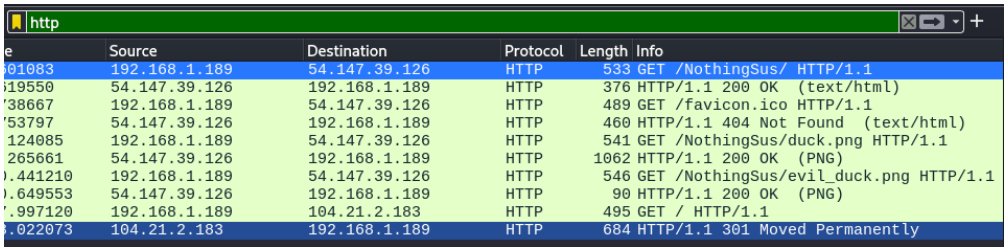
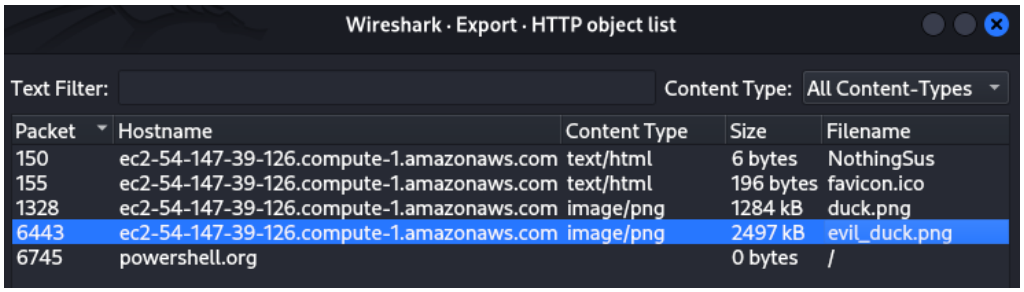
Forensics: Very very very hidden

Student Name: John Bless Santos

1. Download the try_me.pcap
2. Hints given:
    a. I believe you found something, but are there any subtle hints as random queries?
    b. The flag will only be found once you reverse the hidden message
    c. Finding the flag will take many steps
        i. A step may be a hidden map to find the hidden treasure
3. Analysis:
    a. We must analyze network traffic. There may be packets out there that may be distinct to us but we will need to analyze other packets to connect both.
    b. As for the flag that will be found when we reverse the hidden message, we may need to use some sort of script or decryption.
4. We apply common protocol filters like http, dns, ftp, telnet, ssh, icmp and tcp.
    a. We found this suspicious http traffic.
        i. 

e	Source	Destination	Protocol	Length	Info
01083	192.168.1.189	54.147.39.126	HTTP	533	GET /NothingSus/ HTTP/1.1
19550	54.147.39.126	192.168.1.189	HTTP	376	HTTP/1.1 200 OK (text/html)
38667	192.168.1.189	54.147.39.126	HTTP	489	GET /favicon.ico HTTP/1.1
53797	54.147.39.126	192.168.1.189	HTTP	460	HTTP/1.1 404 Not Found (text/html)
124085	192.168.1.189	54.147.39.126	HTTP	541	GET /NothingSus/duck.png HTTP/1.1
265661	54.147.39.126	192.168.1.189	HTTP	1062	HTTP/1.1 200 OK (PNG)
.441210	192.168.1.189	54.147.39.126	HTTP	546	GET /NothingSus/evil_duck.png HTTP/1.1
.649553	54.147.39.126	192.168.1.189	HTTP	90	HTTP/1.1 200 OK (PNG)
.997120	192.168.1.189	104.21.2.183	HTTP	495	GET / HTTP/1.1
.022073	104.21.2.183	192.168.1.189	HTTP	684	HTTP/1.1 301 Moved Permanently

