**Introduction You have been hired by a company as a network analyst. Your first assignment is to begin tuning an IDS sensor running snort. You have been provided a number of example packet captures for review. Some of these captures will trigger alerts for your analysis, others will require you to create signatures to match the traffic patterns. Below is the detail of what you are expected to complete and how you will be assessed. Project package is stored on our course server at /usr/local/src/proj3/347-proj3.tar.gz**

**Group 1 Captures - Port Scans (complete 1 of 2) - 20pts each The two captures in this group are port scans of different kinds. Configure and run snort to identify the type of port scan involved. Then review the packet capture itself to identify its characteristics, specifically the number of source IP addresses of the scanning, the ports involved, the success of the scanning and the rate (packets over time). Submission criteria:**

1. Text of the alert from snort

```
# Portscan detection.  For more information, see README.sfportscan
preprocessor sfportscan: proto  { all } memcap { 10000000 } sense_level { high }
```

In order for the  port scan to work I had to change the **sense level of the built-in portscan capability in snort from low to high and make sure it was uncommented**. Sense level being low won't detect a lot of port scans, while medium and high detects more.

```
stran5@course_server_347_447:~/proj-03/myLogs$ snort -c ./snort.conf -A full -k
none -l ./myLogs/ -r group-01/capture-01.pcap
```

