Laboratory Exercise – Metasploitable CTF Part 5 (Exercise J5)

1. Overview

In this lesson, students will learn how to analyze a web application to discover a hidden flag. They will use Burp Suite to capture requests, write a bash script to capture a file before it is deleted, write a deobfuscation script in Ruby, and base64 decode in the terminal.

2. Resources required

This exercise requires the Kali Linux with Metasploitable (2020.09) running in the Cyber Range.

3. Initial Setup

For this exercise, you will log in to your Cyber Range account, select the Kali Linux with Metasploitable (2020.09) environment, click "start" to start your environment, and then "join" to get to your Linux desktop.

4. Tasks

Task 1: Find the 6_of_Clubs Flag

From our previous scans we saw that port 8181 was open and running a WEBrick 1.3.1 Ruby 2.3.7 version.



| <pre>msf5 > works [*] Workspac msf5 > servi Services =======</pre> | pace m e: met .ces | etasplo asploit | itable able | | |
|--|--------------------------|--------------------|----------------|----------|--|
| host | port | proto | name | state | info |
| 10.1.32.187 | 21 | tcp | ftp | open | ProFTPD 1.3.5 |
| 10.1.32.187 | 22 | tcp | ssh | open | OpenSSH 6.6.1p1 Ubuntu 2ubuntu2 Ubuntu Linux; protocol 2.0 |
| 10.1.32.187 | 80 | tcp | http | open | Apache httpd 2.4.7 |
| 10.1.32.187 | 137 | tcp | netbios-ns | filtered | |
| 10.1.32.187 | 139 | tcp | netbios-ssn | filtered | |
| 10.1.32.187 | 445 | tcp | netbios-ssn | open | Samba smbd 3.X - 4.X workgroup: WORKGROUP |
| 10.1.32.187 | 631 | tcp | ipp | open | CUPS 1.7 |
| 10.1.32.187 | 3000 | tcp | ppp | closed | |
| 10.1.32.187 | 3306 | tcp | mysql | open | MySQL unauthorized |
| 10.1.32.187 | 3500 | tcp | http | open | WEBrick httpd 1.3.1 Ruby 2.3.7 (2018-03-28) |
| 10.1.32.187 | 6607 | tcp | | filtered | |
| 10.1.32.187 | 6667 | tcp | irc | filtered | |
| 10.1.32.187 | 6697 | tcp | irc | open | UnrealIRCd |
| 10.1.32.187 | 8181 | tcp | http | open | WEBrick httpd 1.3.1 Ruby 2.3.7 (2018-03-28) |
| 10.1.32.187 | 8484 | tcp | unknown | filtered | |
| 10.1.32.187 | 8585 | tcp | | filtered | |
| 10.1.32.187 | 8989 | tcp | sunwebadmins | filtered | |
| 10.1.32.187 | 9200 | tcp | wap-wsp | filtered | |
| <u>msf5</u> > | | | | | |

- Start Burp Suite and turn the intercept off.
- (If not already added) In a browser, install the FoxyProxy add-on and set the proxy up for Burp Suite as we have in previous lessons. (See Module 6, Lesson 3 Web App Manual Recon with Burp Task 1.)
- In the browser, navigate to <metasploitable ip>:8181 and press enter.
- Click the link on the web page.

| 10.1.32.187:83 | 181/ | × H | F | | | |
|---|-----------------------------------|--------------------------|--------------------------|----------------------|-----------------------|-------------|
| (←) → C | ۵ | | (i) 10.1 . | 32.187 :8181 | | |
| 🛕 Kali Linux | 🍾 Kali Trai | ning 🌂 | Kali Tools | 🧧 Kali Docs | 🌂 Kali Forums | <u>ک</u> Ne |
| Welcome to If you explo | Metasploi | table3 lication | - Linux ec , you will | lition. be handso | mely rewarde | d. |
| 10.1.32.187:8181/f | ilag 🗙 – | - | | | | |
| $\overleftarrow{\leftarrow}$ \rightarrow \overleftarrow{c} $\overleftarrow{\omega}$ | | (i) 10.1.3 | 32.187 :8181/fla | g | | |
| 🕅 Kali Linux 🌂 I | Kali Training 🌂 | Kali Tools | 🧧 Kali Docs 🎽 | 🔍 Kali Forums 📝 | 🔪 NetHunter 🛛 👖 Offen | isive Secu |
| The /flag route Problem is co | can give you a ookies are sigr | a flag if th ied. Hmm | ne _metasplo m | itable cookie h | has the name of the | e flag. |

• In Burp Suite, click the **Proxy** tab and the **HTTP history** tab.

