CAPTURING PACKETS USING TCPDUMP

A LAB BY DARNELL MAIDEN

```
analyst@2cf2d4692d77:~$ sudo ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1460
        inet 172.17.0.2  netmask 255.255.0.0  broadcast 172.17.255.2
55
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)
        RX packets 686  bytes 13997167 (13.3 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 375  bytes 35560 (34.7 KiB)
        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 53  bytes 8432 (8.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 53  bytes 8432 (8.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

SUMMARY:

OKAY, SO TODAY I AM ON TCPDUMP DUTY (SO LUCKY, RIGHT?). I'VE BEEN GIVEN THE GRAND TASK TO CAPTURE SOME PACKETS USING THIS ENTER THE MATRIX LINUX VIRTUAL SYSTEM. GUESS I'M THE CHOSEN ONE LIKE NEO. WELL FIRST, I HAVE TO IDENTIFY THE NETWORK INTERFACES. SECOND, I'LL USE TCPDUMP TO FILTER THE LIVE TRAFFIC AND THEN I'LL CAPTURE IT WITH TCPDUMP AS WELL. LASTLY, I'LL FILTER THE CAPTURED PACKET DATA, GIVE THE INFORMATION TO THE LEAD, GET AN IMAGINARY PAT ON THE BACK, THEN GO HOME AND CRY WHILE EATING CHICKEN WINGS WITH TELEMUNDO ON. ENOUGH JOKES. LET'S GET TO IT SHALL WE?

IDENTIFYING NETWORK INTERFACES:

SO LET'S DIVE IN. I NEED TO IDENTIFY THE NETWORK INTERFACES THAT CAN BE USED TO CAPTURE NETWORK PACKETS. TIME TO USE THE SUDO PRIVILEGES WITH THE IFCONFIG COMMAND.
COMMAND: [SUDO IFCONFIG]

[SUDO]- ALLOWS AN AUTHORIZED USER TO EXECUTE COMMANDS WITH THE PRIVILEGES OF ANOTHER USER, TYPICALLY THE SUPERUSER (ROOT)

[ICONFIG]- A SYSTEM ADMINISTRATION UTILITY USED IN UNIX-LIKE OPERATING SYSTEMS TO DISPLAY, CONFIGURE, AND CONTROL NETWORK INTERFACE PARAMETERS.

```
analyst@2cf2d4692d77:~$ sudo ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1460
        inet 172.17.0.2  netmask 255.255.0.0  broadcast 172.17.255.2
55
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)
        RX packets 686  bytes 13997167 (13.3 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 375  bytes 35560 (34.7 KiB)
        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 53  bytes 8432 (8.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 53  bytes 8432 (8.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

THIS IS THE OUTPUT THAT I GET AFTER USING THE COMMAND ABOVE. THE ETHERNET NETWORK INTERFACE IS IDENTIFIED BY THE ENTRY WITH THE [ETH] PREFIX. I'LL BE USING THE [ETH0] INTERFACE TO CAPTURE NETWORK PACKET DATA. TO IDENTIFY THE INTERFACE OPTIONS AVAILABLE FOR PACKET CAPTURE I'LL USE:
COMMAND: [SUDO TCPDUMP -D]

```
analyst@2cf2d4692d77:~$ sudo tcpdump -D
1.eth0 [Up, Running, Connected]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.bluetooth-monitor (Bluetooth Linux Monitor) [Wireless]
5.nflog (Linux netfilter log (NFLOG) interface) [none]
6.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
7.dbus-system (D-Bus system bus) [none]
8.dbus-session (D-Bus session bus) [none]
```

AND JUST THAT, THE OPTIONS APPEAR WITH FURTHER INFORMATION.

IDENTIFYING TRAFFIC IN A NETWORK INTERFACE. TCPDUMP STYLE.:

NOW THAT I HAVE YOUR ATTENTION (OR SOME OF IT), I CAN NOW BEGIN TO FILTER LIVE TRAFFIC USING TCPDUMP FROM THE **[ETH0]** INTERFACE USING THE COMMAND BELOW.