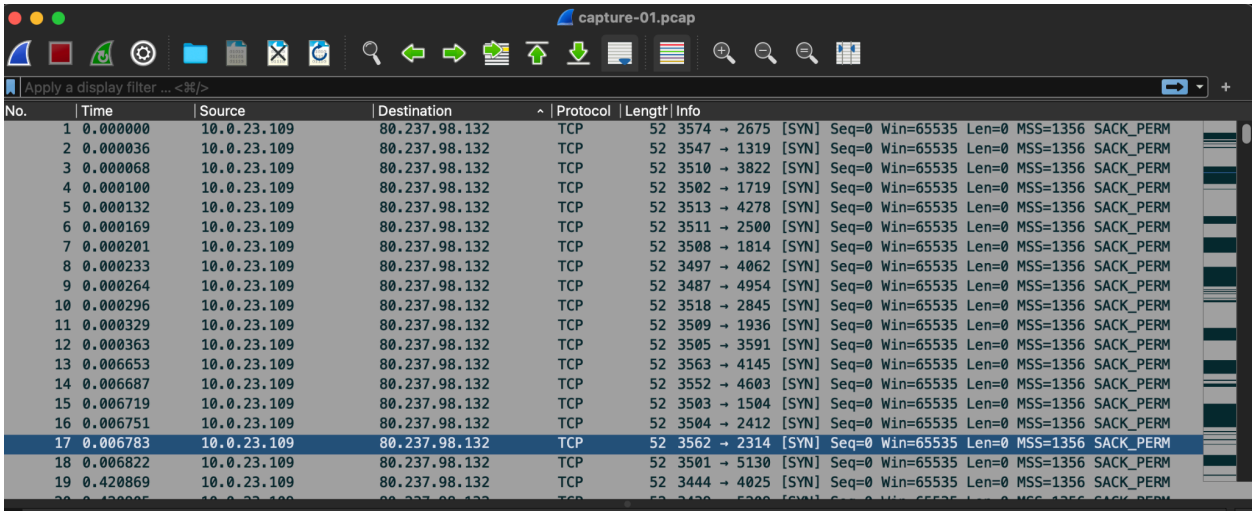
Ran snort on **Capture-01**

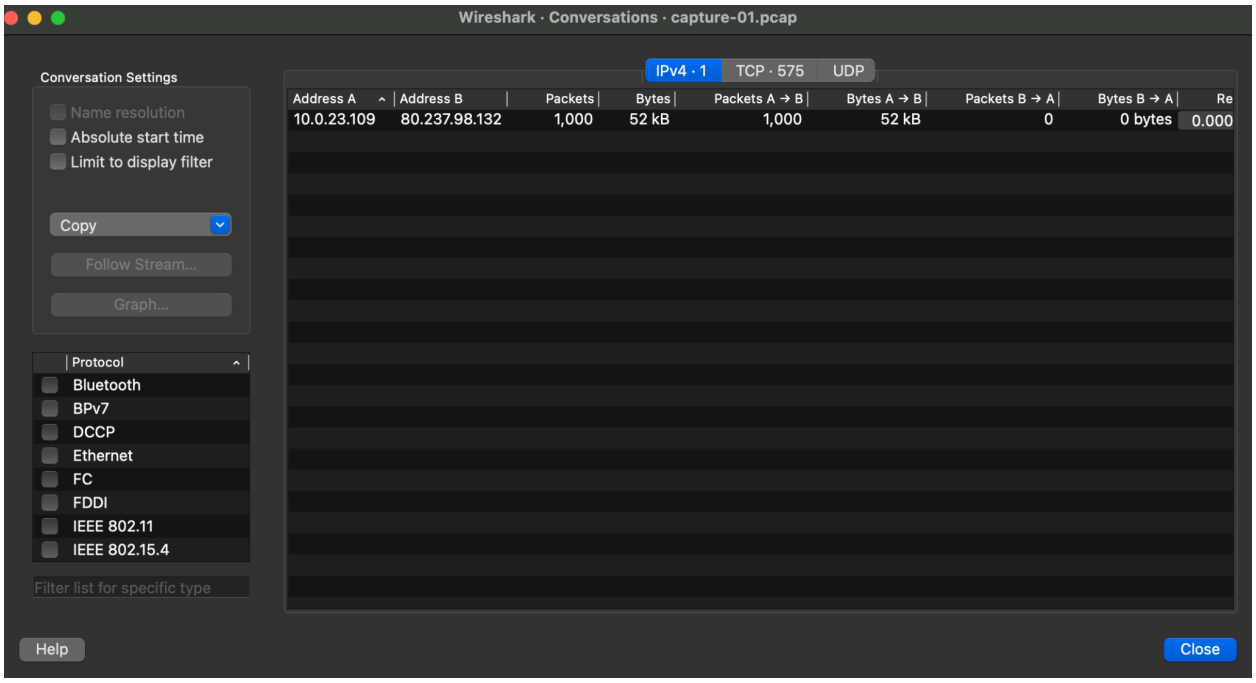
Text alert:

```
stran5@course_server_347_447:~/proj-03/myLogs$ cat alert-v1
[**] [122:5:1] (portscan) TCP Filtered Portscan [**]
[Classification: Attempted Information Leak] [Priority: 2]
09/25-14:42:13.106431 10.0.23.109 -> 80.237.98.132
PROTO:255 TTL:128 TOS:0x0 ID:6017 IpLen:20 DgmLen:165 DF
```

**TCP Filtered Portscan, Attempted Information Leak**

2. Characterization of the packet capture (packets-per second, randomness of ports, source and destination IPs involved) Packet capture content screenshot below to see and analyze the port numbers.




Statistics → Conversations to see ip addresses involved

- Source IP involved: **10.0.23.109**
- Destination IP involved: **80.237.98.132**
- Based on the packets of the TCP SYN scans on a variety of ports, the ports are **randomized**.
- Some of the specific **source ports** are: **3574, 3547, 3510**. They are randomized there isn't really a pattern

- Some of the specific **destination ports** are: **2675, 1319, 3822**. Same with the source ports they are randomized
- Based on these specific ports there are only SYN scan packets and no ACK, which means there is no **return traffic/response** and verifies that the **port scan is a filtered scan.**


Packets-per second: Average is **25.7** packets per second. Went to statistics and looked at the capture file properties to find the pps.