Here you see the webpage we visited in the browser.



• Click on the HTTP History URL that contains the /flag.

| Burp | Project Intruder Repeater Win | ndow Help | р | | |
|---------|---------------------------------|-------------|---------|-----------|--------|
| Dash | board Target <u>Proxy</u> Intr | uder Re | epeater | Sequencer | Decode |
| Inter | cept <u>HTTP history</u> WebSoc | kets histor | ry Opti | ons | |
| Filter: | Hiding CSS, image and general | binary cor | ntent | | |
| # 🔺 | Host | Method | URL | | |
| 1 | http://10.1.32.187:8181 | GET | 1 | | |
| 2 | http://10.1.32.187:8181 | GET | /flag | | |
| 3 | http://10.1.32.187:8181 | GET | /flag | | |
| | | | | | |

• Right click on the Raw request and then on the **Send to Repeater**.

| Request | | |
|--|--|---------------------------------------|
| Raw Params Headers Hex | | |
| Pretty Raw \n Actions ∨ 1 GET /flag HTTP/1.1 2 Host: 10.1.32.187:8181 2 Host: Marilla (5.0 (X1)) Line | | Ti sefer (C |
| 3 User-Agent: Mozilla/5.0 (X11; Line 4 Accept: text/html,application/xhtml | Scan | E1 notox / 68 |
| <pre>5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Referer: http://l0.1.32.187:8181/ 8 Connection: close 9 Cookie: _metasploitable= BAh7B0kiD3NLc3Npb25faWQG0gZFVEkiRT DNhZmMzMGM4MmY1MWMxNWQwMWYxMjIzMzE hoaCwgZG9uJ3QgdGVs%0AbCBhbnlib2R51 NGU2NGY5NDc0MTVhOTRLNWYG0wBU%0Ae</pre> | Send to Intruder <u>Send to Repeat</u> er Send to Sequencer Send to Comparer Send to Decoder Show response in browser Request in browser | Ctrl+I Ctrl+R /j` Ek: Dhr |
| 10 Upgrade-Insecure-Requests: 1 | Engagement tools [Pro version only] | • |
| 11 12 | Copy URL Copy as curl command Copy to file | |
| ⑦ (i) ← → Search | Save item | 0 |

• Click on the **Repeater** tab and click the **Send** button to make the request.

Looking at this response, we can see the cookie is being passed as a base64 encoded string using an old Ruby technique. You may not recognize this as a newcomer; however, the **=BAh** is the first hint for Ruby and Base64. You may have also noticed from previous lessons that Ruby was running on the web application server (WEBrick).





| Suite Community E 📦 FoxyProxy Standard – G 🧿 10.1.87.191:8181/flag 🧿 FoxyProxy - | - FoxyProxy S 🖿 Terminal - student@kali 12:19 AM 🜒 🌲 💿 🖨 |
|---|--|
| Burp Suite Community Edition v202 | 20.9.1 - Temporary Project _ 🗖 🗙 |
| Burp Project Intruder Repeater Window Help Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender P 1 = | roject options User options |
| Send Cancel < * > * | Target: http://10.1.87.191:8181 🖉 🕐 |
| Request Raw Params Headers Hex | Response = = |
| <pre>Pretty Eaw In Actions v 1 Set /flag HTTP/1.1 2 Host: 10.1.87.191:8181 3 Uggrade.Insecure?Requests: 1 4 User-Agent: host11ar5.0 (N11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrose/100.0.0 Safari/s1537.36 5 tsrt/htal.application/signde.eckchang:v=b3;q=0.9 6 Referer: http://10.1.87.191:8181/ 7 Accept-Language: en-US.en;q=0.9 9 Cookie: metsploitable= BAh780kiDSNL:894b25faNQC0a2FVEkiRTV1NeU4ZbkHT1xYJUSHjAAMN0050AHjYSNZ2iY202001}YTYS2 1 MU2EXPRAYEENg 2004'Y yy 2021CHROMOGOADMESSIUK211 dGFz:GxvaXFN*mal1B;aVEkiVFNac hacvy205U330gdfv5t0AbCdFhonlib2FSIFE0aMMy729va211THN1X31dBoyTdfzWiJHjg%shHKOltoA NUCUNKYSNCOMTVHCTMLWYGOADJA0A46f9d4daf312d97e4b955c5442526416b41b15f 10 Connection: close 11 12 </pre> | Pretty Raw Render In Actions w 1 HTTP/1.1 200 OK Image: Standard St |

Here is my screenshot for the above steps.

