

# **CITS1003 Project Report**

Student ID: [24412097]

Student Name: [Kiet Le]

# Part 1 – Cryptography

## Advanced Emu Standard

### Step 1

Following the rule that each character in a plaintext is considered as 8 bits, or 1 byte. Considering that the given command “deactivate\_special\_procedure\_123“ is in fact 32 characters long, therefore having 32 bytes (according to ASCII). We need to make two 16 byte pieces in order to put into the website. this will look like this:

First half: deactivate\_speci

Second half: al\_procedure\_123

### Step 2

We will then need to go to the given website “<http://34.87.251.234:3001>”.

After that we will then need to insert both the first and second half of the 16 byte commands in the “command Encrypter” part in the website, then press the blue “get Ciphertext” button.

The website will give result of the following for the first half:

3155433d53ed30c89aef89b2e7273924

And the second half:

4127efafc809cc1209376d039e0001f1

### Step 3

With the given ciphertext we will need to understand that to transmit and get the final result we must first need to combine the two parts of the given ciphertext together. Like this:

3155433d53ed30c89aef89b2e72739244127efafc809cc1209376d039e0001f1

Now just Input the full cipher text into the “transmit encrypted command” input space. (see attached image)

#### Transmit encrypted command

```
3155433d53ed30c89aef89b2e72739244127efafc809cc1209376d039e0001f1
```

**Transmit command**

#### Step 4

Input the full plain text of “deactivate\_special\_procedure\_123“ into the “Command Encryptor” input space. (see attachment)

#### Command Encryptor

```
deactivate_special_procedure_123
```

**Get ciphertext**

#### Step 5

Proceed to click the “transmit command” button after inputting the content in step 4 and 5. This will provide you with the following result.

#### Result:\*\*

Self-destruct protocol successfully deactivated!

UWA{3CB\_i5\_bL0cK\_Ind3peNd3nt!}

#### Flag Found:

UWA{3CB\_i5\_bL0cK\_Ind3peNd3nt!}

## Emu Cook Book

#### Step 1

Firstly it is important that to decode something you first must identify what type of code it is. Given that the given code has an ending of “+f4AgAAA==” this would only tell that it is a “Base64” text.

Therefore the identified given code is Base64.

## Step 2

In order to decode this we will need to input the code into “CyberChef.io”.

Which will look something like this (see attached image):

The screenshot shows the CyberChef web application interface. On the left, there's a sidebar with various operations like 'Search...', 'Favourites' (which includes 'To Base64' and 'From Base64'), and 'Data format'. The main area is titled 'Recipe' and contains two panes: 'Input' and 'Output'. The 'Input' pane contains the Base64 encoded string: H4sIAEwyx2UA/7VWx/LyIAu8Td+tpaCp33g/97/CHY3hYQVne/15z1VF EKJ35R42D/TEd91xSNv-j6GJ02Ka1dfy7GfhYeZi071smKGjuyXB3fMrbf36fDPEHwVAxx0dA/Ae0f05ibusy91cwAeaKrRleptddrc0JGdx1PfAmhB6P2imf26bv0Yb2928AsuscxXQwrcvq1xEzjjoozEp01hkt44oJ6rdQ-Qe57KYABvUsu1GxHwEVAMe8rlh1N1N42YspGyBMLlbuNAyMTF8In1HaRmmExGt9e9ouxXeAHM1OWhFF7mp01h811zuukejfp01npn0a0Be4ZfIRNZSk0t+hF19fUEKF15mDQNGVQjQSVFLBQp0jhtOHgoEM0R8pFXaevsfqanA7euVbzS0111j01V1RLXx+Vt1duZaiRf+jrYqLs+Poy+1u!uYHmDUOKH+rM2g1DNUctLV2EcBw+1T0a1f1jy9TPgXzmBpu513rlnvYFVE.lHcu041A7xjgevhplbdv3kjhHTCcEZmavUQhj4qhoyiCvv9xb7hv=0Tu8phab0r7A+82QKGZ6c6c404Lu0202mgU36uRmBwTxRfokary/3htSG1c5NueVgt7ix0Dp34Y2qt/jc/k1Ivjk7HEN/n3zpm45TGvYxt9eh+f4AgAAA==. The 'Output' pane shows the decoded string: H4sIAEwyx2UA/7VWx/LyIAu8Td+tpaCp33g/97/CHY3hYQVne/15z1VF EKJ35R42D/TEd91xSNv-j6GJ02Ka1dfy7GfhYeZi071smKGjuyXB3fMrbf36fDPEHwVAxx0dA/Ae0f05ibusy91cwAeaKrRleptddrc0JGdx1PfAmhB6P2imf26bv0Yb2928AsuscxXQwrcvq1xEzjjoozEp01hkt44oJ6rdQ-Qe57KYABvUsu1GxHwEVAMe8rlh1N1N42YspGyBMLlbuNAyMTF8In1HaRmmExGt9e9ouxXeAHM1OWhFF7mp01h811zuukejfp01npn0a0Be4ZfIRNZSk0t+hF19fUEKF15mDQNGVQjQSVFLBQp0jhtOHgoEM0R8pFXaevsfqanA7euVbzS0111j01V1RLXx+Vt1duZaiRf+jrYqLs+Poy+1u!uYHmDUOKH+rM2g1DNUctLV2EcBw+1T0a1f1jy9TPgXzmBpu513rlnvYFVE.lHcu041A7xjgevhplbdv3kjhHTCcEZmavUQhj4qhoyiCvv9xb7hv=0Tu8phab0r7A+82QKGZ6c6c404Lu0202mgU36uRmBwTxRfokary/3htSG1c5NueVgt7ix0Dp34Y2qt/jc/k1Ivjk7HEN/n3zpm45TGvYxt9eh+f4AgAAA==. Below the panes are buttons for 'STEP', 'BAKE!', and 'Auto Bake'.

## Step 3

We will then need to select an operation to decode the Base64 code. Knowing that it is already in Base64 format we will need to search for “Base64” in the search bar (see attached image):

Operations	Recipe	Input
Base64		H4sIAEwyx2UA/7VWX/LyIAw8Td+tpaCP33gS/97/CHY3hYQVn/... (long Base64 string)
To Base64		
From Base64		
Show Base64 offsets		
Fork		
From Base32		
From Base58		
From Base85		

## Step 4

We will then need to click and drag the “from Base64” operation into the “recipe” section. This is because the given code is already in Base64 format, so we will need to decode it from Base64 into something else.

Once selected it should look something like this (see attached image):

Operations	Recipe	Input	Output
Base64		H4sIAEwyx2UA/7VWX/LyIAw8Td+tpaCP33gS/97/CHY3hYQVn/... (long Base64 string)	
To Base64			
From Base64	From Base64		
Show Base64 offsets			
Fork			
From Base32			
From Base58			
From Base85			
Parse SSH Host Key			
To Base32			
To Base58			
To Base85			
<b>Favourites</b>			
Data format			
Encryption / Encoding			
Public Key			
Arithmetic / Logic			
Networking			
Language			
Utils			
Date / Time			

## Step 5

Identifying the given output in the output box it is required that we use “Gunzip” to further decode the text.

To do this we will simply click on the “wand” button next to the output box.

Here's a red circle and arrow to show you (see attached image below):

The screenshot shows the Reversing Shell application interface. On the left, there is a sidebar with various operations like 'Search...', 'Favourites', 'To Base64', 'From Hex', etc. In the center, there is a 'Recipe' section titled 'From Base64' with options for 'Alphabet' (set to 'A-Za-z-0-9+=') and a checked 'Remove non-alphabet chars' option. Below this is a large input text area containing a long, encoded string of characters. To the right of the input is an output text area, also containing a long string of characters. At the bottom, there is a green 'BAKE!' button with a wand icon and an 'Auto Bake' checkbox. A red arrow points from the text above to the 'Output' field, and a red circle highlights the wand icon next to it.

Once clicked it should look something like this:

“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”

The fact that we see not only Gunzip but “URL decode and “From Hexdump” too is that it will decode everything to “Hex” is the default format to display binary data.

## Step 6

We will then need to press the “wand” once more, this will be the easier alternative to search, drag and drop in the recipe section for “From Hex”. (see the wand once again below):

This will result in the following result in the “Output box”:

“VVdBe3RIM19lTW9PNV93MUxMX24zVjNyX3NUb1BfZDAxbkdfdEgzNWVfZFZtQI9jMUJyX2NIM2VGX2NoNExsU30=”

## Step 7

Seeing that the final result in step 6 we can conclude that it ends with “oNExsU30=”. This will mean that it is yet another “Base64” text.