COMMAND: [SUDO TCPDUMP -I ETH0 -V -C5]

THE COMMAND WILL RUN TCPDUMP WITH THESE OPTIONS:

- **-I ETH0:** CAPTURE DATA SPECIFICALLY FROM THE ETH0 INTERFACE.
- **-V:** DISPLAY DETAILED PACKET DATA
- **-C5:** CAPTURE 5 PACKETS OF DATA

```
analyst@2cf2d4692d77:~$ sudo tcpdump -i eth0 -v -c5
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot le
ngth 262144 bytes
05:03:33.851462 IP (tos 0x0, ttl 64, id 15193, offset 0, flags [DF],
 proto TCP (6), length 122)
    2cf2d4692d77.5000 > nginx-us-east1-b.c.qwiklabs-terminal-vms-pro
d-00.internal.43924: Flags [P.], cksum 0x599e (incorrect -> 0x6fb2),
```

```
 seq 70:392, ack 1, win 998, options [nop,nop,TS val 1742227210 ecr
3158479695], length 322
5 packets captured
3 packets received by filter
0 packets dropped by kernel
```

AND IN RESULT, 5 PACKETS WERE CAPTURED. 3 PACKETS WERE RECEIVED BY FILTER. 0 PACKETS DROPPED BY KERNEL. ONCE AGAIN, TECHNOLOGY SHOWS IT'S BEAUTIFUL FACE.

PACKET? I BARELY KNOW IT! (THIS IS THE PACKET ANALYSIS SECTION, I'M SORRY I'M TRASH AT JOKES.):

SO NOW THAT WE EXAMINED SOME LIVE TRAFFIC ACTION, LET'S TAKE A CLOSER LOOK AT WHAT WE CAUGHT ON THE HOOK CALLED TCPDUMP, SHALL WE?

IN THIS SECTION OF THE PACKET OUTPUT, IT'S TELLING US THAT TCPDUMP REPORTED THAT IT WAS LISTENING ON THE ETH0 INTERFACE FROM EARLIER ALONG WITH THE LINK TYPE AND THE CAPTURE SIZE. IF ONLY IT COULD CAPTURE THE SLEEP DEPRIVATION ON MY FACE TYPING THIS.

```
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot le
ngth 262144 bytes
```

THIS IS THE TIMESTAMP FOLLOWED BY THE PROTOCOL TYPE, IP (INTERNET PROTOCOL).

```
05:03:33.851462 IP
```

BECAUSE WE USED THE VERBOSE OPTION (-V), IT GAVE US MORE INFORMATION SUCH AS THE TOS (TYPE OF SERVICE), THE TTL (TIME TO LIVE), OFFSET (NO, NOT HIM), FLAGS, THE INTERNAL PROTOCOL (TCP), AND THE LENGTH OF THE OUTER IP IN BYTES.

```
05:03:33.851462 IP (tos 0x0, ttl 64, id 15193, offset 0, flags [DF],
 proto TCP (6), length 122)
```

THIS IS THE SECTION WHERE THE DATA DISPLAYS INFORMATION ABOUT THE SYSTEMS THAT ARE COMMUNICATING WITH EACH OTHER. TCP CONVERTS IP ADDRESSES INTO NAMES BY DEFAULT. MY VM NAME APPEARS IN THE PROMPT AS THE SOURCE FOR ONE PACKET AND THE DESTINATION FOR THE SECOND PACKET. SO IN THE LIVE DATA, IT'LL BE A DIFFERENT SET OF LETTERS AND NUMBERS.

```
    2cf2d4692d77.5000 > nginx-us-east1-b.c.qwiklabs-terminal-vms-pro
d-00.internal.43924: Flags [P.], cksum 0x599e (incorrect -> 0x6fb2),
 seq 336836417:336836487, ack 3439963359, win 998, options [nop,nop,
TS val 1742227138 ecr 3158479637], length 70
```

THE DIRECTION OF THE ARROW (>) SHOWS THE DIRECTION OF TRAFFIC FLOW IN THE PACKET ITSELF. EACH SYSTEM NAME INCLUDES A SUFFIX WITH THE PORT NUMBER AT THE END, WHICH IS USED BY THE SOURCE AND DESTINATION SYSTEMS FOR THE PACKET. AS FAR AS THE FLAG SECTION GOES, THAT [P.] IS A PUSH FLAG AND THE PERIOD INSIDE OF IT INDICATES IT'S AN ACK FLAG. THIS SIMPLY MEANS THE PACKET IS PUSHING OUT DATA.

THE TCP CHECKSUM VALUE IS USED FOR DETECTING ERRORS IN DATA.
OTHER COMPONENTS INCLUDE SEQUENCE AND ACKNOWLEDGEMENT NUMBERS, WINDOW SIZE, AND THE LENGTH OF THE INNER TCP PACKET IN BYTES.