- In the Request, select all the red text starting with and after Bah, right click on it, and then click **Send to Decoder**.
- Click the **Decoder** tab and click **Smart decode**.

| BAh780kiD3Nic3Npb25faWQG0gZFVEikRTdINTALZDYxHJASNDZhOGjkZWI0 <mark>607</mark> NTBhNDc3YmFhNYY2YmM3NDhZmMzMGM4MmY1MWMxNWQwMWYxMjizMzBhNjcG <mark>&rQ</mark> OwBGSSiUX21ldGFzCGxvaXbhYmxiBjsAVEkiVFN oaGhoaCwgZG9uj3OgdGVs <mark>111</mark> bCBhbnlib2R5iHRoaXMgY29va2lliHNIY3JdDogYTdhZWjjMjg3YmjhMGV <mark>111</mark> NGU2NGY5NDcOMTVhOTRINWYGOwBU <mark>9716</mark> -e9c05038d38be0371d05bdb86697770103be7ea5 | O Text O Hex O Decode as V Encode as V Hash Smart decode |
|---|--|
| BAh780kiD3NiC3Npb25fawQGogZPrElisRTdIMTA120YxNiJASNDZhOGjkZWI0 NTEND023Ym#hYY2YmM3R0NhZmMxMGM4MmY3MiJAH2BhBJEG OwBGSSUUX211GfzsCxxxxRhYmxBj3AVKkVFNoaGhoaCwgZGsU30gdCvS bCBhbniL28ZHRAXMy23va2UHINYJIB0g0YTdIA2WJMJg3YmJhMGVI NGU2XGYSNDC0MTVhOTRIWWYGOwBU -e9c0S038d38be0371d05bd886697770103be7eaS | Text Hex Decode as Encode as Hash Smart decode |

• Now click the **Decode as** drop down and choose **Base64**.

This appears to be the key to decode the cookie.



