http://www.moderntechno.de/index.php/meit/article/view/meit29-01-074
DOI: 10.30890/2567-5273.2023-29-01-074

УДК 004.2

SOFTWARE DEVELOPMENT TO AUTOMATE JWT TESTING

РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ДЛЯ АВТОМАТИЗАЦІХ ТЕСТУВАННЯ JWT

Posuvailo M.-M.V./ Посувайло М.-М.В.
bachelor / бакалавр
Lviv Polytechnic National University,
Lviv, Stepan Bandera, 12, 79000
Національний університет "Львівська Політехніка",
Львів, Степана Бандери, 12, 79000

Анотація. *У статті розглядається JSON Web Token, відносно "молода", але популярна технологія, яка в основному використовується для відстеження сеансів користувачів.*

Метою цієї роботи є створення інтуїтивно зрозумілого програмного засобу, який може автоматизувати тестування JWT.

Для виконання цього завдання ми провели дослідження, під час якого зібрали якомога більше інформації про відомі вразливості та характеристики вже існуючих програм. Ми проаналізували їх і отримали чітке уявлення про кінцевий продукт.

Врешті-решт завдання було виконано. Фахівці з усього світу отримали доступ до інтуїтивно зрозумілої та багатофункціональної програми. На даний момент, це єдине програмне забезпечення, здатне автоматизувати тестування вже відомих і можливих вразливостей.

Ключові слова. *JWT, JSON Web Token, автоматизація тестування, вразливості, stateful cookie, stateless cookie, JwtTester, JSON Web Tokens, jwtcat, jwt_tool, jwtXploiter, JwtCracker.*

Вступ.

Авторизація тривалий час використовувала локально збережені сесії клієнтів (кукі). Користувачі вводять свої облікові дані, генерується унікальний ідентифікатор сеансу (кукі), зберігаємо його на стороні сервера та повертаємо користувачеві. Усі дані користувача повинні зберігатися на стороні сервера. Будь-яка служба, якій потрібна якась інформація про користувача, повинна звернутися до сховища даних. Якщо дані користувача централізовані, їх неможливо підробити. Крім того, це усуває проблему застарілих даних. Все зберігається в центральному місці.

Такий підхід називають – "stateful", протилежний підхід – "stateless".

Файли cookie із збереженням стану (далі - Stateful) [1] і файли cookie без збереження стану (далі - Stateless) - це прикметники, які описують, чи призначений сервер або певна програма для запису та запам'ятовування однієї чи кількох попередніх подій у певній взаємодії з користувачем, іншим комп'ютером чи програмою, пристроєм чи іншим зовнішнім елементом. Без збереження стану означає, що не існує записів про попередні взаємодії, і кожен запит на взаємодію має оброблятися виключно на основі отриманої інформації. Файли cookie із збереженням стану та без збереження стану походять від використання стану як набору умов у певний момент часу.

Коротко підсумуємо:

- Stateful - відстеження збереженої інформації, яка використовується для взаємодії із клієнтом.
- Stateless – жодна інформація не зберігається на стороні сервера.

Якщо повернутись саме до “stateful” підходу, для складних архітектур, такі кукі можуть спричинити проблеми: отримання даних із центрального сховища для кожної операції може стати проблемним рішенням, враховуючи кількість запитів, що будуть виконуватись щосекунди для отримання тої чи іншої інформації. Вони не тільки займатимуть обчислювальні потужності сервера, але й понижуватимуть пропускну здатність мережі. На додаток до цього, ми живемо у реальному світі, тому важливо враховувати те, що у будь-якій частині нашої інфраструктури може статись збій. Це означає, що нам потрібно витрачати додаткові ресурси на створення другого серверу, який у разі збою, підстраховуватиме головний хост.

Є декілька можливих рішень такої проблеми, однак в цій роботі, ми розглядатимемо лише використання Stateless Cookie, а саме JWT.

Основний текст.

Вирішенням раніше вказаних проблем може стати JSON Web Token [2]. Це закодований блок інформації, який може містити велику кількість даних (на відміну від Cookie) і має криптографічний підпис. Коли хост отримує токен, він може бути впевненим, що даним, які містить отриманий JWT, можна довіряти, оскільки вони підписані відправником. Жоден посередник не може змінити вміст JWT після його відправлення без наявності у нього ключа.

Огляд токена. JWT представлений у вигляді послідовності трьох секцій, розділених символами крапки ('.'). Кожна частина містить закодоване значення у форматі base64url, це означає, що токен може без проблем надсилатись через HTTPS протокол. Загалом токен містить:

- Заголовок
- Пейлоад
- Підпис

Тобто загальна схема JSON Web Token має наступний вигляд: aaa.bbb.ccc , де aaa – “Заголовок”, bbb – “Пейлоад” і ccc – “Підпис”.