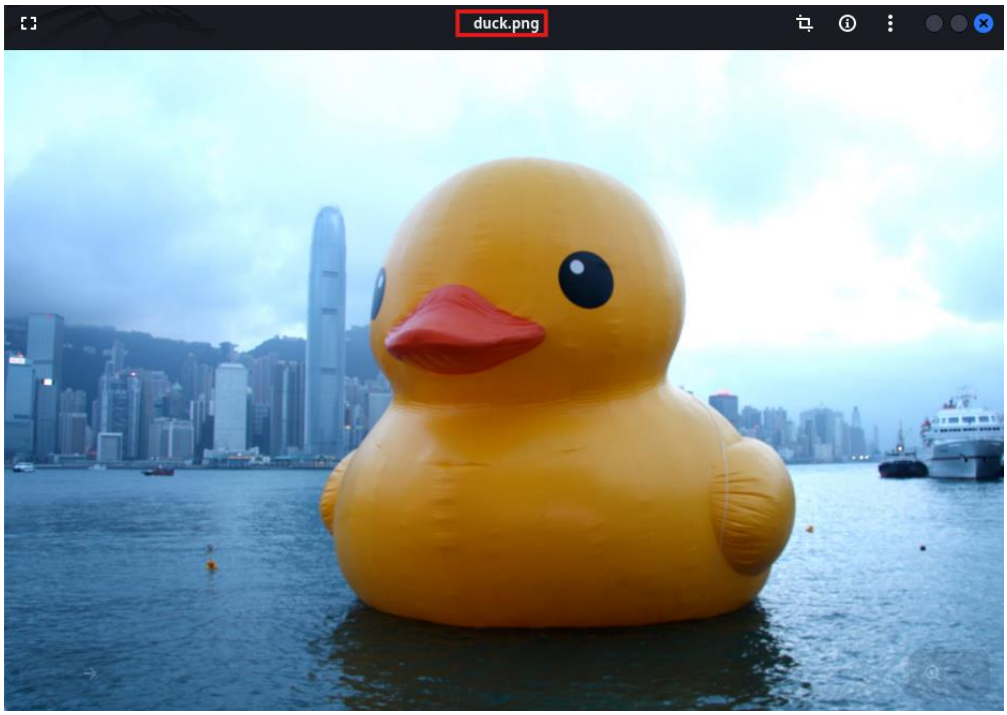
        ii. We can see the GET request /NothingSus/, /NothingSus/duck.png, /NothingSus/evil_duck.png
        iii. We will export the images: File -> Export Objects -> HTTP
        iv. 

Wireshark · Export · HTTP object list

Text Filter: Content Type: All Content-Types

Packet	Hostname	Content Type	Size	Filename
150	ec2-54-147-39-126.compute-1.amazonaws.com	text/html	6 bytes	NothingSus
155	ec2-54-147-39-126.compute-1.amazonaws.com	text/html	196 bytes	favicon.ico
1328	ec2-54-147-39-126.compute-1.amazonaws.com	image/png	1284 kB	duck.png
6443	ec2-54-147-39-126.compute-1.amazonaws.com	image/png	2497 kB	evil_duck.png
6745	powershell.org		0 bytes	/

    b. We opened duck.png and evil_duck.png.

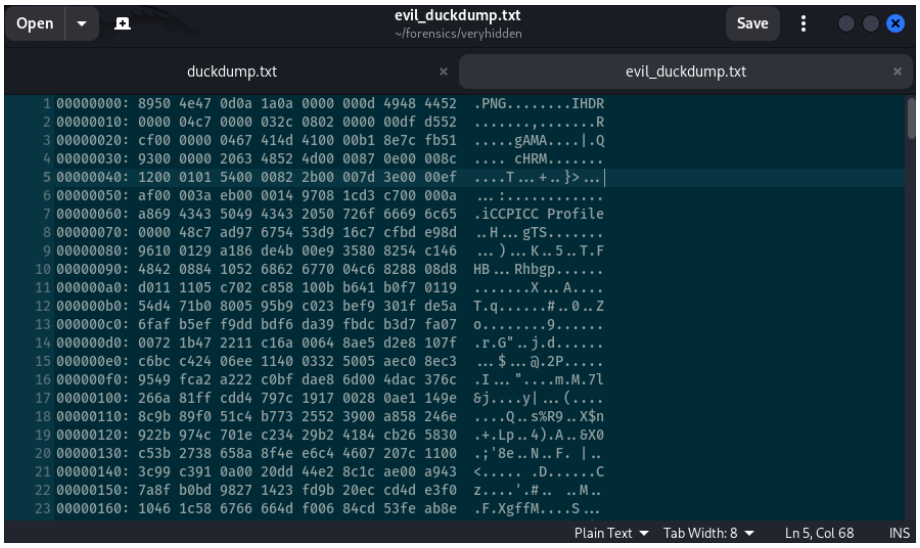
i. 

ii. 