Joshua Lane CYSE450 Section 23190 11/08/2022

| B/ | Ah7B0kiD3I FBhNDc3Yr | Nic3Npb2 nFhYjY2Y | 5faWQG(mM3NDN |)gZFVEki hZmMzM | RTdIMTA12 IGM4MmY1 | ZDY×NjA5N LMWM×NW0 | NDZhOGJKZ QwMWYxMjiz | WIO zMzBhNjEG | | | | | | | | | | | Text O Hex | |
|-------------|-----------------------------------|-----------------------------------|--------------------------------|---------------------------------|----------------------------------|------------------------|---------------------------|------------------------|----------------|----------------|----------------|----------------|----------------------------|----------------|----------------|-------------------------|---------------------------|--|-------------------|---|
| Di b(| wBGSSIUX Bhbnlib2R GU2NGY5N | 21IdGFzc SIHRoaXM IDc0MTVh | GxvaXRh 4gY29va3 OTRINW1 | (mxlBjsA 2lliHNlY3j GOwBU | VEkiVFNoa IdDogYTdh | GhoaCwg ZWJjMjg3Y | ZG9uJ3Qgd (mJhMGVI | GVs | | | | | | | | | | | Plain | |
| | 9c05038d | 38be037 | 1d05bdb | 36697770 | 0103be7ea | 15 | | | | | | | | | | | | | URL HTML | |
| | | | | | | | | | | | | | | | | | | | ASCII hex | |
| 6 | | 22 | 51 14 | 52 5f | 52 6d | 55 65 | 55 74 | 50 61 | 01 73 | 50 70 | 51 60 | 00 6f | 0a 69 | 50 74 | 61 | 40 | 49 6c | = //_metasploitabl | A Octal Binary | |
| 8 9 | | 64 6f | 6f 64 | 6e 79 | 27 20 | 74 74 | 20 68 | 74 69 | 65 73 | 6c 20 | 0a 63 | 6c 6f | 20 6f | 61 6b | 6e 69 | 79 65 | 62 20 | don't tell anyb ody this cookie | Gzip Encode as | P |
| a b c | | 73 37 34 | 65 62 31 | 63 62 35 | 72 61 61 | 65 30 39 | 74 65 34 | 3a 65 65 | 20 0a 35 | 61 34 66 | 37 65 06 | 61 36 3b | 65 34 00 | 62 66 54 | 63 39 0a | 32 34 2d | 38 37 2d | secret: a7aebc28 7bba0ee4e64t947 415a94e5f ///.T | Hash | |
| d e | | 7b d6 | d7 fc | 34 eb | e7 af | 4d 7b | fc ef | 77 bd | 7f 35 | 1b d3 | 7b 76 | 4d de | fb ed | d5 e6 | dd b9 | 39 | 6d | {×4çMuw∳{MûŐÝ9m Öüē`{ï½5ÓvÞíæ* | Smart decode | _ |
| e/#/ | client/NG | YOMTZh' | YTgtMjY: | (My00M | DhjLWE4Z | ZDgtYWY! | 5NDljNmU: | xNzFhLzAx | xNDA1MTQ | WLTI3MDA | tNDQ4Z0 | Timdu0 | LTBhN2Uzi | 12JjOW | Q4NC9kZ | ZmQ2NGUy | OC0xYTNkL | rriytktodii 볩 🖻 | * 🗖 🏟 | |
| o Su | ite Comm | nunity E. | . 🧆 Fa | xyProxy | Standard | d – G 🔇 | 10.1.87 | .191:8181 | l/flag 🤇 | FoxyPr | oxy - Fo | cyProxy S | 5 🔳 Ter | minal - | student | @kali | | 12:22 AM 🌒 | A 0 6 | |
| | Burn Proj | ect Intru | der Rep | ater Wir | adaw Help | | | Burp | o Suite Commu | unity Edition | v2020.9. | 1 - Tempor | ary Project | | | | | | _ = × | |
| | Dashboa | rd Tan | get Pro | xy Intr | uder Rej | peater s | Sequencer | Decoder | Comparer | Extender | r Projec | t options | User opti | ons | | | | | | |
| ıbl | BAh7B0 | kiD3Nlc3M | Vpb25faW | QGOgZF | VEkiRTY1N | mU4ZDkxt | MTIxYjU3NjA | wNWQ0 <mark>%0/</mark> | MjY3N2ZiY2 | QZODIjYTY5 | ZjM4M2E3 | M2M3YzE | zNj <u>g20</u> GRiYj | gyZDZI | DTM20WM | G <mark>%0A</mark> OwBG | SSIUX21IdGFz | cG 🖲 Text 🕞 Hex | ? | |
| I | af312d9 | (mxlBjsA\)7e4b955 | /EkiVFNo/ c544252(| iGhoaCw 5e416b41 | gZG9uJ3Q(b15f | gdGVs <mark>%0A</mark> | bCBhbnlib2 | R5IHRoaXM | 1gY29∨a2lliHN | lY3JldDogY1 | ſdhZWJjMj | g3YmjhMG | GVI <mark>% DA</mark> NGU: | NGY5NI | 0c0MTVhO | TRINWYGOW | BU <mark>%0A</mark> 46f9(| Decode as | | |
| I | | | | | | | | | | | | | | | | | | Hash | | |
| I | | | | | | | | | | | | | | | | | | Smart decor | le | |
| I | BAh7B0 MjY3N23 | kiD3Nlc3i ZiY2QzOD | Npb25faW IljYTY5Zji | QGOgZF 44M2E3M | VEkiRTY1Ni I2M3YzEzNji | mU4ZDkxt ig2OGRiYjg | MTIXYjU3NjA jyZDZIOTM2 | wNWQ0 2OWMG | | | | | | | | | | Text Hex | | |
| I | OwBGS bCBhbn NGU2N0 | SIUX211d 11b2R51HR GY5NDc01 | GFzcGxva oaXMgY2 4TVhOTR | XRhYmxli 9va2lliHN NWYGOw | BjsAVEkiVFI IlY3JIdDogY BU | NoaGhoaC TdhZWJjMj | wgZG9uJ30 jg3YmJhMGV |)gdGVs ∕I | | | | | | | | | | Encode as | • | |
| I | 46f9d4 | daf312d | 97e4b955 | c544252 | 6e416b41b | 515f | | | | | | | | | | | | Hash | V | |
| | | | | | | | | | | | | | | | | | | Smart decod | le | |
| | | | | | | | | | | | | | | | | | | | | |

Here is my screenshot for decoding the base64.