Кожна із вище перечислених секцій може бути вразливою до того чи іншого типу атак, наприклад:

- *Заголовок*: зазвичай містить метадані про токен, наприклад алгоритм, який використовується для підпису. Загальна вразливість [3] в цій секції може виникнути, коли JWT підписуються за допомогою слабких алгоритмів або застарілих стандартів, які не є безпечними. У таких випадках зловмисник може використати вразливі місця в алгоритмі підпису, щоб зламати токен і отримати доступ до інформації в корисному навантаженні або навіть змінити її.
- *Пейлоад*: містить дані користувача та параметри, пов’язані з процесом автентифікації та авторизації. Потенційні вразливості в цій секції можуть включати незахищене зберігання даних, несправний контроль доступу та викрадення сеансу. Зловмисник може отримати доступ до інформації користувача, перевизначивши претензії, зроблені в JWT, і змінивши їх на бажані значення. Це може поставити дані користувача під загрозу та потенційно призвести до витоку даних.
- *Підпис*: ця секція включає зашифрований хеш повідомлення розділів заголовка та пейлоаду. Поширена вразливість у цьому розділі може бути

пов'язана зі схемою керування ключами, яка використовується для створення підпису. Якщо зловмисник може викрасти закритий ключ, який використовується для підпису JWT, він може створити новий маркер і підписати його вкраденим ключем.

Враховуючи те, що JWT є об'ємною технологією, зрозуміло, що і площина для пошуку вразливостей також є великою. У мережі Інтернет немає конкретного списку із усіма можливими вразливостями, але за моїми дослідженнями, їх щонайменше 15.

Тому, звичайно ж, такий об'єм ручної роботи потрібно автоматизовувати, і певні напрацювання є. Давайте перейдемо до них, і розберемось наскільки вони хороші, яким критеріям відповідають та які вразливості здатні знайти.

Огляд утиліт для тестувань на вразливості. Загалом, ми розглянемо наступні скрипти: JSON Web Tokens (Burp's extention), jwtcat, jwt_tool, jwtXploiter і jwtcracker. Більшість із них є безкоштовними і розробленими на мові програмування Python, однак є і виключення, але до цього ми ще дійдемо.

JSON Web Tokens (Burp's extention)

До основних функцій можна віднести [4]:

- Зміна алгоритму з RSA на HMAC (CVE-2016-10555). Зловмисник може використовувати це, щоб обхитрити сервер і використовувати відкритий ключ для перевірки.
- Відсутність алгоритму (CVE-2015-2951). Це може призвести до того, що сервер взагалі не буде виконувати перевірку підпису і відразу перейде до обробки корисного навантаження токена.
- Відсутність підпису (CVE-2020-28042). Спосіб схожий на попередній, але відмінність у тому, що цього разу перша частина токена не змінюється, третя просто відкидається.
- Перевірка маркерів. Перевірка підпису та перших двох частин токена.
- Редагування токена. Якщо у вас є секретне слово чи ключ, можна змінити вміст токена і підпис залишитися дійсним.
- Ін'єкція ключа (CVE-2018-0114). Ми можемо додати наш ключ до першої частини токену. Вразливий сервер використовує його для верифікації підпису.
- Перевірка на погані налаштування. Закінчується тестування багатьох варіантів поганого налаштування веб-токена JSON.

Jwtcat

До його функціональності можна віднести лише три компоненти [5]:

- Відсутність алгоритму (CVE-2015-2951). Це може призвести до того, що сервер взагалі не буде перевіряти підпис і відразу перейде до обробки токена.
- Bruteforce для HS256. Зловмисник може спробувати вгадати секрет, який використовувався для створення підпису в цьому шифрі. Ця атака може використовувати або список найбільш розширених слів, або грубу силу.
- Редагування токена. Якщо у вас є секретне слово або ключ, ви можете змінити вміст маркера, і підпис залишиться дійсним.

JWT Tool

До функціональності можна віднести [6]:

- міна алгоритму з RSA на HMAC. Зловмисник може використати це, щоб змусити сервер використовувати відкритий ключ для перевірки.
- Відсутність алгоритму (CVE-2015-2951). Це може призвести до того, що сервер взагалі не виконує перевірку підпису та негайно переходить до обробки корисного навантаження маркера.
- Відсутність підпису (CVE-2020-28042). Спосіб схожий на попередній, але відмінність у тому, що цього разу перша частина токена не змінюється, третя просто відкидається.
- Спричинення помилок. Цей метод атаки покликаний на тестування повідомлень про помилки. Ці повідомлення можуть містити цінну інформацію.
- Атака грубого перебору для HS256. Зловмисник може спробувати вгадати секрет, використаний для створення підпису в цьому шифрі.
- Перевірка токена. Перевірка підпису та перших двох частин токена.
- Підробка позначки часу.
- Редагування токена. Якщо ви маєте секретне слово чи ключ, можливо змінити вміст токена і підпис залишиться валідним.
- Ін'єкція ключа (CVE-2018-0114). Ми можемо додати наш ключ у першу частину токена. Вразливий сервер використає його для верифікації підпису.
- Генерація ключів RSA і ECDSA.

jwtXploiter

Присутня наступна функціональність [7]:

- Редагування токена. Маючи всі необхідні компоненти на місці, ми можемо змінювати вміст токена.
- Експлуатація вразливих заголовків. Якщо JWT містить одне з потенційно вразливих значень у другій частині токена, ми можемо спробувати використати його.
- Перевірка токена. Перевірка підпису та перших двох частин токена.
- Видобуток публічного ключа через SSL з'єднання. Ця утиліта здатна самостійно отримати публічний ключ із веб-сервера.
- Підтримує усі алгоритми підпису та шифрування

JWT Cracker

На жаль, ця утиліта не має велику кількість функцій, однак вона є найефективнішою для [8]:

- Перебір секретів для HS алгоритмів.