1. Evil_duck.png looks heavily pixelated compared to duck.png. It may be hiding something.

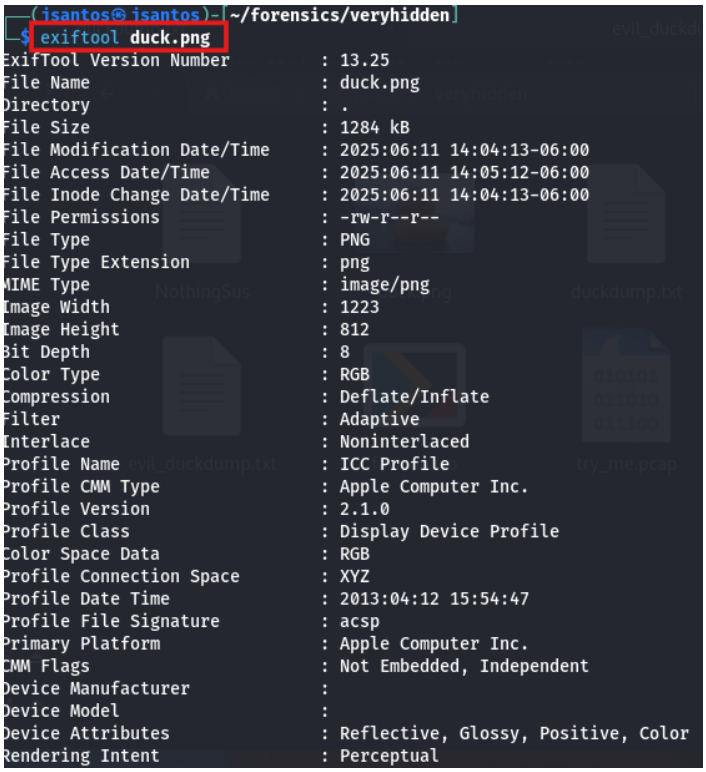
c. Steganography analysis:

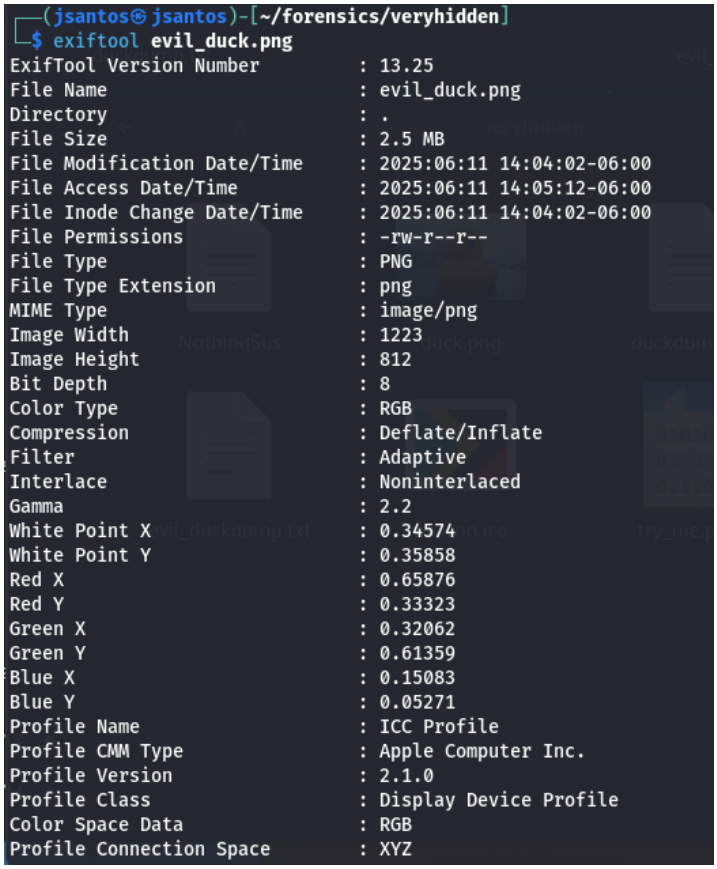
i. We dumped the hex values of both images into a text file and analyzed its contents. We looked for any clues related to things like maps, Pico, http, ctf, flags, and clues, but there were none.

1. 

2. We opened the text files with gedit and searched for keywords using the search function (ctrl + f).

ii. We used exiftool to analyze the metadata of the images. There were no clues found in the metadata.

1. 

2. 

```
┌──(jsantos㉿jsantos)-[~/forensics/veryhidden]
└─$ exiftool evil_duck.png
ExifTool Version Number         : 13.25
File Name                       : evil_duck.png
Directory                       : .
File Size                       : 2.5 MB
File Modification Date/Time     : 2025:06:11 14:04:02-06:00
File Access Date/Time           : 2025:06:11 14:05:12-06:00
File Inode Change Date/Time     : 2025:06:11 14:04:02-06:00
File Permissions                : -rw-r--r--
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 1223
Image Height                    : 812
Bit Depth                       : 8
Color Type                      : RGB
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Gamma                           : 2.2
White Point X                   : 0.34574
White Point Y                   : 0.35858
Red X                           : 0.65876
Red Y                           : 0.33323
Green X                         : 0.32062
Green Y                         : 0.61359
Blue X                          : 0.15083
Blue Y                          : 0.05271
Profile Name                    : ICC Profile
Profile CMM Type                : Apple Computer Inc.
Profile Version                 : 2.1.0
Profile Class                   : Display Device Profile
Color Space Data                : RGB
Profile Connection Space        : XYZ
```