Resulting in the following see below:

## Result:\*\*

UWA{tH3\_eMoO5\_w1LL\_n3V3r\_sToP\_d01nG\_tH35e\_dVmB\_c1Br\_cH3eF\_ch4LIS}

## Flag Found:

UWA{tH3\_eMo05\_w1LL\_n3V3r\_sToP\_d01nG\_tH35e\_dVmB\_c1Br\_ch3eF\_ch4L1S}

# Emu Casino

## Step 1

It is important to identify what the given hits tell you, as stated by ChatGPT and Google(Palisade) “A seed in a random number generator serves as the initial input that determines the starting state of the algorithm.”

Looking at the given code “flip\_coin.py” we can see that the seed is based on two given aspects, the “round” and the “session id”.

Therefore the outcome of the coin flip will be based on the “round” and “session id”.

## Step 2

We will then need to collect this information, in order to collect the session id we will need to get the cookie of the website. We do this by going to the given website on the question “<http://34.87.251.234:3000>”

## Step 3

We will then need to right click on the website, Click on the “inspect” button (see attached image bellow):

# Welcome to Emu Casino

Fact: 99% of Emus quit just before they hit it big



Current Round: 1  
Your balance: 10 E-Bucks

Reset

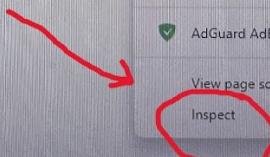
Place your bet

Heads Tails

Enter bet amount

Place Bet

Back	Alt+Left arrow
Forward	Alt+Right Arrow
Reload	Ctrl+R
Save as...	Ctrl+S
Print...	Ctrl+P
Cast...	
Search images with Google	
Send to your devices	
Create QR code for this page	
Translate to English	
AdGuard AdBlocker	
View page source	Ctrl+U
Inspect	



This will result in something like this afterwards.(see attachment below):

# Welcome to Emu Casino

Fact: 99% of Emus quit just before they hit it big



Current Round: 1

Your balance: 10 E-Bucks

Reset

Place your bet

Heads Tails

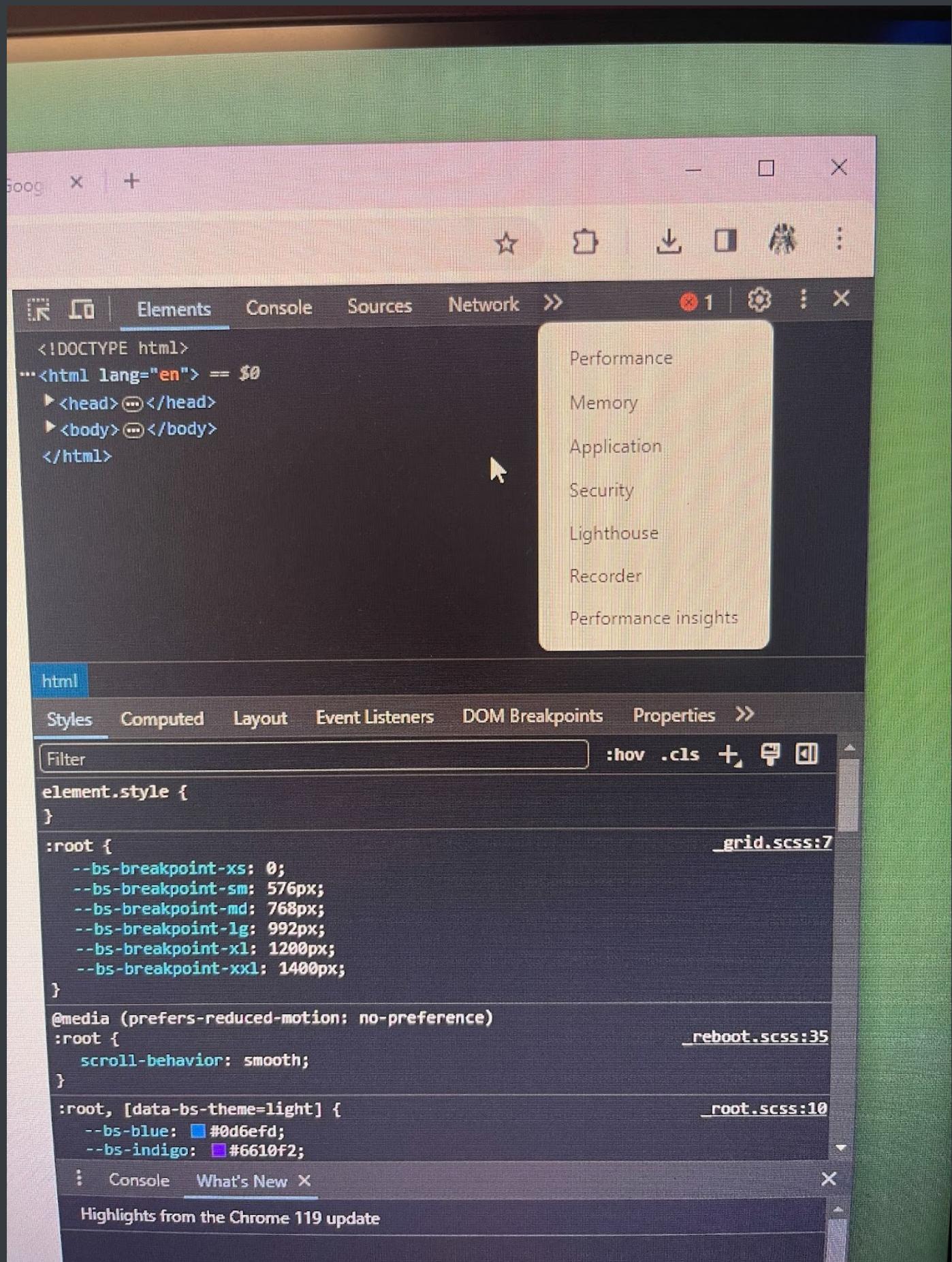
Enter bet amount

Place Bet

The screenshot shows the Chrome DevTools Elements tab open. The page content is visible on the left, and the DevTools interface is on the right. A red arrow points from the 'Inspect' item in the browser's context menu to the 'Inspect' button in the DevTools toolbar. The DevTools interface includes the Element tree, Styles panel, Computed panel, and Layout panel.

## **Step 4**

We will then need to click on the arrow keys next to the word “network” on the top. This will result in something like this afterwards.(see attachment below):



Step 5

We will proceed to click on the “Application” button. This will result in something like this afterwards.(see attachment):

The screenshot shows the Chrome DevTools interface with the "Application" tab selected. On the left, there's a sidebar with sections for "Manifest", "Service workers", and "Storage". Under "Storage", "Cookies" is expanded, showing a list of cookies. One cookie, "http://34.87.251.234:30", is selected and highlighted with a blue border. To the right of the sidebar is a table titled "Application" with columns for Name, Value, and various status indicators (D, P, E, S, H, S, S, P, M). The table contains one row for the selected cookie, showing its name and value. A tooltip "Select a cookie to preview its value" points to the selected cookie in the list. At the bottom of the sidebar, there are sections for "Background services" and "API". Below the sidebar, there are tabs for "Console" and "What's New", and a message about the "Highlights from the Chrome 119 update". On the far right, there's a small video player window showing a blue grid with a white circle and a red play button.

Name	Value	D.	P.	E.	S..	H.	S..	S..	P.	M..
session	eyJcmVkaXRz...	3...	/	S...	1...	✓				

Select a cookie to preview its value

ace Bet

Improved @property section in Elements > Styles

You can now edit the @property rule in Elements > Styles.

## Step 6

Click on this button, this will provide you with the “value” of the website.

The screenshot shows the Chrome DevTools Application tab. On the left, there's a sidebar with sections for Application (Manifest, Service workers, Storage), Storage (Local storage, Session storage, IndexedDB, Web SQL), and Cookies. The Cookies section is expanded, showing entries for http://34.87.251.234:30, Private state tokens, Interest groups, Shared storage, and Cache storage. The entry for http://34.87.251.234:30 is highlighted with a red oval, and a red arrow points to it from the left.

Name	Value	D.	P..	E.	S..	H.	S..	S..	P..	P..
session	eyJcmVkaXRz...	3...	/	S...	1...	✓				M..