**Group 2 Capture - Malformed Packets (complete 1 of 1) - 25pts The capture in this group consists of malformed packets. Run the snort tool on this capture and interpret the results. This will involve the triggered alert(s) as well as analysis of the capture itself. You should consider the number of packets involved and the duration of the event. Submission criteria:**

1. Text of the alert from snort

```
include $RULE_PATH/malware-backdoor.rules
include $RULE_PATH/malware-cnc.rules
include $RULE_PATH/malware-other.rules
include $RULE_PATH/malware-tools.rules
```

I had to make sure the built in **malware detection rules** in snort were uncommented. My goal for this capture is to look at the **malformed packets**.

```
stran5@course_server_347_447:~/proj-03$ snort -c ./snort.conf -A full -k none -l
 ./myLogs/ -r group-02/capture-03.pcap
```

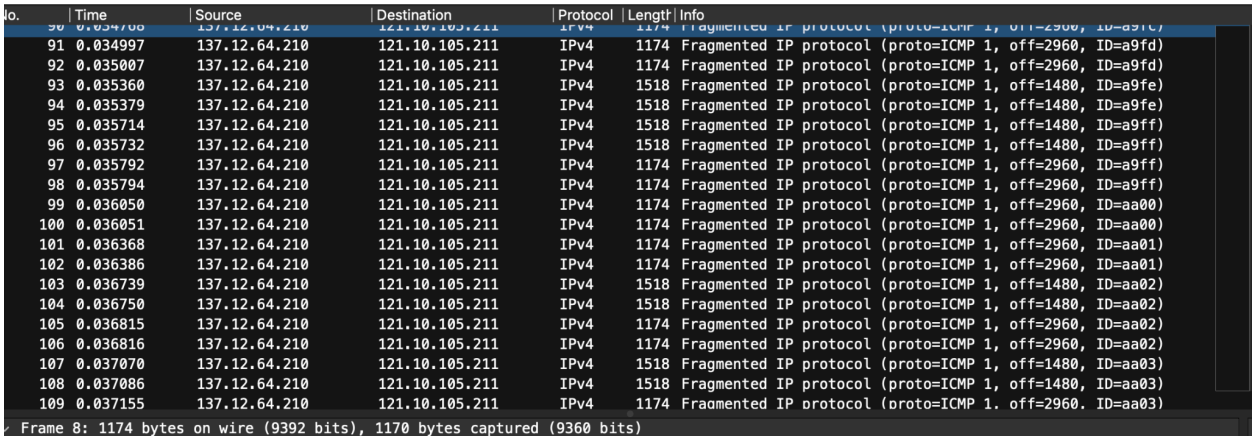
Ran snort on **Capture-03**

Text alert:

```
[stran5@course_server_347_447:~/proj-03/myLogs$ cat alert-v1 | more
[**] [123:8:2] (spp_frag3) Fragmentation overlap [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
05/06-20:40:19.468485 137.12.64.210 -> 121.10.105.211
ICMP TTL:61 TOS:0x0 ID:43466 IpLen:20 DgmLen:1156
Frag Offset: 0x0172   Frag Size: 0x0470

[**] [123:8:2] (spp_frag3) Fragmentation overlap [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
05/06-20:40:19.470211 137.12.64.210 -> 121.10.105.211
ICMP TTL:61 TOS:0x0 ID:43471 IpLen:20 DgmLen:1500 MF
Frag Offset: 0x00B9   Frag Size: 0x05C8
```