5. We analyzed the Wireshark capture filtered by DNS traffic. The system 192.168.1.189 makes a high number of DNS queries to the DNS server 192.168.1.1 which is likely to be the router.
    a. There were Lots of queries to google.com, gstatic.com, doubleclick.net, microsoft.com, github.com.
        i. These queries, s3.amazonaws.com, elb.amazonaws.com may indicate access to cloud-hosted resources.
        ii. These queries, avatars.githubusercontent.com, user-images.githubusercontent.com may be likely tied to GitHub-hosted images and may link to hidden flags or steganographic images.
    b. We notice this query, raw.githubusercontent.com. This is frequently used to host steganography images, encrypted text or code samples or payloads.
    c. Another noticeable query near the end of the DNS traffic is powershell.org. The author may have used PowerShell to run some sort of script.

i.

192.168.1.1	192.168.1.189	DNS	169	Standard query response 0x2757 A stats.g.doubleclick.net CNA
192.168.1.189	192.168.1.1	DNS	76	Standard query 0xca1e A c1.microsoft.com
192.168.1.189	192.168.1.1	DNS	76	Standard query 0xca1e A c1.microsoft.com
192.168.1.1	192.168.1.189	DNS	160	Standard query response 0xca1e A c1.microsoft.com CNAME c.ms
192.168.1.1	192.168.1.189	DNS	160	Standard query response 0xca1e A c1.microsoft.com CNAME c.ms
192.168.1.189	192.168.1.1	DNS	70	Standard query 0xeb9e A c.bing.com
192.168.1.189	192.168.1.1	DNS	70	Standard query 0xeb9e A c.bing.com
192.168.1.1	192.168.1.189	DNS	172	Standard query response 0xeb9e A c.bing.com CNAME c-bing-com
192.168.1.189	192.168.1.1	DNS	88	Standard query 0x607a A presence.teams.microsoft.com
192.168.1.189	192.168.1.1	DNS	88	Standard query 0x607a A presence.teams.microsoft.com
192.168.1.1	192.168.1.189	DNS	220	Standard query response 0x607a A presence.teams.microsoft.co
192.168.1.189	192.168.1.1	DNS	85	Standard query 0x850f A login.microsoftonline.com
192.168.1.189	192.168.1.1	DNS	85	Standard query 0x850f A login.microsoftonline.com
192.168.1.1	192.168.1.189	DNS	318	Standard query response 0x850f A login.microsoftonline.com C
192.168.1.189	192.168.1.1	DNS	81	Standard query 0x2d1c A wpad.fios-router.home
192.168.1.1	192.168.1.189	DNS	131	Standard query response 0x2d1c No such name A wpad.fios-rout
192.168.1.189	192.168.1.1	DNS	79	Standard query 0x9d65 A teams.microsoft.com
192.168.1.189	192.168.1.1	DNS	79	Standard query 0x9d65 A teams.microsoft.com
192.168.1.1	192.168.1.189	DNS	186	Standard query response 0x9d65 A teams.microsoft.com CNAME t
192.168.1.189	192.168.1.1	DNS	74	Standard query 0x42fb A powershell.org
192.168.1.189	192.168.1.1	DNS	74	Standard query 0x42fb A powershell.org
192.168.1.1	192.168.1.189	DNS	106	Standard query response 0x42fb A powershell.org A 104.21.2.1
192.168.1.189	192.168.1.1	DNS	80	Standard query 0x5ddf A fonts.googleapis.com
192.168.1.1	192.168.1.189	DNS	96	Standard query response 0x5ddf A fonts.googleapis.com A 142.
192.168.1.189	192.168.1.1	DNS	80	Standard query 0xe68a A cdnjs.cloudflare.com
192.168.1.1	192.168.1.189	DNS	112	Standard query response 0xe68a A cdnjs.cloudflare.com A 104.
192.168.1.189	192.168.1.1	DNS	67	Standard query 0xa686 A s.w.org
192.168.1.189	192.168.1.1	DNS	67	Standard query 0xa686 A s.w.org
192.168.1.1	192.168.1.189	DNS	83	Standard query response 0xa686 A s.w.org A 192.0.77.48

6. We researched these DNS queries in relation to steganography, and we got a lead when we searched for “steganography PowerShell”.
7. PowerShell Steganography
   a. https://pcsxcetrasupport3.wordpress.com/2020/07/22/powershell-steganography/

PowerShell Steganography

Posted on July 22, 2020 by pcsxcetrasupport3

Any programming language that can have access to the pixels of a picture file can do a form of byte and pixel modification to hide data within the pixel bytes.