This particular method of grabbing the flag can be tricky. We would have to use the extracted key, add 6_of_clubs to the _metasploitable cookie, sign the cookie, and then send the request. There is another way to grab the flag because we have access to the application on the metasploitable box itself. Let's see what else we can find to solve this flag. Logon as luke_skywalker to see what we can find about the Ruby web application.

3a73c7c13

36 38 63 06 6c 6f 68 68 02 6c

- In a root terminal, log on as luke_skywalker using the **ssh** command as completed in previous lessons.
- Type cd /opt and press enter.

Notice the sinatra directory. Sinatra is an open-source Ruby web application library. To see all running processes that are being executed by the sinatra library, we can use the **ps aux** command. You can visit the following site for a better understanding of the **ps aux** command if you are not familiar with it:

https://www.computernetworkingnotes.com/linux-tutorials/ps-aux-command-and-ps-command-explained.html&sa=D&source=editors&ust=1626829763004000&usg=AOvVaw1ITpucA7Y8cXehZNahZ7Vq

- In the SSH terminal, type cd sinatra and press enter.
- In the SSH terminal, type **ps aux** | **grep sinatra** and press enter.



Joshua Lane CYSE450 Section 23190 11/08/2022



Here is my screenshot after logging into luke_skywalker and using the above commands.

Wow, this is a great hint. The output that is important reads:

```
"require 'obfuscate'; Obfuscate.setup { |c| c.salt = 'sinatra'; c.mode =
:string }; cr = Obfuscate.clarify(File.read('.raIhUJTLEMAfUW3GmynyFySPw'));
File.delete('.raIhUJTLEMAfUW3GmynyFySPw') if
File.exists?('.raIhUJTLEMAfUW3GmynyFySPw'); eval(cr)" --
```

This is the Ruby code for obfuscating the hidden file .ralhUJTLEMAfUW3GmynyFySPw. I am betting this is our flag. The code then deletes the hidden file and this is why we can't see it. We need to grab the file before it is deleted. Once we have the file, we can deobfuscate it with this code. For more information, the Ruby Obfuscator github repository can be found here.

We first need to obtain the file. Since we know sinatra is the service creating the file, we can create a quick bash script to copy the file when it is seen.

- On the metasploitable ssh terminal, type **mkdir** deobfuscate and press enter.
- Type cd deobfuscate and press enter.
- Type nano flagfile.txt and press enter.
- Copy and paste the following script:



```
#!/bin/bash
while :
do
    if [ -f /opt/sinatra/.raIhUJTLEMAfUW3GmynyFySPw ]
    then
        cp /opt/sinatra/.raIhUJTLEMAfUW3GmynyFySPw
/opt/sinatra/deobfuscate
        echo "Hidden file copied!"
        exit 0
    fi
done
```

• Press CTRL+X and when prompted to save type **Y** and press enter.



We need to make the script executable and have it run in the background.

- Type chmod +x flagfile.txt and press enter.
- Type ./flagfile.txt & and press enter.

luke_skywalker@ip-10-1-32-187:/opt/sinatra/deobfuscate\$ chmod +x flagfile.txt luke_skywalker@ip-10-1-32-187:/opt/sinatra/deobfuscate\$./flagfile.txt & [2] 2317

Before the script will fire, we need to stop and start the sinatra service.



- Type sudo service sinatra stop and press enter.
- You may need to type the password for luke_skywalker and press enter.
- Type sudo service sinatra start and press enter.

You will see a prompt "hidden file copied!"