This will result in something like this afterwards.(see attachment below):

The screenshot shows a Casino game interface on the left and the Chrome DevTools Application tab on the right. The game interface displays "Current Round: 1" and "Your balance: 10 E-Bucks". A red "Reset" button is visible. The DevTools Application tab shows the same cookie entry as in the previous screenshot, with the value "eyJcmVkaXRz..." highlighted.

Name	Value	D.	P..	E.	S..	H.	S..	S..	P..	P..
session	eyJcmVkaXRz...	3...	/	S...	1...	✓				M..

## Step 7

We will then need to click on the content under the “value” button (see attached image):

The screenshot shows a browser's developer tools with the 'Application' tab selected. Under 'Storage', the 'Session storage' section is expanded. A single cookie entry is visible, labeled 'session' with a value of 'eyJcmVkaXRzIjoxMCwicm91bmQiOjEslNlc3Npb25faWQiOizZTRjYWI2NDUyMDI5MGVhZWU4NDIwM2IwMjgyNTQ4ZCJ9.ZkjrlQ.VjQ5DkGJK28mF032j0laFFb4dLc'. This value is circled in red, and a red arrow points from the text above to this circled area.

This will result in something like this afterwards. This will give the result with the cookie value of:

“eyJcmVkaXRzIjoxMCwicm91bmQiOjEslNlc3Npb25faWQiOizZTRjYWI2NDUyMDI5MGVhZWU4NDIwM2IwMjgyNTQ4ZCJ9.ZkjrlQ.VjQ5DkGJK28mF032j0laFFb4dLc”

## Step 8

To use this cookie value we must decode it, i find that the best tool to do this is the website [“https://www.kirsle.net/wizards/flask-session.cgi”](https://www.kirsle.net/wizards/flask-session.cgi)

You just simply input the cookie value in the “session cookie” input space and click the “decode cookie” button. (see attachment below):

### Flask Session Cookie Decoder

This is a simple Python script to decode [Flask](#) session cookies. Flask, by default, uses the URL-safe signed serializer "[itsdangerous](#)" to encode its *client-side* session cookies. A Flask app uses a secret key to sign the session cookie so that the client can't modify it.

The session cookie itself looks like a bunch of random gibberish, but it's actually really trivial to decrypt. See for yourself by looking at [the source code of this script](#). The decoded session cookie will show you its raw contents and it includes the signature for those contents (so you can look but you can't touch, at least not unless you know the secret key that the Flask app used to sign that cookie!)

Decode a Flask Session Cookie

You can paste in your Flask session cookie here to decode it. Don't know how to get your cookie? See [getting\\_your\\_cookie](#) for tips.

**Session Cookie:**

Paste your session cookie here.

#### Getting your Session Cookie

- **Mozilla Firefox:**
  1. Right-click a web page and click "View Page Info"
  2. Click on the "Security" tab
  3. Click "View Cookies"
  4. Find the session cookie (it's usually named "session", but not always), and copy/paste its "Content" value into the form above.
- **Google Chrome:**
  1. Right-click a web page and click "View Page Info"
  2. Click on "Show cookies and site data"
  3. Browse for the session cookie (it's usually named "session", but not always), and copy/paste its "Content" value into the form above.

#### Source Code

This will result in something like this afterwards.(see attachment below):

## Flask Session Cookie Decoder

Here is the decoded Flask session cookie:

```
{  
    "credits": 10,  
    "round": 1,  
    "session_id": "3e4cab64520290eaee84203b0282548d"  
}
```

This is a simple Python script to decode [Flask](#) session cookies. Flask, by default, uses the URL-safe signed serializer "[itsdangerous](#)" to encode its *client-side* session cookies. A Flask app uses a secret key to sign the session cookie so that the client can't modify it.

The session cookie itself looks like a bunch of random gibberish, but it's actually really trivial to decrypt. See for yourself by looking at [the source code of this script](#). The decoded session cookie will show you its raw contents and it includes the signature for those contents (so you can look but you can't touch, at least not unless you know the secret key that the Flask app used to sign that cookie!).

Decode a Flask Session Cookie

You can paste in your Flask session cookie here to decode it. Don't know how to get your cookie? See [getting\\_your\\_cookie](#) for tips.

Session Cookie:

```
eyJjcmV0aXRzIjoxMjwicm91bmQjOjEsInNlc3Npb25faWQjOiiIzZTRjYWl2ND  
UyID15IGVhZW4NDiwI21wMjgyNTQ4ZC39.ZkjriQ.VjQ5DkGJK28mF032j0la  
FFb4dLc
```

## Step 9

With the given session id of “3e4cab64520290eaee84203b0282548d” from step 9 we can now predict the outcome of the coinflip.

We do this by adding the session id into the (“”) section within the “solution\_template” That was given in the question next to the “session\_id” (see attached image):

```
1  from random import choice, seed  
2  
3  
4  def flip_coin():  
5      # Change this line  
6      session_id = ""  
7      # Change this line  
8      round = 0  
9  
10     seed(str(round) + "_" + session_id)  
11  
12     print(choice(["tails", "heads"]))  
13  
14  
15 flip_coin()  
16
```

This will result in something like this afterwards.(see attachment below):

```
1  from random import choice, seed  
2  
3  
4  def flip_coin():  
5      # Change this line  
6      session_id = "3e4cab64520290eaee84203b0282548d"  
7      # Change this line  
8      round = 0  
9  
10     seed(str(round) + "_" + session_id)  
11  
12     print(choice(["tails", "heads"]))  
13  
14  
15 flip_coin()  
16 |
```

## Step 10

Now that we have the session id, we need the round number, this will depend on what round you are in the coin flip website.

For example if you are in round 1 the the code will looks something like this (see attached image):

```
1  from random import choice, seed
2
3
4  def flip_coin():
5      # Change this line
6      session_id = "3e4cab64520290eaee84203b0282548d"
7      # Change this line
8      round = 1
9
10     seed(str(round) + "_" + session_id)
11
12     print(choice(["tails", "heads"]))
13
14
15 flip_coin()
16
```