The less of a degree you modify the pixel data the less change that the modified file will be noticed as hiding some form of data.

To me this is more of true steganography than the types that just append an exe to the end of the picture data because it is modifying the the pixel data.

The downside is you have to have some program or script to decode and extract the data which will point directly to the picture file used.

i.

ii. As mentioned earlier, evil_duck.png seems to be pixelated and low quality compared to the other retrieved image, duck.png. Most likely the evil_duck.png is hiding something.

The next question is where did this picture encoding come from ?

It came from here https://github.com/peewpw/Invoke-PSImage we also find a entry in the MITRE | ATT&CK framework here https://attack.mitre.org/software/S0231/

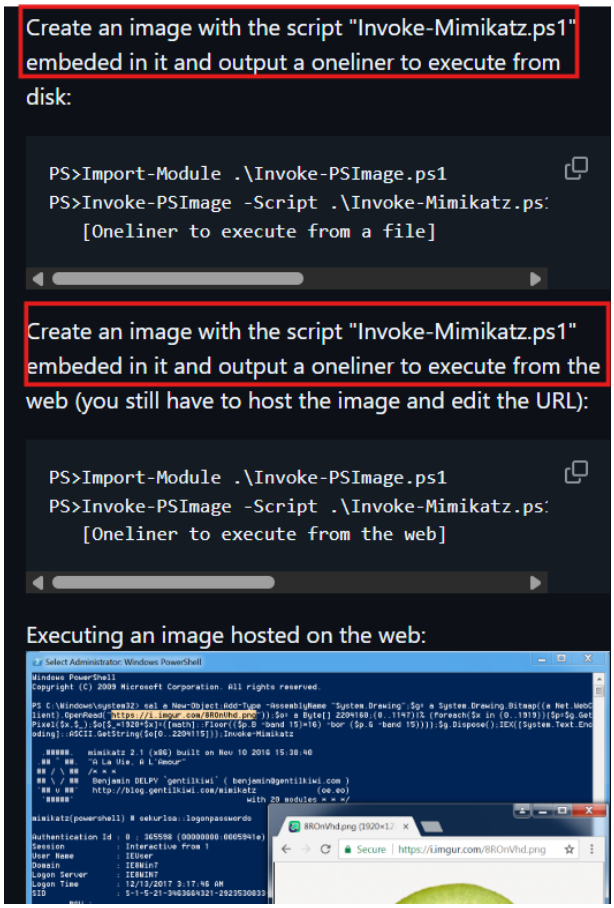
Although the code to decode the picture file remains mostly the same the variables are usually all different including the height and width of the picture file and the variable names for the function calls.

The tool to extract the data can be found on my Github here https://github.com/PCsXcetra/Decode_PS_Stego.

iii.

1. When we analyze this, the picture **encoding** came from the https://github.com/peewpw/Invoke-PSImage/blob/master/Invoke-PSImage.ps1 website.

b. We will explore the PowerShell steganography tool on GitHub. https://github.com/peewpw/Invoke-PSImage/blob/master/Invoke-PSImage.ps1

i. 

ii. Most likely, they used this tool to hide a script within the image. Invoke-PSImage.ps1 embeds a script within the image.

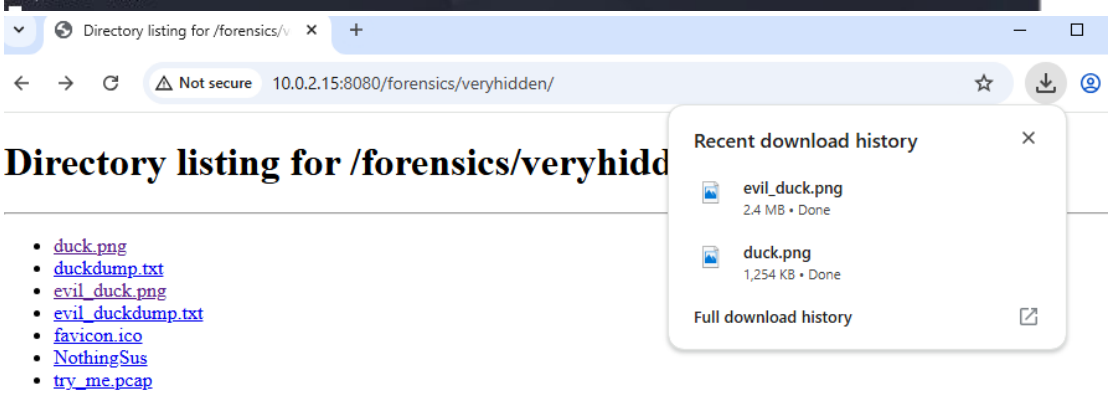
1. It correlates with the hint given, "*The flag will only be found once you reverse the hidden message*".
2. Following the clues given, we would need a tool that would reverse this tool and extract the script.