TIME TO PACK UP: CAPTURING A TCP PACKET (THE FUN STUFF, YAY.):

NOW THEN, LET'S ACT LIKE A 6 IN HER SELFIES AND DO SOME MORE FILTERING ALONG WITH THE TCPDUMP COMMAND. EXCITING. WE'LL CONFIGURE THE OPTIONS TO SAVE A SAMPLE THAT ONLY CONTAINS WEB NETWORK PACKET DATA A.K.A. TCP PORT 80.
I'LL USE THIS TO SEARCH FOR IT:

[SUDO TCPDUMP -I ETH0 -NN -C9 PORT 80 -W CAPTURE.PCAP &]

```
[1]+  Done                    sudo tcpdump -i eth0 -nn -c9 port 80 -
w capture.pcap

analyst@4e6daee2ace0:~$ sudo tcpdump -i eth0 -nn -c9 port 80 -w capt
ure.pcap &
[1] 13419
analyst@4e6daee2ace0:~$ tcpdump: listening on eth0, link-type EN10MB
 (Ethernet), snapshot length 262144 bytes
```

NOW KEEP IN MIND, THIS WILL RUN IN THE BACKGROUND, BUT I CAN STILL BREAK DOWN THESE COMPONENTS FOR YOU FOR A CLEAR UNDERSTANDING OF WHAT I'M DOING. THIS IS WHAT I'M SAYING:

- -I ETH0: CAPTURE DATA FROM THE ETH0 INTERFACE.
- -NN: DO NOT ATTEMPT TO RESOLVE IP ADDRESSES OR PORTS TO NAMES.THIS IS BEST PRACTICE FROM A SECURITY PERSPECTIVE, AS THE LOOKUP DATA MAY NOT BE VALID. IT ALSO PREVENTS MALICIOUS ACTORS FROM BEING ALERTED TO AN INVESTIGATION.
- -C9: CAPTURE 9 PACKETS OF DATA AND THEN EXIT.
- PORT 80: FILTER ONLY PORT 80 TRAFFIC. THIS IS THE DEFAULT HTTP PORT.

- -W CAPTURE.PCAP: SAVE THE CAPTURED DATA TO THE NAMED FILE.
- &: THIS IS AN INSTRUCTION TO THE BASH SHELL TO RUN THE COMMAND IN THE BACKGROUND.

GOT IT? COOL. NEXT TASK.

NOW, I'LL USE THE CURL COMMAND TO GENERATE SOME HTTP NETWORK TRAFFIC. LIKE THIS:

[CURL OPENSOURCE.GOOGLE.COM]

```
curl opensource.google.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html;chars
et=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="https://opensource.google/">here</A>.
</BODY></HTML>
analyst@4e6daee2ace0:~$ 9 packets captured
10 packets received by filter
0 packets dropped by kernel
```

WELL LOOK AT THAT, SOME TRAFFIC THAT CAN BE CAPTURED. LET'S GET TO IT.

NOW WE'LL VERIFY THAT THE PACKET WAS CAPTURED WITH THIS:

[LS -L CAPTURE.PCAP]

```
ls -l capture.pcap
-rw-r--r-- 1 tcpdump tcpdump 1401 Mar 11 23:27 capture.pcap
[1]+  Done                    sudo tcpdump -i eth0 -nn -c9 port 80 -
w capture.pcap
analyst@4e6daee2ace0:~$
```

"DONE" IS THE INDICATION THAT THE PACKET WAS CAPTURED. WE'RE DOING GOOD SO FAR. NEXT TASK.

FILTERING CAPTURED PACKET INFO (NO, THERE'S NOT A JOKE HERE, SORRY.):

SO LET'S SWITCH IT UP. LET'S USE TCPDUMP TO FILTER DATA FROM THE PREVIOUS PACKET CAPTURE. COME ON. WE'RE ALMOST THERE.
LET'S USE THIS:

[SUDO TCPDUMP -NN -R CAPTURE.PCAP -V]

```
analyst@4e6daee2ace0:~$ sudo tcpdump -nn -r capture.pcap -v
reading from file capture.pcap, link-type EN10MB (Ethernet), snapsho
t length 262144
23:27:05.601202 IP (tos 0x0, ttl 64, id 35150, offset 0, flags [DF],
 proto TCP (6), length 60)
    172.17.0.2.51686 > 173.194.215.138.80: Flags [S], cksum 0x318f (
incorrect -> 0x132d), seq 3515786102, win 65320, options [mss 1420,s
ackOK,TS val 3997488881 ecr 0,nop,wscale 6], length 0
23:27:05.602121 IP (tos 0x0, ttl 126, id 0, offset 0, flags [DF], pr
oto TCP (6), length 60)
    173.194.215.138.80 > 172.17.0.2.51686: Flags [S.], cksum 0x0ff4
(correct), seq 1286347729, ack 3515786103, win 65535, options [mss 1
```

