Assignment 3

Venkata Diwakar Reddy Kashireddy

***Abstract*—In this lab, we learned about JWT tokens and refreshing an access token. There are two types of tokens: an access token and a refresh token. We completed tasks related to potential vulnerabilities related to JWT tokens.**

I. Introduction

IN this lab we were asked to perform attacks on the types of vulnerabilities in the area of injection and authentication: JSON web tokens. We learned about injection flaws and performed the attacks on WebGoat. The security of web applications often depends on the integrity and management of authentication tokens, specifically JSON Web Tokens (JWTs). Understanding the manipulation and refreshment of these tokens is crucial for both protecting and penetrating security barriers. We performed some task to get a deeper understanding of the potential vulnerabilities related to JWT tokens.

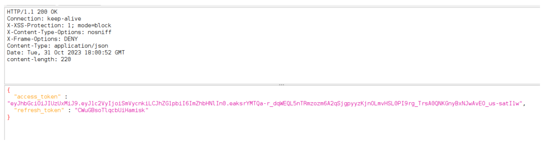
II. Tools

• KVM (Kernel-based Virtual Machine): It is a full virtualization solution for Linux. [1]
• WebGoat: A deliberately insecure web application maintained by OWASP designed for teaching web application security concepts. [2]
• Overleaf: It is a collaborative cloud-based LaTeX editor that helps to create documents easily by providing standard formats. [3]
• GitHub: GitHub is an Internet hosting service for software development and version control using Git. [4]
• Red: – IU Research Desktop (RED) is a virtual desktop service for users with accounts on the Carbonate research supercomputer at IU. [5]
• ZAP: OWASP ZAP (Zed Attack Proxy) is an open-source web application security scanner. It is used by those new to application security and professional penetration testers. [6]
• Firefox DevTools: Firefox Developer Tools is a set of web developer tools built into Firefox. It can be accessed to inspect the web page. [7]

III. Refreshing a token

It's important to know how to refresh an access token properly. This task comes from finding a vulnerability issue in a private bug bounty program on Bugcrowd. We got a log from a past security breach that had the token in it. The goal was to figure out how to buy books so that Tom would be charged for them. When we looked at the checkout process with ZAP, we noticed an error that said "Bearer null" and "JWT is not valid."

In the POST request, /JWT/refresh/login endpoint, we observed the user was 'jerry' and there were two tokens: access

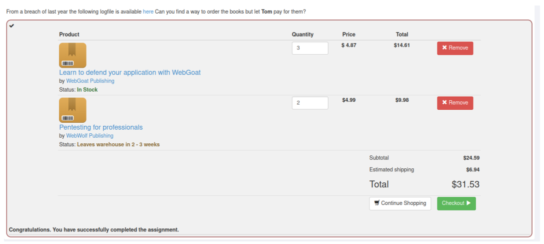


```
HTTP/1.1 200 OK
Connection: keep-alive
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Tue, 31 Oct 2023 18:00:52 GMT
content-length: 220

{
  "access_token" :
"eyJhbGciOiJIUzUxMiJ9.eyJlc2VyIjoiSmVycnkiLCJhZGlpbiI6ImZhbHNlIn0.eaksrYMTQa-r_dqWEQL5nTRmzozm6A2qSjgpyyzKjnOLmvHSL0PI9rg_TrsA0QNKGnyBxNJwAvEO_us-satI1w",
  "refresh_token" : "CWuGBsoTlqcbUiHamisk"
}
```

Fig. 1. Jerry login request and tokens.

tokens and refresh tokens. I tried to use these tokens to buy books as Tom by changing some token settings with a tool called webwolf JWT. I changed the following: alg: None, User: Tom, and no signature. This worked, and the payment was done with Tom successfully.



From a breach of last year the following logfile is available here Can you find a way to order the books but let **Tom** pay for them?

Product	Quantity	Price	Total	
Learn to defend your application with WebGoat by WebGoat Publishing Status: In Stock	3	$ 4.87	$14.61	✖ Remove
Pentesting for professionals by WebWolf Publishing Status: Leaves warehouse in 2 - 3 weeks	2	$4.99	$9.98	✖ Remove
		Subtotal	$24.59	
		Estimated shipping	$6.94	
		Total	$31.53	

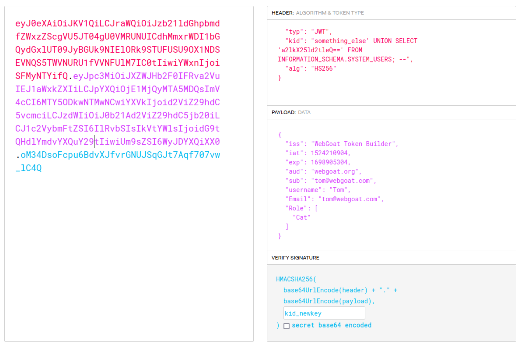
Congratulations. You have successfully completed the assignment.

Fig. 2. Successfully completed the tasks.