8. Since we need to be in a Windows OS, we will transfer the photos found from Kali Linux into windows 10. We opened the http server in Kali Linux and let our windows connect to it and grab the required files.

a.

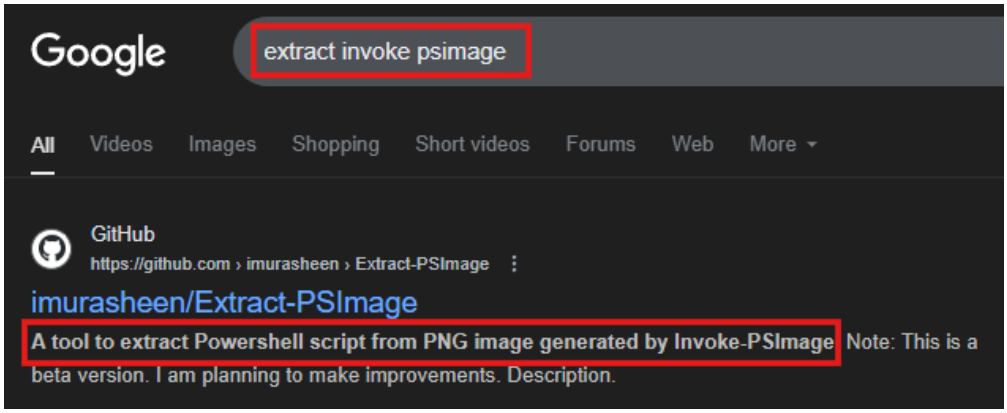
```
┌──(jsantos㉿jsantos)-[~]
└─$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
10.0.2.19 - - [11/Jun/2025 18:37:19] "GET / HTTP/1.1" 200 -
10.0.2.19 - - [11/Jun/2025 18:37:19] code 404, message File not found
10.0.2.19 - - [11/Jun/2025 18:37:19] "GET /favicon.ico HTTP/1.1" 404 -
10.0.2.19 - - [11/Jun/2025 18:37:24] "GET / HTTP/1.1" 200 -
10.0.2.19 - - [11/Jun/2025 18:37:43] "GET / HTTP/1.1" 200 -
10.0.2.19 - - [11/Jun/2025 18:37:57] "GET / HTTP/1.1" 200 -
10.0.2.19 - - [11/Jun/2025 18:38:06] "GET /forensics/ HTTP/1.1" 200 -
10.0.2.19 - - [11/Jun/2025 18:38:23] "GET /forensics/veryhidden/ HTTP/1.1" 200 -
10.0.2.19 - - [11/Jun/2025 18:38:27] "GET /forensics/veryhidden/duck.png HTTP/1.
1" 200 -
10.0.2.19 - - [11/Jun/2025 18:38:59] "GET /forensics/veryhidden/evil_duck.png HT
TP/1.1" 200 -
```

b.



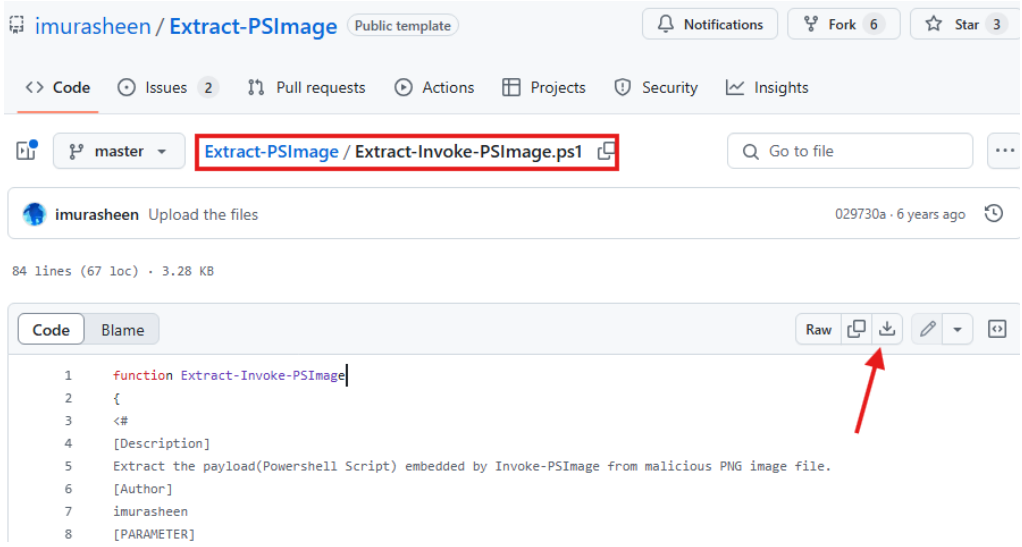
c. We searched for a tool to extract PowerShell script from the image created by PSImage-Invoke.