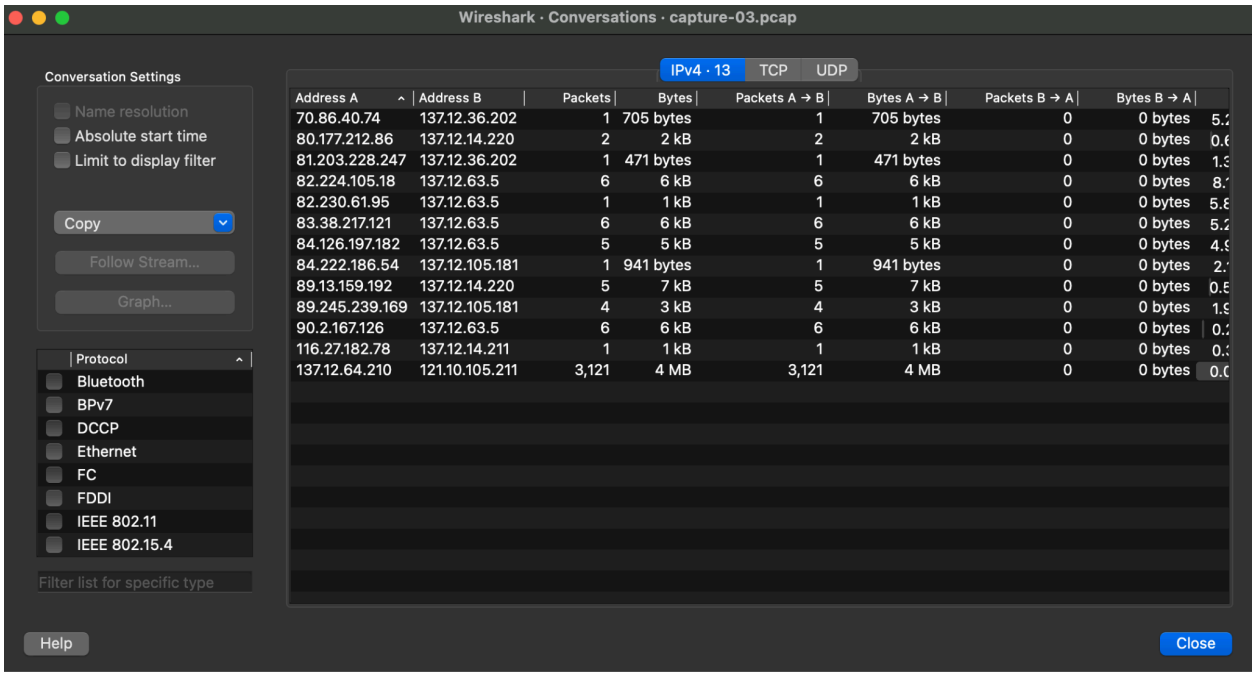
**Fragmentation overlap, Generic Protocol command Decode**

2. Characterization of the packet capture (packets-per second, randomness of ports, source and destination IPs involved) Packet capture contents screenshot below:



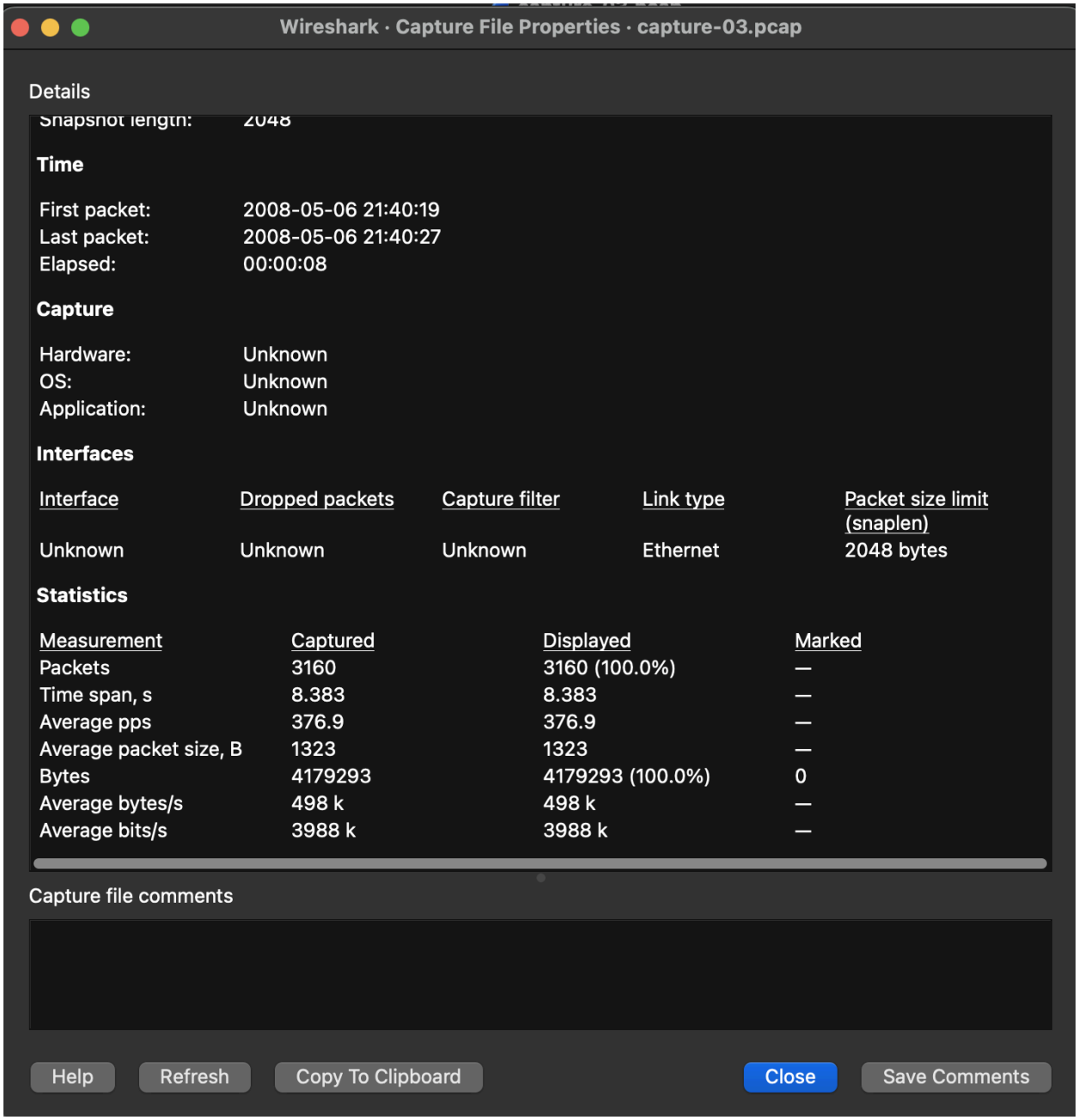
| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 91 | 0.034997 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1174 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=a9fd) |
| 92 | 0.035007 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1174 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=a9fd) |
| 93 | 0.035360 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1518 | Fragmented IP protocol (proto=ICMP 1, off=1480, ID=a9fe) |
| 94 | 0.035379 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1518 | Fragmented IP protocol (proto=ICMP 1, off=1480, ID=a9fe) |
| 95 | 0.035714 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1518 | Fragmented IP protocol (proto=ICMP 1, off=1480, ID=a9ff) |
| 96 | 0.035732 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1518 | Fragmented IP protocol (proto=ICMP 1, off=1480, ID=a9ff) |
| 97 | 0.035792 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1174 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=a9ff) |
| 98 | 0.035794 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1174 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=a9ff) |
| 99 | 0.036050 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1174 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=aa00) |
| 100 | 0.036051 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1174 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=aa00) |
| 101 | 0.036368 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1174 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=aa01) |
| 102 | 0.036386 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1174 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=aa01) |
| 103 | 0.036739 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1518 | Fragmented IP protocol (proto=ICMP 1, off=1480, ID=aa02) |
| 104 | 0.036750 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1518 | Fragmented IP protocol (proto=ICMP 1, off=1480, ID=aa02) |
| 105 | 0.036815 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1174 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=aa02) |
| 106 | 0.036816 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1174 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=aa02) |
| 107 | 0.037070 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1518 | Fragmented IP protocol (proto=ICMP 1, off=1480, ID=aa03) |
| 108 | 0.037086 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1518 | Fragmented IP protocol (proto=ICMP 1, off=1480, ID=aa03) |
| 109 | 0.037155 | 137.12.64.210 | 121.10.105.211 | IPv4 | 1174 | Fragmented IP protocol (proto=ICMP 1, off=2960, ID=aa03) |

Frame 8: 1174 bytes on wire (9392 bits), 1170 bytes captured (9360 bits)




Statistics → Conversations to find ip addresses involved

- Source IP involved: **70.86.40.74, 80.177.212.86, 81.203.228.247, 82.224.105.18, 82.230.61.95, 83.38.217.121, 84.126.197.182, 84.222.186.54, 89.13.159.192, 89.245.239.169, 90.2.167.126, 116.27.182.78, 137.12.64.210**
- Destination IP involved: **137.12.36.202, 137.12.14.220, 137.12.36.202, 137.12.63.5, 137.12.105.181, 137.12.14.211, 121.10.105.211**
- There are no ports in this capture because these are ICMP packets being sent. ICMP packets don't use ports.