However, we were asked to update Tom's access token from the log and use Jerry's refresh token from the lesson page. I also tried a different way by changing Tom's token's expiry to now and the algorithm to 'None'. This way, I could use ZAP to place an order as Tom.

IV. JWT Token Final Challenge

The task was to delete Tom's account by manipulating a JWT token. When clicked on the delete button, it will give an error indicating a signature mismatch in the POST request. So, ZAP was used to intercept the POST request. The JWT token from the error is analyzed on jwt.io, revealing potential keys for manipulation such as kid, sub, username, and email.



```
eyJ0eXAiOiJKV1QiLCJraWQiOiJzb21ldGhpbmd
fZWxzZScgVU5JT04gU0VMRUNUICdhMmxrWDI1bG
QydGxlUT09JyBGUk9NIElORk9STUFUSU9OX1NDS
EVNQS5TWVNURU1fVVNFUlM7IC0tIiwiYWxnIjoi
SFMyNTYifQ.eyJpc3MiOiJXZWJHb2F0IFRva2Vu
IEJ1aWxkZXIiLCJpYXQiOjE1MjQyMTA5MDQsImV
4cCI6MTY5ODkwNTMwNCwiYXVkIjoid2ViZ29hdC
5vcmciLCJzdWIiOiJ0b21Ad2ViZ29hdC5jb20iL
CJ1c2VybmFtZSI6IlRvbSIsIkVtYWlsIjoidG9t
QHdlYmdvYXQuY29tIiwiUm9sZSI6WyJDYXQiXX0
.oM34DsoFcpu6BdvXJfvrGNUJSqGJt7Aqf707vw
_lC4Q
```

HEADER: ALGORITHM & TOKEN TYPE

```
  "typ": "JWT",
  "kid": "something_else' UNION SELECT
'a2lkX25ld2tleQ==' FROM
INFORMATION_SCHEMA.SYSTEM_USERS; --",
  "alg": "HS256"
}
```

PAYLOAD: DATA

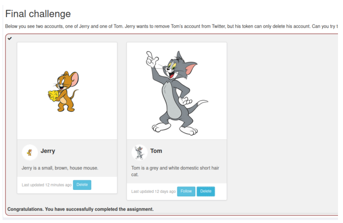
```
{
  "iss": "WebGoat Token Builder",
  "iat": 1524210904,
  "exp": 1698905304,
  "aud": "webgoat.org",
  "sub": "tom@webgoat.com",
  "username": "Tom",
  "Email": "tom@webgoat.com",
  "Role": [
    "Cat"
  ]
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  kid_newkey
) ☐ secret base64 encoded
```

Fig. 3. Modified JWT token.

The ’kid’ parameter, which is a key identifier, is identified as a candidate for tampering to create a valid JWT token. WebGoat hints suggest manipulating the ’kid’ parameter using SQL injection. A SQL query in WebGoat’s source code concatenates the ’kid’ value directly, making it vulnerable to SQL injection. An SQL injection string is crafted to force a new key, basically creating a new valid token. The injection fails the original query intentionally, allowing the UNION SELECT to return an arbitrary value, which is the new key. Since the key is base64 decoded in the WebGoat source code, the new key is base64 encoded before it is used in the JWT token. A new JWT token is crafted on jwt.io using Tom’s details: email, user, sub, and alg to ”None”, and the base64 encoded ’kid’ in the JWT header, and a clear text key for the signature.


Final challenge

Below you see two accounts, one of Jerry and one of Tom. Jerry wants to remove Tom's account from Twitter, but his token can only delete his account. Can you try t

Jerry

Jerry is a small, brown, house mouse.

Last updated 12 minutes ago

Tom

Tom is a grey and white domestic short hair cat.

Last updated 12 days ago

Congratulations. You have successfully completed the assignment.

Fig. 4. Successfully completed the final task.

After creating the new valid JWT token, it was replaced with the original token, and the request was forwarded and it successfully completed the task of deleting Tom’s account.

V. Conclusion

In this lab, two tasks on JSON web tokens were performed on webgoat. We learned about refreshing an access token. There are two types of tokens: an access token and a refresh token. Access tokens are used for API requests and have a limited lifespan. To address this, refresh tokens with a longer lifespan are employed. When managing JWTs, it is recommended to lock the algorithm to prevent client alterations. Using a symmetric key of adequate length for token signing is crucial. Avoid adding personal data to token claims; if additional information is necessary, consider token encryption.

References

[1] KVM
https://www.linux-kvm.org/page/MainPage..
[2] WebGoat
https://owasp.org/www-project-webgoat/..
[3] OverLeaf
https://www.overleaf.com/.
[4] Github
https://en.wikipedia.org/wiki/GitHub.
[5] RED
https://kb.iu.edu/d/apum.
[6] OWASP ZAP
https://www.zaproxy.org/
[7] Firefox DevTools
https://firefox-source-docs.mozilla.org/devtools-user/