Порівняння програмних засобів. Співставивши наявний список відомих вразливостей і функціонал усіх розглянутих утиліт, можна дійти до висновку, що жодна із них не покриває виявлення усіх вразливостей. Після практичного використання, також можна додаьи, що зазначені програмні засоби можуть бути важкими у використанні, особливо для людей із малою кількістю досвіду.

Для того, щоб вирішити описані проблеми, ми вирішили написати власну утиліту, яка б покривали увесь спектр вразливостей та була інтуїтивно зрозумілою у використанні.

За результатами випробувань, JwtTester довела, що здатна знаходити усі потрібні вразливості. Наявність мануалу робить її зрозумілою навіть для початківців. Результати тестувань наведені нижче:

Таблиця 1 – Порівняння утиліт

	JwtTester	Burp	jwtcat	Jwt_tool	jwtXploiter	Jwt cracker
RSA to HMAC	+	+		+	+	
Alg: None	+	+	+	+		
Strip sign	+	+		+	+	
Errors	+			+		
Brute for HS256	+		+	+		+
Verify token	+	+		+	+	
Timestap tampering	+			+		
Path traversal	+					
SQLi	+					
OS Injection	+					
SSRF	+					
JWT Editor	+	+	+	+	+	
Price	Free	Free	Free	Free	Free	Free
Key injection vulnerability	+	+		+		
Platform	Any OS	Burp	Any OS	Any OS	Linux	Any OS
Blank password	+	-	-	+	+	
RSA key generation	+			+		
ECDSA key generation	+			+		
Interface	CLI	GUI	CLI	CLI	CLI	CLI
kid	+(sqli, path,rce,ssrf)			+(sqli, path,rce)	+(path,sqli)	
jku	+			+	+	
x5u	+				+	
x5c	+					
jti	+					

Висновки.

Було розглянуто теоретичну інформацію щодо JWT, відомі вразливості та наявні програмні рішення для знаходження слабких місць. Також ми проаналізували та порівняли утиліти для тестування токену.

В результаті, було отримано ряд показників, із яких ми виділили речі, які вирішили покращити. Після написання коду та завершальних тестувань, результати показали, що поставлене завдання було виконано.

Література.

1. Java M. stateful, stateless, cookie and session. Bit and Bytes. URL: https://sethuramanmurali.wordpress.com/2013/07/07/stateful-stateless-cookie-and-

session/.

2. JSON Web Tokens. *Auth0*. URL: https://auth0.com/docs/secure/tokens/json-web-tokens.

3. JWT attacks | Web Security Academy. *PortSwigger*. URL: https://portswigger.net/web-security/jwt#:~:text=JWT%20vulnerabilities%20typically%20arise%20due,many%20implementation%20details%20for%20themselves.

4. JSON Web Tokens. *PortSwigger*. URL: https://portswigger.net/bappstore/f923cbf91698420890354c1d8958fee6.

5. GitHub - aress31/jwtcat: A CPU-based JSON Web Token (JWT) cracker and - to some extent - scanner. *GitHub*. URL: https://github.com/aress31/jwtcat.

6. GitHub - ticarpi/jwt_tool: :snake: A toolkit for testing, tweaking and cracking JSON Web Tokens. *GitHub*. URL: https://github.com/ticarpi/jwt_tool.

7. GitHub - DontPanicO/jwtXploiter: A tool to test security of json web token. *GitHub*. URL: https://github.com/DontPanicO/jwtXploiter.

8. GitHub - lmammino/jwt-cracker: Simple HS256 JWT token brute force cracker. *GitHub*. URL: https://github.com/lmammino/jwt-cracker.

Abstract. *This work involves the creation of an intuitive software tool for testing JSON Web Tokens (JWT), which is a solution to traditional authorization problems with cookies. Despite their security features, each aspect of JWT can be vulnerable to attacks, requiring an automated solution to address over 15 known vulnerabilities. JwtTester, a utility that can identify all necessary vulnerabilities and is easy to use, was developed. This work reviewed theoretical information, known vulnerabilities, and software solutions to identify weaknesses in JWT. After thorough testing, the final product successfully completed the original aim.*

Key words. *JWT, JSON Web Token, testing automation, vulnerabilities, stateful cookie, stateless cookie, JwtTester, JSON Web Tokens, jwtcat, jwt_tool, jwtXploiter, JwtCracker.*