THIS WILL RUN TCPDUMP WITH THESE OPTIONS:

- -NN: DISABLE PORT AND PROTOCOL NAME LOOKUP.
- -R: READ CAPTURE DATA FROM THE NAMED FILE.
- -V: DISPLAY DETAILED PACKET DATA.
- YOU MUST SPECIFY THE -NN SWITCH AGAIN HERE, AS YOU WANT TO MAKE SURE TCPDUMP DOES NOT PERFORM NAME LOOKUPS OF EITHER IP ADDRESSES OR PORTS, SINCE THIS CAN ALERT THREAT ACTORS.

THIS RETURNS OUTPUT DATA SIMILAR TO THE FOLLOWING:

```
reading from file capture.pcap, link-type EN10MB (Ethernet), snapsho
t length 262144
23:27:05.601202 IP (tos 0x0, ttl 64, id 35150, offset 0, flags [DF],
 proto TCP (6), length 60)
    172.17.0.2.51686 > 173.194.215.138.80: Flags [S], cksum 0x318f (
```

JUST LIKE THE LAST OUTPUTS, YOU CAN SEE ALL THE INFORMATION INCLUDING THE TTL, THE PROTOCOL, THE LENGTH, ETC. PRETTY NEAT RIGHT?

NOW LET'S USE THE TCPDUMP COMMAND TO FILTER THE EXTENDED PACKET DATA FROM THE **CAPTURE.PCAP** CAPTURE FILE.

```
analyst@4e6daee2ace0:~$ sudo tcpdump -nn -r capture.pcap -X
reading from file capture.pcap, link-type EN10MB (Ethernet), snapsho
t length 262144
23:27:05.601202 IP 172.17.0.2.51686 > 173.194.215.138.80: Flags [S],
 seq 3515786102, win 65320, options [mss 1420,sackOK,TS val 39974888
81 ecr 0,nop,wscale 6], length 0
        0x0000:  4500 003c 894e 4000 4006 800d ac11 0002  E..<.N@.@.
......
        0x0010:  adc2 d78a c9e6 0050 d18e a376 0000 0000  .......P..
.v....
        0x0020:  a002 ff28 318f 0000 0204 058c 0402 080a  ...(1.....
......
        0x0030:  ee44 d6f1 0000 0000 0103 0306            .D........
```

THIS COMMAND WILL RUN TCPDUMP WITH THE FOLLOWING OPTIONS:

- -NN: DISABLE PORT AND PROTOCOL NAME LOOKUP.
- -R: READ CAPTURE DATA FROM THE NAMED FILE.
- -X: DISPLAY THE HEXADECIMAL AND ASCII OUTPUT FORMAT PACKET DATA. SECURITY ANALYSTS CAN ANALYZE HEXADECIMAL AND ASCII OUTPUT TO DETECT PATTERNS OR ANOMALIES DURING MALWARE ANALYSIS OR FORENSIC ANALYSIS.
- HEXADECIMAL, ALSO KNOWN AS HEX OR BASE 16, USES 16 SYMBOLS TO REPRESENT VALUES, INCLUDING THE DIGITS 0-9 AND LETTERS A, B, C, D, E, AND F. AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII) IS A CHARACTER ENCODING STANDARD THAT USES A SET OF CHARACTERS TO REPRESENT TEXT IN DIGITAL FORM.

CONCLUSION:

AFTER GIVING MY WORK TO THE LEAD AND CLOCKING OUT FOR THE DAY, I FELT PROUD OF MYSELF BECAUSE OF WHAT I LEARNED. TODAY, I LEARNED HOW TO IDENTIFY NETWORK INTERFACES, USED THE TCPDUMP COMMAND TO CAPTURE NETWORK DATA FOR INSPECTION, INTERPRET AND INSPECT THE DATA GIVEN, AND I ALSO LEARNED HOW TO SAVE AND LOAD THE PACKET I CAPTURED. NOW THAT SOUNDS LIKE AN AMAZING DAY TO ME.

IF YOU ENJOYED THIS LAB, CLICK THE LINK AND SHOOT ME A FOLLOW OR A MESSAGE ON LINKEDIN. I'M HAPPY TO COLLABORATE AND WORK. THANK YOU FOR READING.