Packets-per second: Average is **376.9** packets per second. Went to statistics and looked at the capture file properties to find the pps.

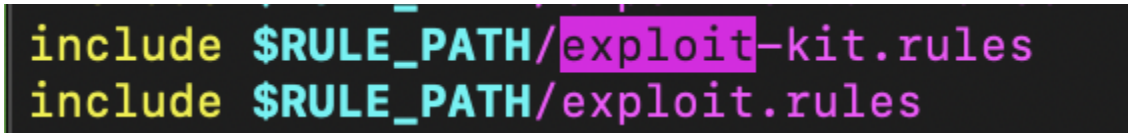
3. Your summary of what is happening in the capture

Based on this capture there are a lot of source ip's, destination ip and port numbers. This represents a high volume of network activity. This high volume of network activity can either represent a distributed denial-of-service (DDoS) or another malicious activity that is overwhelming the system. Since the system is being bombarded with network

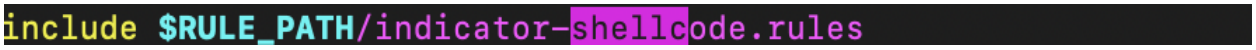
traffic, tasks that involve using the network will be slower and some important data can be lost.

**Group 3 Capture - Malicious Payload (complete 1 of 2) - 25pts The capture in this group has a malicious payload. Run the snort tool on the capture and interpret the results. This will involve the triggered alert(s) as well as analysis of the capture itself. With some knowledge of what you are up against, use wireshark to find the malicious payload and take a screenshot of what you found. For one of the captures, the payload will be a file, for the other it will be a lie told in public. Submission criteria:**

1. Text of the alert from snort

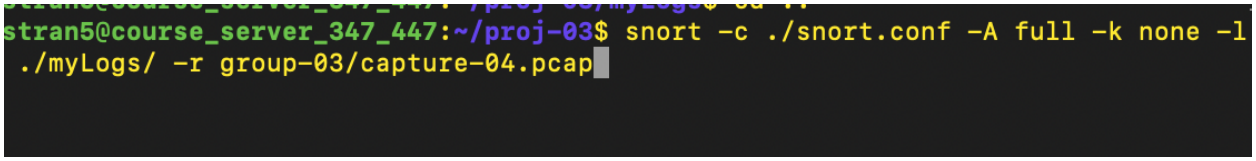



In snort I had to make sure the exploit built-in rule was uncommented




I also had to make sure the shellcode built-in rule was uncommented.

Both the shellcode and the exploit rules affect the payload. Exploits are known attacks on the payload, and the payload is the shellcode. So, that is why these rules are crucial for detecting malicious payload.




Ran snort on **Capture-04**

Text alert:

```
stran5@course_server_347_447:~/proj-03/myLogs$ cat alert-v1
[**] [123:3:2] (spp_frag3) Short fragment, possible DoS attempt [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
09/08-22:11:26.616090 10.1.1.1 -> 129.111.30.27
UDP TTL:64 TOS:0x0 ID:242 IpLen:20 DgmLen:56 MF
Frag Offset: 0x0000   Frag Size: 0x0024

[**] [123:5:2] (spp_frag3) Zero-byte fragment packet [**]
[Classification: Attempted Denial of Service] [Priority: 2]
09/08-22:11:26.616445 10.1.1.1 -> 129.111.30.27
UDP TTL:64 TOS:0x0 ID:242 IpLen:20 DgmLen:24
Frag Offset: 0x0003   Frag Size: 0x0004

stran5@course_server_347_447:~/proj-03/myLogs$
```

**Short fragment, possible DOS attempt, Generic Protocol Command Decode**
**Zero-byte fragment packet, Attempted Denial of Service**

2. Research as to the meaning of the alert

**Short fragment, possible DOS attempt, Generic Protocol Command Decode:** This alert means an IP fragmentation attack. It is a type of DOS attack. The packets break up data into multiple fragmented packets that fit within the MTU (Maximum Transmission Unit). So, a malicious user can send these broken up data packets, which makes it really hard for the intended/regular user to connect to the network and do tasks that involve it. One example is websites being unable to load for the intended user.