This will result in an answer of either head. Keep in mind It could be either head or tails, it depends on the round. (see attachment below):

```
1  from random import choice, seed
2
3
4  def flip_coin():
5      # Change this line
6      session_id = "3e4cab64520290eaee84203b0282548d"
7      # Change this line
8      round = 1
9
10     seed(str(round) + "_" + session_id)
11
12     print(choice(["tails", "heads"]))
13
14
15 flip_coin()
16
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\BINH LE> & "C:/Users/BINH LE/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/BINH LE/Desktop/solution\_template (1).py"
heads
PS C:\Users\BINH LE>

## Step 11

We will do this until we reach 10 thousand “E-Bucks” on the

Website. This will result in the given flag:

**Result:**

UWA{R0LL111Llli1iNg\_1N\_C4\$\$\$\$\$h!11!}

**Flag:**

UWA{R0LL111Llli1iNg\_1N\_C4\$\$\$\$\$h!11!}

**EWT**

**Step 1**

A clear, and detailed description.

**Step 2**

**Step X**

Flag Found

UWA{xxxxxxxxxx}

## Part 2 – Forensics

### Caffeinated Emus – Part 1

**Step 1**

In order to acquire the flag we must need to understand the question. To find the flag we must find a way to access the metadata within the given “.png” image. We can do this through many ways, i found that the best tool would be through a website called “<https://jimpl.com/results/2484d3q4wc6ankWtKnV6GmrJ?target=exif>”.

## Step 2

Entering the given website of

[“https://jimpl.com/results/2484d3q4wc6ankWtKnV6GmrJ?target=exif”](https://jimpl.com/results/2484d3q4wc6ankWtKnV6GmrJ?target=exif)

We will be greeted with something that looks like this (see attached image):

The screenshot shows the homepage of the Jimpl Online EXIF data viewer. At the top left is the Jimpl logo. The main title is "Online EXIF data viewer". Below it is a subtitle: "Uncover hidden metadata from your photos. Find when and where the picture was taken. Remove EXIF data from the image to protect your personal info.". A rating section shows "★★★★★ 4.1 Based on 60524 votes". Below that is a button to "To leave a vote, upload an image". To the right is a large dashed box containing a cloud icon and the text "Drag and drop an image here or click to upload". Below this box is the text "Up to 50 MB. Your uploads are private. We'll delete all files after 24 hours." Further down are links for "or upload from URL" and a text input field with the placeholder "Example: https://jimpl.com/og-image.png". At the bottom, there's a section titled "How to view EXIF metadata from your photos" with three numbered steps: 1. "Upload a photo" (description: "Upload any photo to Jimpl EXIF Viewer. We'll read its metadata and show what's inside."), 2. "View the metadata" (description: "View EXIF metadata recorded in the photo. Date, time, camera settings, geolocation coordinates, and many more."), and 3. "Remove the metadata" (description: "Download the photo with EXIF data removed. It's good to protect your privacy and reduce the photo file size.").

## Step 3

We will then need to input the given PDF file of “loganpal\_emos.png” in the “drag and drop” box. This will result in something like this afterwards.(see attachment below):

 Jimpl

[IMAGE METADATA](#) [LOCATION](#) [FULL METADATA](#)

[UPLOAD ANOTHER IMAGE](#)



**Image metadata**

Name	loganpaul-emos (1).png
File size	861 KB (881681 bytes)
File type	PNG
MIME type	image/png
Image size	862 x 575 (0.496 megapixels)

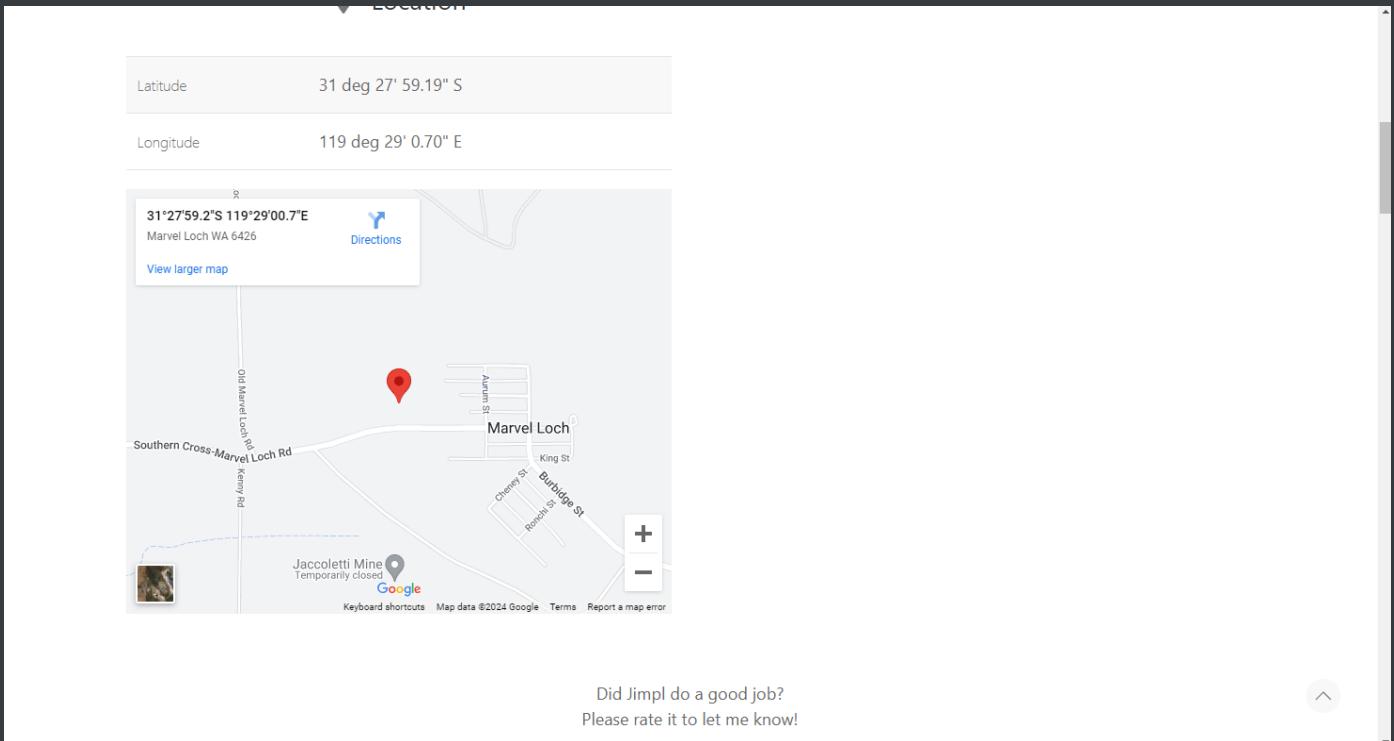
Metadata takes **656 Bytes (0.1%)** of this image and **includes location data**. To protect your privacy, download this image without metadata by clicking the button below.

[REMOVE METADATA](#)

[Location](#)

## Step 4

If we just scroll down in the website from step 3, we will see something like this (see attached image):



Latitude: 31 deg 27' 59.19" S

Longitude: 119 deg 29' 0.70" E

31°27'59.2"S 119°29'00.7"E  
Marvel Loch WA 6426

Directions

View larger map

Old Marvel Loch Rd

Southern Cross-Marvel Loch Rd

Jaccoletti Mine Temporarily closed

Auburn St

King St

Cheney St

Burbridge St

Did Jimpl do a good job?  
Please rate it to let me know!

In the given box of the map on the left we can easily identify that there is a town located near the given pinpoint, that being “Marvel Loch”.

In the original question, it was stated that the flag would be “UWA{town name}”. So gathering what was stated before the final flag for the answer would be “UWA{Marvel Loch}.

Resulting in:

**Result:\*\***

UWA{Marvel Loch}

**Flag Found:**

UWA{Marvel Loch}

## Flightless Data

### Step 1

we will need to first download the html and open it on a tab, this wil allow us to download the image and save it,I will be naming mine "password.jpeg" in downloads.

we will no go into Kali Linux and put in the following which will put us in access to the dowloaded files:

```
cd Downloads
```

### Step 2

we will need to then input the following, this will allow us to access the access the message. this is mentioned in the hint within the question:

```
steghide extract -sf password.jpeg
```

It will ask you for a passphrase, this is located at the bottom of the image of the secret message that being:

```
theEMusWILLRo0ld3w0oRlD
```

this will result in the following:

```
wrote extracted data to "secret.txt".
```

## Step 3

next we just simply enter:

```
cat secret.txt
```

This will enter the secret message and read whats inside. this will return the following:

```
Hello my fellow Emu.  
container by ru  
Fortunately the hoomans arent big brain like use and would not look for  
this secret message in the least significant bits of an image!
```

```
UNAIfLigHtL355_d4Ta_uNd3r_tH3_r4dAT]
```

## Flag Found

```
UWA{fLigHtL355_d4Ta_uNd3r_tH3_r4dAT}
```

# Ruffled Feathers

## **Step 1**

we need first to download ghex within Kali Linux as said in the given hint. we do this by entering the following:

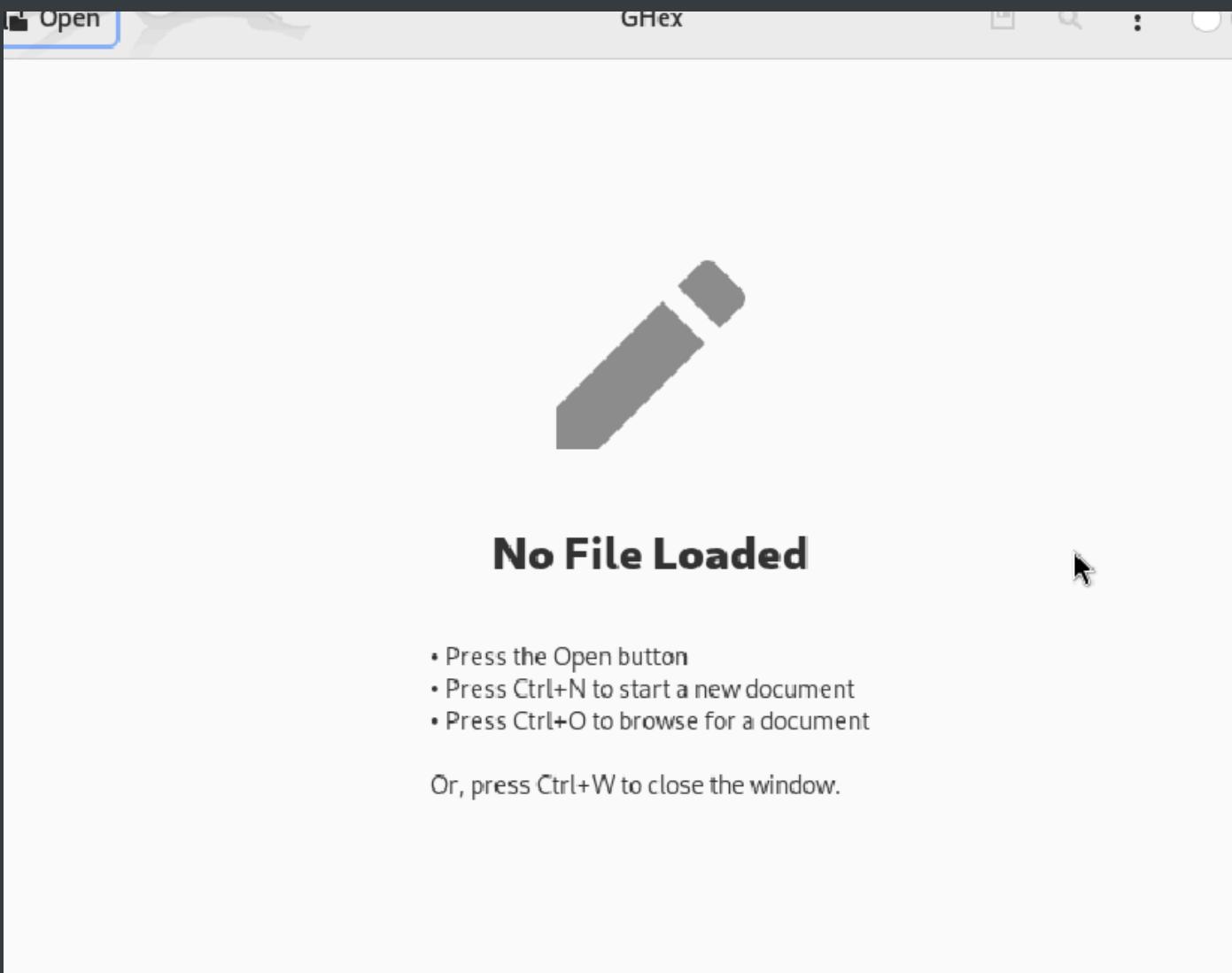
```
sudo apt install ghex
```

We will need to wait for everything to download, if an error comes up simply enter the following and re-enter the above:

```
sudo apt update
```

## **Step 2**

once that is done simply enter "ghex" into the terminal, this will display a display like this:



### Step 3

we will then need to download the given pdf file and click and drag it into this box. this will give us something like this:

00000000	25 50 44 46 2D 31 2E 35 0A 25 D0 D4 C5 D8 0A 36	%PDF-1.5.%.....6
00000010	20 30 20 6F 62 6A 0A 3C 3C 0A 2F 43 6F 72 72 75	0 obj.<<./Corru
00000020	70 74 32 37 32 20 20 20 20 20 20 0A 2F 46 69	pt272 ./Fi
00000030	6C 74 65 72 20 2F 46 6C 61 74 65 44 65 63 6F 64	lter /FlateDecod
00000040	65 0A 3E 3E 0A 73 74 72 65 61 6D 0A 78 DA 5D 51	e.>>.stream.x.]Q
00000050	B1 52 C3 30 0C DD FD 15 62 73 86 38 B6 15 3B CE	.R.0....bs.8..;..
00000060	CA 95 72 65 E3 EA 8D 63 08 89 21 5C 48 5C 92 16	..re...c...!\\H\\..
00000070	7E 1F 27 6E 4A C1 1E A4 A7 27 E9 C9 32 87 37 E0	~.'nJ....'..2.7.
00000080	70 4F F8 3F 7B 6B 49 B6 45 05 22 67 98 6B 09 F6	p0.?{kI.E."g.k..
00000090	15 04 22 2B B4 01 AD 0A 26 03 65 1B 78 A2 FB D3	.."+"....&.e.x...
000000A0	21 49 51 50 37 06 53 28 BA 59 B0 BC 60 9B 94 39	!IQP7.S(.Y...`...9
000000B0	F5 87 08 F6 AE 1E DD 31 79 B6 0F D9 56 96 50 B2	.....1y....V.P.
000000C0	52 4B 3D B7 4E 39 43 65 20 95 82 19 19 1B DB 46	RK=.N9Ce .....
000000D0	85 22 44 DA FA 24 95 86 FA BE 1A A6 18 99 5A 7F	."D...\$.....Z.
000000E0	FA 68 A2 3F B8 AF 24 B0 8B 1C E6 B4 1B FC 0C BF	.h.?...\$.....
000000F0	23 B9 8B C1 AA 8F B0 3A 97 BF D7 73 4E 17 C9 A9	#.....:....sN...
00000100	AB 8E EE 25 48 14 D4 57 63 E3 C6 9B 79 3E 72 67	...%H..Wc....y>rg
00000110	89 08 8B E0 20 2E EF 56 5C 31 A5 0A A8 7B F2 49	.... ..V\1...{.I
00000120	04 E3 E1 60 4C B9 F2 23 29 25 33 26 72 BF 6E A0	...`L..#)%3&r.n.
00000130	B2 5D 2F 60 E3 C9 63 B8 EB A6 57 99 74 D5 49 AF	.]/`...c...W.t.I.
00000140	84 96 6F F8 B3 2A E4 C8 42 26 98 92 A1 3E 7F 82	..o...*..B&...>..
00000150	58 46 5E 1B 86 D1 7F 00 34 A3 6A 10 0A 65 6E 64	XF^....4.j...end
00000160	73 74 72 65 61 6D 0A 65 6E 64 6F 62 6A 0A 34 20	stream.endobj.4
00000170	30 20 6F 62 6A 0A 3C 3C 0A 2F 54 79 70 65 20 2F	0 obj.<<./Type /

we will only be focusing on the right section of it, comparing the corrupted file to a normal one we can see that in the corrupted we can see the following:

```
%PDF-1.5.%.....6
0 obj.<<./Corru
pt272 ./Fi
lter /FlateDecod
```

## Step 4

caomparing it to a normal PDF we can see that instead of "Corrupt272" it has "Length 5 0 R" (see attachment):

```
%PDF-1.3.%..... .
....4 0 obj.<<
/Length 5 0 R /
Filter /FlateDec
```

## step 5

knowing this to fix the corrupted PDF we must change the "Corrupt272" to "Length 5 0 R", this will fix the PDF completly and will give you the output:

# Super Duper Top Secret

The hoomans should never know I am a sick skateboarder.



**Flag**

UWA{uNrUffLed\_p}

## Emu in the Shell

### Step 1

A clear, and detailed description.

### Step 2

### Step X

Flag Found

UWA{xxxxxxxxxx}

# Part 3 – Linux and Networking

## EMU Hack#1 - Backdoored

### step 1

As said in the question we first must use the "nmap" command to scan. In order to see the message we must create a code that can access a port to access. We will also need to use the -Pn option as well as the given range, which is between 61000-61500 with the server at the end. this is a lot but if we put all of it together it should look like this:

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

this will take a while (around 40 seconds for me). it will spit out a bunch of things but the most important part would be the given port of 61337:

PORT	STATE	SERVICE
61337/tcp	open	unknown

### Step 2

we will then need to send a message through the port that we found using "nc", having trouble i utilised researched on google and came to the following:

```
echo 'EMO' | nc 34.116.68.59 61337
```

this will give us the following as a response:

```
(kali㉿kali)-[~]
$ echo 'EMU' | nc 34.116.68.59 61337
```

The terminal shows a directory tree with folders like Music, Pictures, Videos, Downloads, Device, and Network. Below the tree, a message from the 'EMU' backdoor is displayed:

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\_5c4nN1nG?1!?1}

This looks about right as it is powered by the "angry Emus", just like the one in the question.

## Flag