• Press **CTRL+C** to exit the prompt from the script.

```
luke_skywalker@ip-10-1-32-187:/opt/sinatra/deobfuscate$ sudo service sinatra stop
stop: Unknown instance:
luke_skywalker@ip-10-1-32-187:/opt/sinatra/deobfuscate$ sudo service sinatra start
sinatra start/running, process 2385
luke_skywalker@ip-10-1-32-187:/opt/sinatra/deobfuscate$ hidden file copied!
zc1iMdU0LTBhN2UzN2JjOWQ4NC9kZmQ2NGUyOC0xYTNkLTRIYTktODII... 📋 🖄 🛧 🗌
                                                                         /Proxy ... 🔲 Terminal - student@kal... 🔲 Terminal - luke_skywal... 12:29 AM 🌒 🛕 📀 |
               Terminal - luke_skywalker@ip-10-1-87-191: /opt/sinatra/deobfuscate
                                                                               D X
   File Edit View Terminal Tabs Help
   tra'; c.mode = :string }; cr = Obfuscate.clarify(File.read('.raIhUJTLEMAfUW3Gmy
  nyFySPw')); File.delete('.raIhUJTLEMAfUW3GmynyFySPw') if File.exists?('.raIhUJTL
  EMAfUW3GmynyFySPw'); eval(cr)" --
E3
  root 1813 1.3 2.3 240528 23800 ? Sl Nov08 0:22 ruby -e require
'obfuscate'; Obfuscate.setup { |c| c.salt = 'sinatra'; c.mode = :string }; cr =
            1813 1.3 2.3 240528 23800 ?
  root
   Obfuscate.clarify(File.read('.raIhUJTLEMAfUW3GmynyFySPw')); File.delete('.raIhU
  JTLEMAfUW3GmynyFySPw') if File.exists?('.raIhUJTLEMAfUW3GmynyFySPw'); eval(cr)
  luke_sk+ 2261 0.0 0.0 10476 924 pts/3
                                                S+ 00:24 0:00 grep --color=au
  to
  luke_skywalker@ip-10-1-87-191:/opt/sinatra$ mkdir deobfuscate
  luke_skywalker@ip-10-1-87-191:/opt/sinatra$ cd deobfuscate
  luke_skywalker@ip-10-1-87-191:/opt/sinatra/deobfuscate$ nano flagfile.txt
  luke_skywalker@ip-10-1-87-191:/opt/sinatra/deobfuscate$ chmod +x flagfile.txt
  luke_skywalker@ip-10-1-87-191:/opt/sinatra/deobfuscate$ ./flagfile.txt δ
  [1] 2267
  luke_skywalker@ip-10-1-87-191:/opt/sinatra/deobfuscate$ sudo service sinatra sto
  [sudo] password for luke_skywalker:
  sinatra stop/waiting
  luke_skywalker@ip-10-1-87-191:/opt/sinatra/deobfuscate$ sudo service sinatra sta
  rt
  sinatra start/running, process 2283
  luke_skywalker@ip-10-1-87-191:/opt/sinatra/deobfuscate$ Hidden file copied!
```

Here is my hidden file copied.

• Type **ls** -**a** and press enter.

luke_skywalker@ip-10-1-32-187:/opt/sinatra/deobfuscate\$ ls -a
. flagfile.txt flag.txt .raIhUJTLEMAfUW3GmynyFySPw
luke_skywalker@ip-10-1-32-187:/opt/sinatra/deobfuscate\$





Here you can see my ls -a command.

Now we need to create a script to deobfuscate the code. This script was given to us when we found the sinatra process running using the *ps aux* | *grep sinatra* command. In our case, however, we are not going to add the delete file section of the script and we need to direct the "Obfuscate.clarify" to the location of our file.

- Open a text editor on the Kali VM.
- Copy and paste the following code and save it as **deobfuscate.rb** to your **flags** directory.

CODE:

```
require 'obfuscate'
Obfuscate.setup do |c|
c.salt = 'sinatra'
c.mode = :string
end
cr =
Obfuscate.clarify(File.read('/opt/sinatra/deobfuscate/.raIhUJTLEMAfUW3GmynyFy
SPw'))
print cr
```



• In a new root terminal, change directory to **flags** and copy the deobfuscate.rb to the metasploitable system by typing:





scp deobfuscate.rb luke_skywalker@<metasploitable ip>:/opt/sinatra and pressing enter.

```
ZarRonRoot@/Flags$:scp deobfuscate.rb luke_skywalker@10.1.32.187:/opt/sinatra
luke_skywalker@10.1.32.187's password:
deobfuscate.rb
ZarRonRoot@/Flags$:
```

- In the metasploitable ssh terminal, be sure you are in the sinatra directory; if not, type cd /opt/sinatra and press enter.
- Type ruby deobfuscate.rb > /opt/sinatra/deobfuscate/flag.txt and press enter.

luke_skywalker@ip-10-1-32-187:/opt/sinatra\$ ruby deobfuscate.rb > /opt/sinatra/deobfuscate/flag.txt

• Type nano /opt/sinatra/deobfuscate/flag.txt and press enter.