i.



ii. https://github.com/imurasheen/Extract-PSImage

iii.



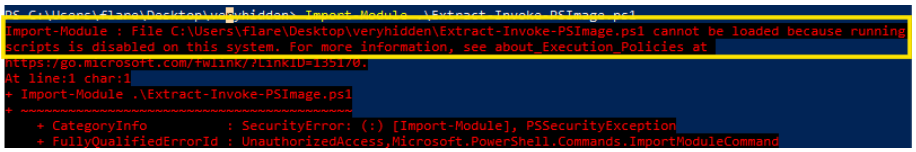
1. We download the script and move it to the folder where the extracted images are located. Make sure Windows Defender is disabled to allow the PowerShell script to be downloaded.

iv.

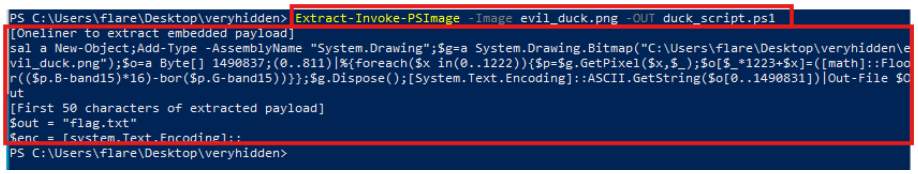
Usage

PS> Import-Module .\Extract-Invoke-PSImage.ps1
PS> Extract-Invoke-PSImage -Image [path to the PNG image] -Out [path to the .ps1 file]
[Oneliner to extract embedded payload]
[First 50 characters of extracted payload]

1. We will follow the execution.



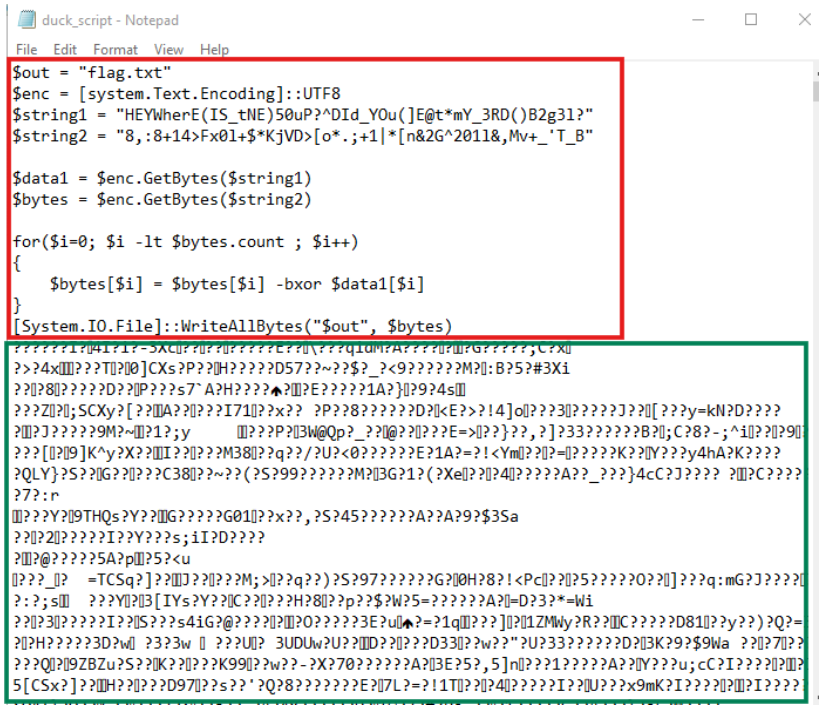
2. Import-Module .\Extract-Invoke-PSImage.ps1
3. We get an error stating that running scripts is disabled on the system. We will need to disable it.
   a. Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
   b. Press “Y” to confirm the change.
4. We ran the script again.