```
UWA{4dvanC3d_p0r7_5c4nN1nG?1!?1}
```

## EMU Hack #2 - Git Gud

### Step 1

first we must figure out how to get the access to the file using "git". this was quite challenging for me so i used ChatGPT and it came out with this:

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

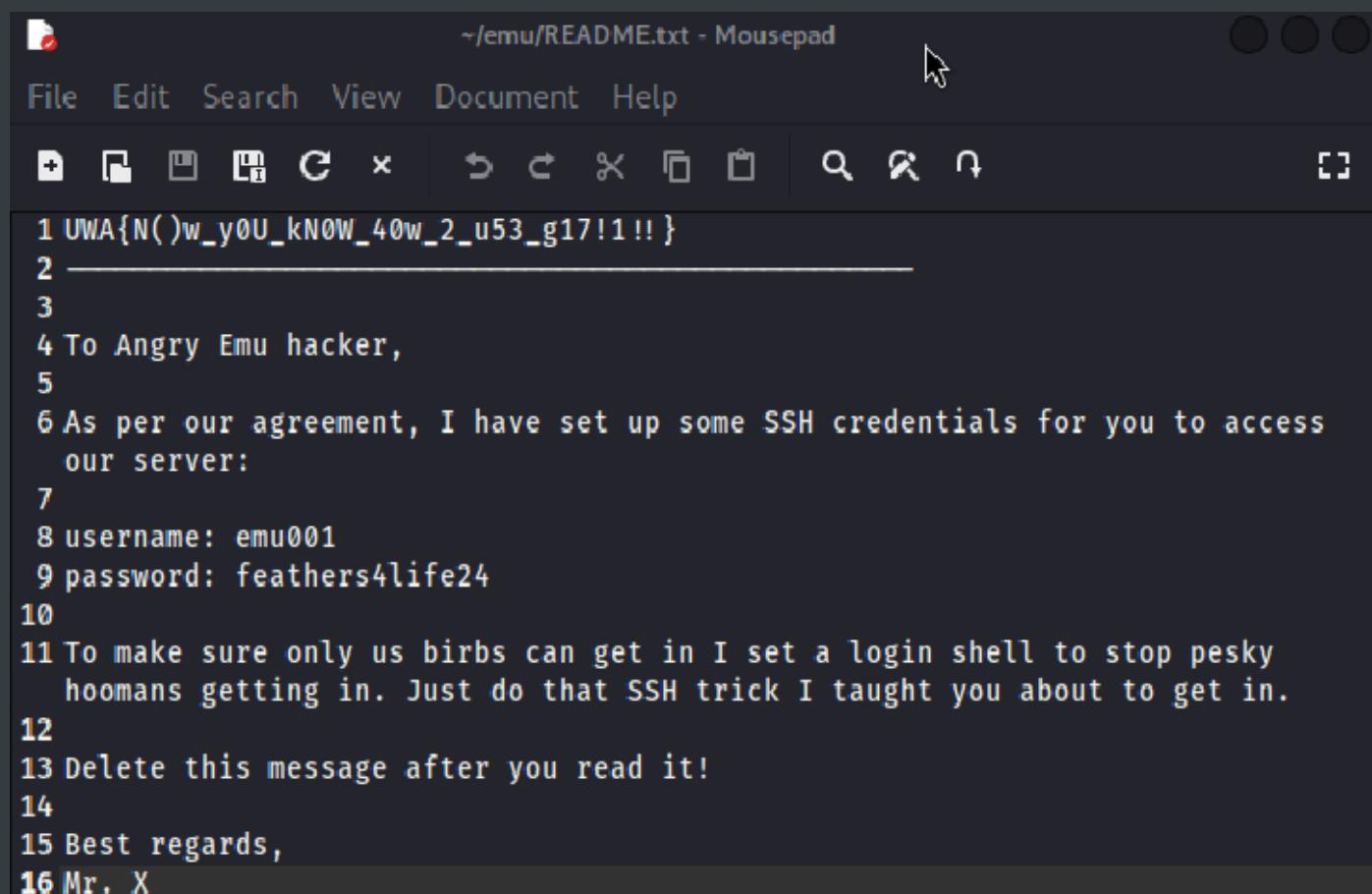
by using "clone" we are trying to clone the already existing Git repository. Once we put this in it will return this:

```
(kali㉿kali)-[~]
└─$ git clone http://34.116.68.59:8000/emu.git
Cloning into 'emu'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

## step 2

Step 1 would have downloaded a file within your device, if you are having trouble finding the file it would be easier if you typed "ls -al", this will provide you with files that you have and when they were downloaded. the name of the dowloaded would be "emu" or "README.txt"

once located open the file and within should be:



The screenshot shows a terminal window with the command "git clone http://34.116.68.59:8000/emu.git" run in it. The output shows the cloning process for a repository named "emu". Below the terminal, a "Mousepad" application window is open, displaying the contents of the "README.txt" file. The file contains the following text:

```
1 UWA{N()w_yOU_kNOW_40w_2_u53_g17!1!!}
2 _____
3
4 To Angry Emu hacker,
5
6 As per our agreement, I have set up some SSH credentials for you to access
our server:
7
8 username: emu001
9 password: feathers4life24
10
11 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 about to get in.
12
13 Delete this message after you read it!
14
15 Best regards,
16 Mr. X
```

## Flag

```
UWA{N()w_y0U_kN0w_40w_2_u53_g17!1!!}
```

## EMU hack #3 - SSH Tricks

### Part 1

as mentioned we will have to do a SSH command, a bash would be the best option as any other command like SSH will cause corruption, so we can't use it, the only solution would be:

```
ssh -h
```

this will give us the following:

```
[kali㉿kali)-[~]# ssh -h
unknown option -- h
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 destination] [-L address] [-l login_name] [-m mac_spec]
           [-O ctl_cmd] [-o option] [-P tag] [-p port] [-Q query_option]
           [-R address] [-S ctl_path] [-W host:port] [-w local_tun[:remote_tu
n]] [-r--r--] kali kali 0 May 19 23:55 .sudo_as_admin_successful
           destination [command [argument .... ]]]sion-errors
           [-t timeout] [-T tag] [-W host:port] [-w local_tun[:remote_tu
n]] [-r--r--] kali kali 303 May 20 14:23 .zsh_history
[kali㉿kali)-[~]#
```

### part 2

Now to create the bash, we will be using the port 2022 and the hosted server to do this, it will look something like:

```
ssh -p 2022 -t emu001@34.116.68.59 'bash -i'
```

Once we input this a passphrase is asked, this will be the one from last question, that being "feathers4life24"

### **step 3**

now we will need to find the name of the given image, just like last question we will enter "ls -al", this will give us all the files, (see attached image):

```
drwxr-xr-x 1 root root 4096 Mar 10 12:03 .
drwxr-xr-x 1 root root 4096 Mar 10 12:02 ..
-rwxr-xr-x 1 root root 6121 Mar 10 12:02 .profile
-rw-r--r-- 1 root root 313 Mar 10 12:02 note_to_angry_emu_hacker.txt
-rw-r--r-- 1 root root 486105 Mar 10 12:02 top_secret.png
emu001@e4b74d8b74d2:~$ 
```

### **step 4**

## Part 4 – Vulnerabilities

### Feathered Forum – Part 1

#### **Step 1**

First we must click on the given hint within the question and identify what is needed to authenticate logins. Looking at the given code it is seen that the authentication relies on the “username cookie” and whether it matches with an existing username or not.

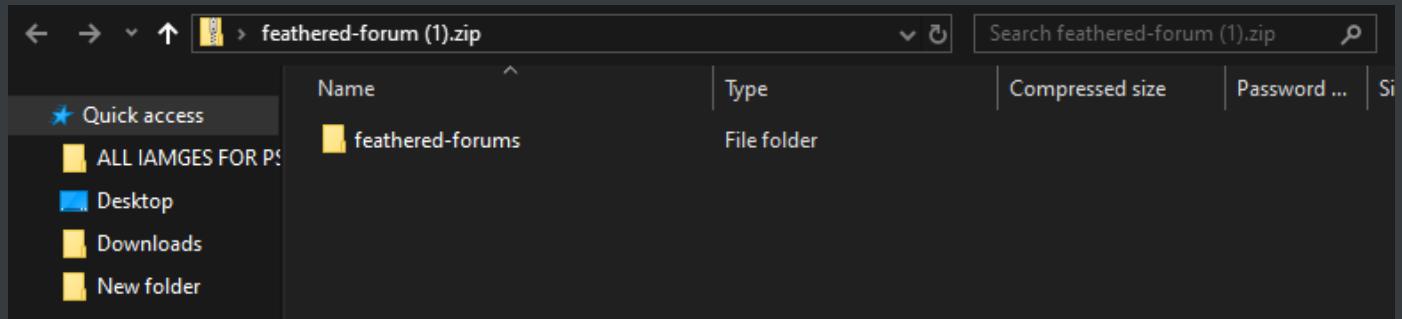
This can be identified here (see attached image):

```
emu_usernames = [account["username"] for account in EMU_USERS_ACCOUNTS]
# Just check that the "username" cookie matches an existing user
if not request.cookies.get('username', None) in emu_usernames:
    return redirect(url_for('index'))
```

