифікації користувача під час важливих дій, таких як зміна паролю, адреси та ін. Вибір надійного та сучасного підходу до аутентифікації та авторизації користувачів у поєднанні з загальною архітектурою модуля безпеки та врахуванням вищезазначеного дасть змогу реалізувати ефективний механізм захисту web-орієнтованої системи, що реалізується під час розроблення системи підтримки прийняття рішень під час планування виконання договорів.

1. Олійник Г. В., Грибков С. В., Литвинов В. А. Задача планування виконання договорів та підходи до її ефективного вирішення / Г. В. Олійник, С. В. Грибков, В. А. Литвинов // "Математические машины и системы". – К. : ІПММС НАНУ, 2015. – С. 61–70. 2. Кононова В. О., Грибков С. В., Харкянен О. В. Оцінка засобів захисту інформаційних ресурсів / В. О. Кононова, С. В. Грибков, О. В. Харкянен // Вісник Нац. ун-ту "Львівська політехніка". – 2014. – № 806. – С. 99–105. 3. Шелупанова А. А., Груздева С. Л., Нахаева Ю. С. Аутентификация. Теория и практика обеспечения доступа к информационным ресурсам. / А. А. Шелупанова, С. Л. Груздева, Ю. С. Нахаева. – М. : Горячая линия – Телеком, 2009. – 552 с. 4. Олійник Г. В., Грибков С. В., Литвинов В. А. Обрання програмної платформи для побудови модуля безпеки web-орієнтованої системи підтримки прийняття рішень / Г. В. Олійник, С. В. Грибков, В. А. Литвинов // Вісник Нац. ун-ту "Львівська політехніка", серія "Автоматика, вимірювання та керування". – 2016. – № 852. – С. 137–142. 5. JSON Web Token и sliding expiration в web-приложении [Електронный ресурс] – режим доступа: https://habrahabr.ru/post/267349, 2015. 6. Jones. M. JSON Web Token (JWT) [Електронний ресурс] / M. Jones, J. Bradley, N. Sakimura // Internet Engineering Task Force. – 2015. – Режим доступу до ресурсу: https://tools.ietf.org/html/rfc7519. 7. Jones. M. JSON Web Algorithms (JWA) [Електронний ресурс] / M. Jones // JOSE Working Group. – January 16, 2012. – Режим доступу до ресурсу: http://self-issued.info/docs/draft-ietf-jose-json-web-algorithms-00.html 8. Ado Kukic. Cookies vs Tokens: The Definitive Guide [Електронний ресурс] / Ado Kukic // auth0.com. – 2016. – Режим доступу до ресурсу: https://auth0.com/blog/cookies-vs-tokens-definitive-guide/. 9. Mikey Stecky-Efantis. 5 Easy Steps to Understanding JSON Web Tokens (JWT) [Електронний ресурс] / Mikey Stecky-Efantis // vandium software. – 2016. – Режим доступу до ресурсу: https://medium.com/vandium-software/5-easy-steps-to-understanding-json-web-tokens-jwt-1164c0adfcec