5.

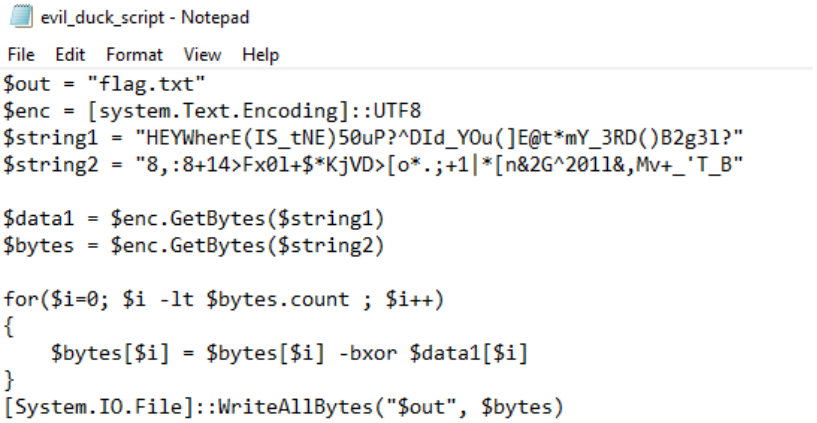
a. Extract-Invoke-PSImage -Image [path to the PNG image] -Out [path to the .ps1 file]
b. Extract-Invoke-PSImage –Image evil_duck.png -OUT duck_script.ps1
    i. Duck_script.ps1 is the PowerShell file which contains the retrieved PowerShell script from the image.

6. We open duck_script.ps1

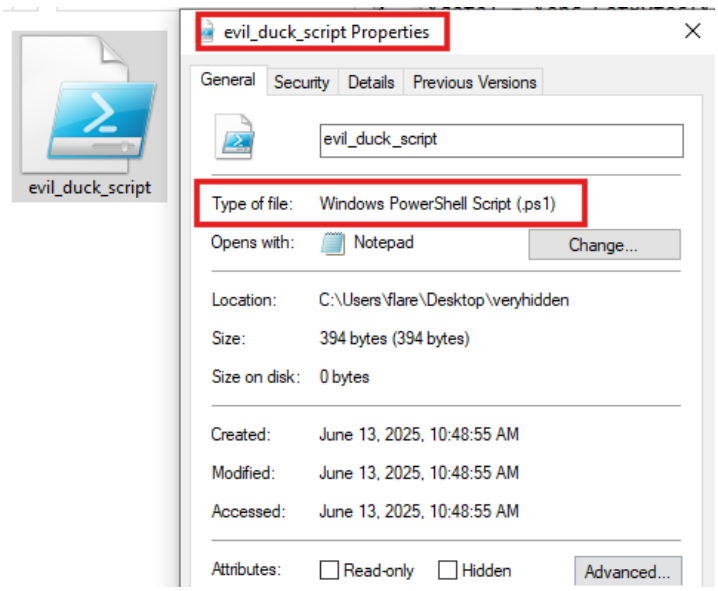


a.
b. We can see that there is a script inside the file. Underneath, it may be the hex dump for the png file.

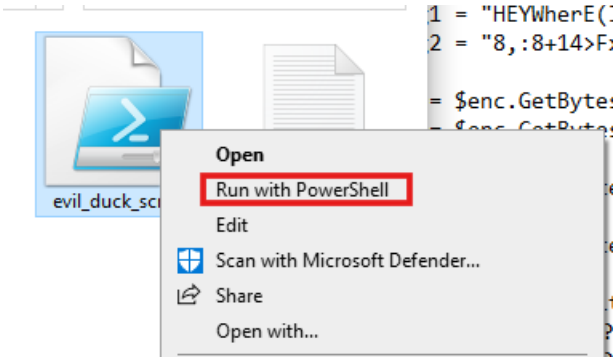
7. We will copy the script and create a new file with it and run it.



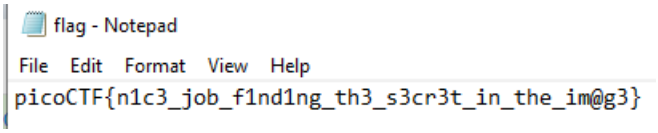
a.
b. In the script, we can see that there are 2 strings being xor'd together. This will produce the flag.
c. We copied it to a file called evil_duck_script.ps1.

i. 

d. We will run the script with PowerShell.

e. 

f. It produces a flag.txt file with the contained flag.

g. 

h. picoCTF{n1c3_job_f1nd1ng_th3_s3cr3t_in_the_im@g3}

i.