#### **Step 2**

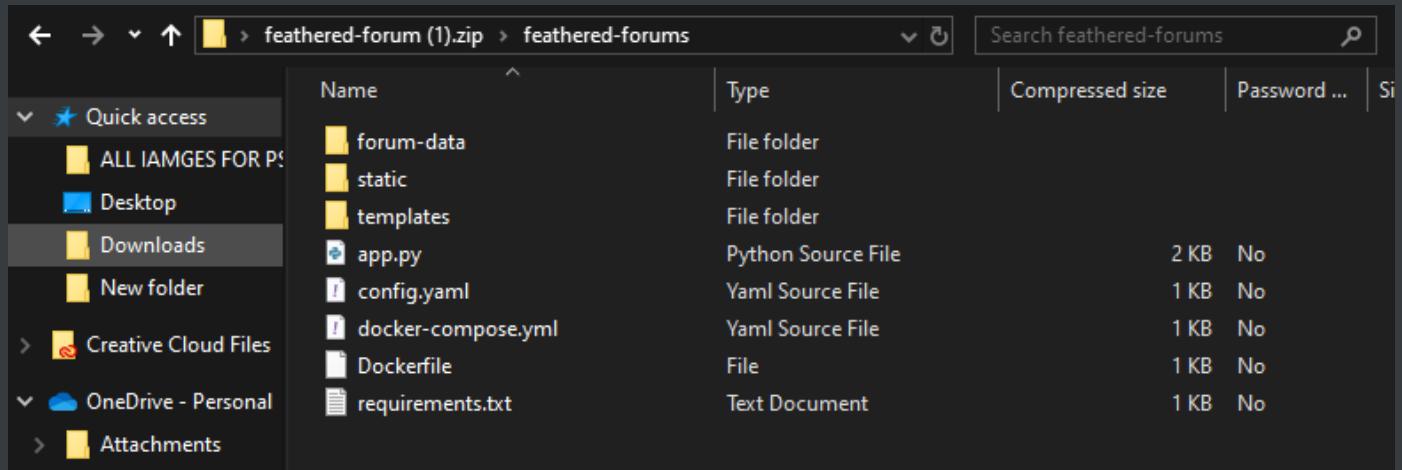
We will then need to figure out how we can access the already existing username as well as make a “username” cookie to match with the existing username.

To gain access to the already existing username we open the given “application code” that was given in the question. Opening it we are greeted with something like this, keep in mind this is a Windows device, but the file will look the same on IOS (see attached image):



### Step 3

We will open the file and will see something like this, keep in mind this is a Windows device, but every file name will look the same on IOS also (see attached image):



### Step 4

We will then proceed to open the “app.py” file with any software that supports python. VScode would be recommended and will be the software I will personally use in the explanation.

(see attached image):

feathered-forum (1).zip > feathered-forums			
	Name	Type	Compressed size
Quick access	forum-data	File folder	
ALL IAMGES FOR P	static	File folder	
Desktop	templates	File folder	
Downloads	app.py	Python Source File	2 KB No
New folder	config.yaml	Yaml Source File	1 KB No
Creative Cloud Files	docker-compose.yml	Yaml Source File	1 KB No
OneDrive - Personal	Dockerfile	File	1 KB No
Attachments	requirements.txt	Text Document	1 KB No

Once that is done you will see something like this (see attached image):

```

1 # Challenge Name: Feathered Forums Parts 1, 2 and 3
2 # Challenge Category: Vulnerabilities
3
4 from flask import Flask, render_template, request, redirect, url_for, send_file
5 from functools import wraps
6 import json
7 import os
8 import yaml
9
10 app = Flask(__name__)
11
12 # Hard code the allowed users so human hackers cannot make their own accounts
13 EMU_USERS_ACCOUNTS = [
14     {
15         "username": "BeakMaster",
16         "password": os.urandom(16).hex()
17     },
18     {
19         "username": "OstrichOutlaw",
20         "password": os.urandom(16).hex()
21     },
22     {
23         "username": "H4ck3r3mu123",
24         "password": os.urandom(16).hex()
25     }
26 ]
27
28 # Load in the secret config.yaml file with the super duper top secret secret
29 # I actually need to use this secret for something
30 with open('config.yaml', 'r') as f:
31     APP_CONFIG = yaml.safe_load(f)
32
33 # Load in posts for the forum (I haven't figured out how to use a database yet)
34 with open('./forum-data/posts.json', 'r') as f:
35     POSTS = json.load(f)['posts']
36     for index, post in enumerate(POSTS):
37         post['id'] = index
38         post['replies_len'] = len(post['replies'])
39
40
41 # I know websites use Cookies to handle authentication
42 # Pretty sure I did this correctly.
43 def auth_required(f):
44     @wraps(f)
45     def decorated_function(*args, **kwargs):
46         emu_usernames = [account['username'] for account in EMU_USERS_ACCOUNTS]
47         Just check that the "username" cookie matches an existing user
48         if not request.cookies.get('username', None) in emu_usernames:
49             return redirect(url_for('index'))

```

## Step 5

Without needing to scroll down we can already see that there is already a display of existing usernames of existing users already. (see attached image):

```

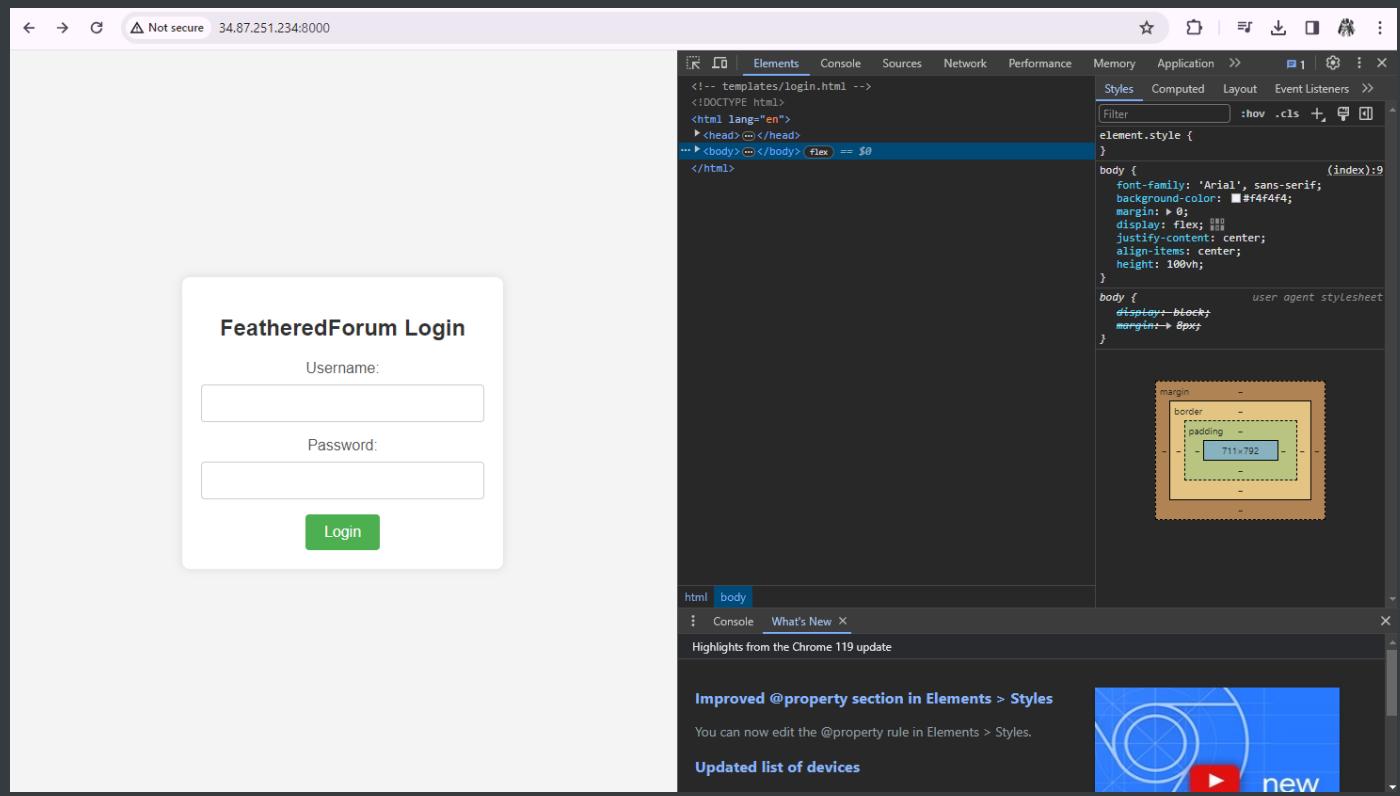
11 # Hard code the allowed users so human hackers cannot make their own accounts
12 EMU_USERS_ACCOUNTS = [
13     {
14         "username": "BeakMaster",
15         "password": os.urandom(16).hex()
16     },
17     {
18         "username": "OstrichOutlaw",
19         "password": os.urandom(16).hex()
20     },
21     {
22         "username": "H4ck3r3mu123",
23         "password": os.urandom(16).hex()
24     },
25 ]
26

```

We will use this to help us to match it with the "username" cookie.