You can mouse scroll to analyze the whole file.

```
#!/usr/bin/env ruby
require 'sinatra'
require 'erubis'
require 'active_support'
require 'webrick'
require 'base64'
MYSECRET = 'a7aebc287bba0ee4e64f947415a94e5f'
set :environment, :development
set :bind, '0.0.0.0'
set :port, 8181
# These settings are specific for Sinatra 2.0.0rc2
set :logging, false
set :quiet, true
dev_null = WEBrick::Log::new("/dev/null", 7)
set :server_settings, {:Logger => dev_null, :AccessLog => dev_null}
use Rack::Session::Cookie,
  :key => "_metasploitable",
:path => "/",
  :expire_after => 1800,
  :secret
                => MYSECRET
```



Joshua Lane CYSE450 Section 23190 11/08/2022



Here is showing the message in the file.

Now we can see the same cookie secret we discovered by using the base64 decode when we captured the request earlier using Burp Suite. We also see the base64 blob that contains the flag. We have to extract only the base64 code so we can decode the png. This can be tricky in the SSH terminal.

- Delete all the lines except for the line that contains b64. There are several ways of doing this; however, what worked for me was using the arrow keys and CTRL+K to delete each line.
- Delete everything before the i on the b64 line using the arrow keys and the delete button.
- Delete everything after == at the end of the line. To get to the end of the line press CTRL+E.

Beginning of the line:



GNU nano 2.2.6

File: flag.txt

iVBORw0KGgoAAAANSUhEUgAAAfQAAAK8CAYAAAAZNU0WAAAAAXNSR0IArs4c6QAAQABJREFUeAHsvQmYrUdV91u7u3ePp8+UeQ4JEASigqiIASSgIlw+RWRS8V\$

End of the line:

| GNU nano 2.2.6 Fi | ile: flag.txt | | | |
|--|---|---------------------|------------------|-----|
| zyoHKgcqByoHKga3gALjUAa9mwaVZypttuq3n+X2QLdEPg | gle/ag93f3LVVVcd3gpaXq2 | 2М/w9TcBjiHMpKN | wAAAABJRU5ErkJgg | g=: |
| 1iMDU0LTBhN2UzN2JjOWQ4NC9kZmQ2NGUyOC0xYTNkLTF | RIYTktODII 🖺 🖻 🖈 |) 🗆 🎲 🗄 | | |
| 🔲 Terminal - stud 🔲 Terminal - luke 🖳 Terminal | l - stud 12:40 AM 🌗 🖠 | • 0 | | |
| Terminal - luke_skywalker@ip-10-1-87-19 | 91: /opt/sinatra | _ = × | | |
| File Edit View Terminal Tabs Help | | | | |
| GNU nano 2.2.6 File: /opt/sinatra/deobf | uscate/flag.txt | Modified | | |
| \$BjiHMpKNwAAAABJRU5ErkJggg== | | 3 | | |
| | | | | |
| | | (v) | | |
| | | (*) | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| [^] G Get Help [^] O WriteOut [^] R Read File [^] Y Prev [^] X Exit [^] J Justify [^] W Where Is [^] V Next | /Page <mark>^K</mark> Cut Text <mark>^C</mark> Page <mark>^U</mark> UnCut Text <mark>^T</mark> | Cur Pos To Spell | | |

Here is my screenshot of the end of the line.

- Press CTRL+X.
- Type **Y** and press enter.
- Type cd deobfuscate and press enter.

Now we can decode the base64 png.

• Type cat flag.txt | base64 -d > 6_of_clubs.png and press enter.



To see the flag, we will have to transfer it to the Kali VM.

• In the Kali VM terminal, type

scp luke_skywalker@<metasploitable ip>:/opt/sinatra/deobfuscate/6_of_clubs.png
/home/student/Desktop/flags/ and press enter.



To open the flag, navigate to the **flags** folder and double click the **6_of_clubs.png** file. Congratulations, you have successfully obtained the flag!



MY SCREENSHOT OF THE FLAG IS ON NEXT PAGE



13



Here you can see the screenshot of the 6 of clubs flag.

In this lesson, you learned how to analyze a web application to discover a hidden flag. You used Burp Suite to capture requests and decode a cookie, analyzed a running web application using the ps aux command, wrote a bash script to capture a file before it was deleted, wrote a deobfuscation script in Ruby, and base64 decoded an image in the terminal.