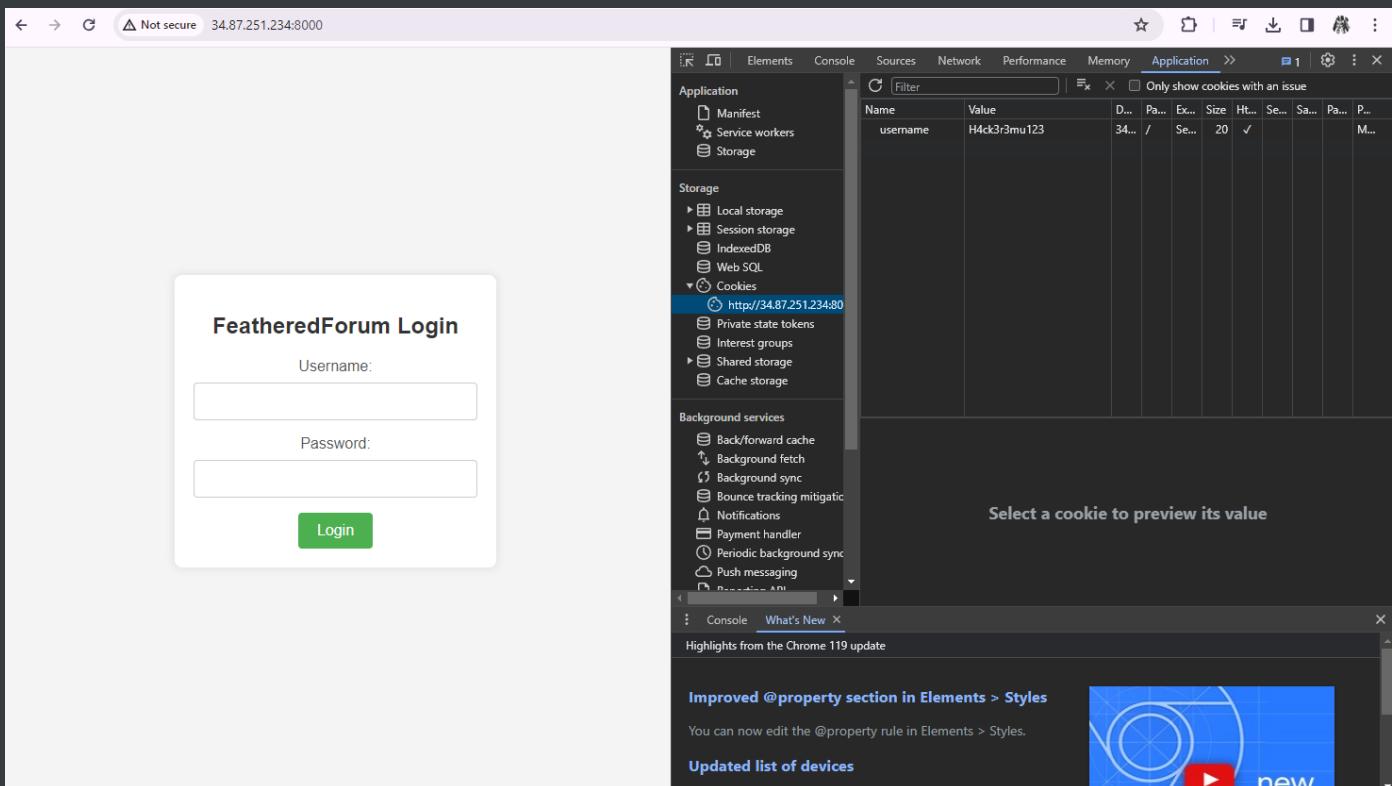
## Step 6

We will then need to go to the website that the question gave “<http://34.87.251.234:8000>”. We will the need to right click on the website once in and select on the “inspect” button, it will look like this (see attachment):



## Step 7

We will then proceed to click the “application” button in the top bar and click on the drop down bar under “cookies”. This will give us something like this (see attachment):



## Step 8

We will then need to replace the value/word within the “name” bar with “username”.

We will also need to replace the value within the “value” bar with any username of an existing user from the python code, i will use “H4ck3r3mu123”.

The final result should look like this:

The screenshot shows the Chrome DevTools interface with the 'Application' tab selected. On the left, the 'Storage' section is expanded, showing 'Cookies'. Under 'Cookies', there is an entry for the URL 'http://34.87.251.234:80'. This entry contains a single cookie named 'username' with the value 'H4ck3r3mu123'. The 'Value' column shows the raw hex value 'H4ck3r3mu123'. Other columns include 'D...', 'Pa...', 'Ex...', 'Size', 'Ht...', 'Se...', 'Sa...', 'Pa...', and 'P...', with some values partially visible.

Then proceed to click enter.

This is us creating a username cookie, by putting an existing username in the “Value” bar we are matching it with an existing one, which is that.

## Step 9

Simply click on the search engines search bar and add “/forum” to the URL. This is mentioned in the original question.

The final URL on the search engine would look like this “<http://34.87.251.234:8000/forum>”

Once in you will see something like this:

## Welcome to BeakMaster's Super Top Secret Forum

UWA{C00k13333z\_4r3\_Th3\_W4y\_T0\_4n\_3mu's\_H34rt}!

You were invited to this forum because you are recognised as one of the most elite Outback hackers. These forums are designed to share important information with each other. So be my guest as I invite you to engage with each other's posts.

**Please note:** Top contributers will receive a hefty reward

Only valuable contributions will be noted...

Btw please ignore the minimalistic web design... I haven't got around to making it very pretty yet.

We are now in the forum for all the juicy stuff.

**Flag:**

UWA{C00k13333z\_4r3\_Th3\_W4y\_T0\_4n\_3mu's\_H34rt}

## Feathered Forum – Part 2

### Step 1

we can identify that the once we are in the forum in part 1, that there are chats below. once we are given the flag we can scroll down to the different chats that are available. (see attachment):

Post	Created By	Total Replies
<a href="#">Welcome to My FeatheredForum!</a>	BeakMaster	1
<a href="#">what about dat top secret cyber op thingy ma bob</a>	OstrichOutlaw	2
<a href="#">Super Duper Advanced Cryptography Hacking</a>	H4ck3r3mu123	1
<a href="#">Securing Networks with AI: A Deep Dive</a>	BeakMaster	1
<a href="#">PLANNING THE FIRST EMU IN SPACE!</a>	OstrichOutlaw	1
<a href="#">You Should Do Emu Yoga</a>	H4ck3r3mu123	1
<a href="#">What on earth is a path traversal???</a>	BeakMaster	4

## Step 2

we will then proceed to click on the last post "What on earth is a path traversal???" this will provide us with the given chat (see attachment):

**What on earth is a path traversal???**

*From: BeakMaster*  
Uuuuhhhh someone sent me this email saying this site has a path traversal vulnerability??? What on earth is that??? Y is this person sending me this CWE-22 thing?



---

*From: OstrichOutlaw*  
I asked ChatGPT and it said this, "Path traversal is a web security vulnerability where attackers exploit insufficient input validation to navigate to files outside of the web server's intended directory, potentially accessing sensitive information or executing arbitrary code. Prevention involves robust input validation and restricting file access to authorized directories."

---

*From: H4ck3r3mu123*  
So you can just read other files??? How is that bad???

---

*From: BeakMaster*  
Yeah idk, I am just going to respond to that email and say it is a feature and ignore it.

---

[Back to other posts](#)

## Step 3

from this, if we read we can easily identify that the CWE number is "22", therefore the flag would be UWA{CWE-22}

### Flag Found

UWA{CWE-22}

## Feathered Forum – Part 3

### Step 1

A clear, and detailed description.

**Step 2**

**Step X**

Flag Found

UWA{xxxxxxxxxx}

## Emu Apothecary

**Step 1**

A clear, and detailed description.

**Step 2**

**Step X**

Flag Found

UWA{xxxxxxxxxx}