**Zero-byte fragment packet, Attempted Denial of Service:** This is a fragmentation attack as well. It's mostly the same as the first alert, but this one the packets have no data payload. That is why it is called zero-byte. Even though there is no payload data for these packets, they still require processing from the network. The processing can leak and show the weaknesses in the system, which lets malicious users know how to exploit the system.

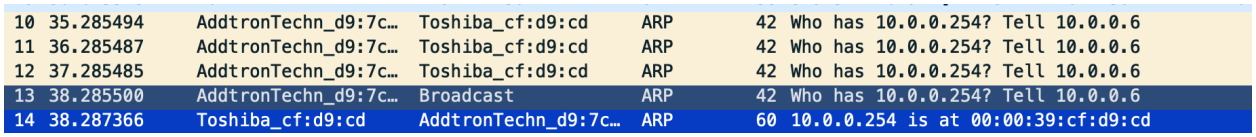
3. Identification of the malicious payload

```
# ARP spoof detection.  For more information, see the Snort Manual - Config
  Snort - Preprocessors - ARP Spoof Preprocessor
preprocessor arpspoof
# preprocessor arpspoof_detect_host: 192.168.40.1 f0:0f:00:f0:0f:00
preprocessor arpspoof_detect_host: 10.0.0.6 00:00:39:cf:d9:cd
```

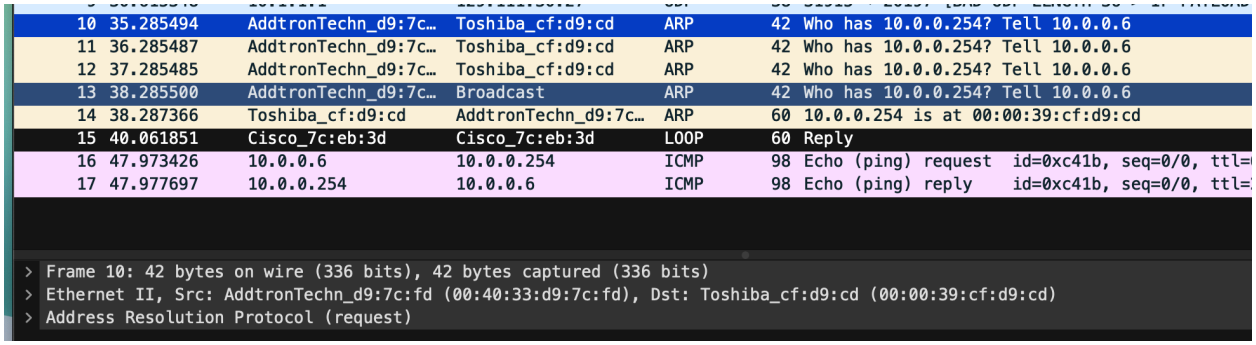
Had to edit the snort.conf rule. Had to add the line preprocessor **arp_spoof_detect_host: 10.0.0.6 00:00:39:cf:d9:cd**. Had to uncomment **preprocessor arpspoof**. Had to run snort again with those added.

```
[**] [112:4:1] (spp_arpspoof) Attempted ARP cache overwrite attack [**]
09/08-22:11:31.286591

[**] [112:4:1] (spp_arpspoof) Attempted ARP cache overwrite attack [**]
09/08-22:11:32.286584

[**] [112:4:1] (spp_arpspoof) Attempted ARP cache overwrite attack [**]
09/08-22:11:33.286582

[**] [112:4:1] (spp_arpspoof) Attempted ARP cache overwrite attack [**]
09/08-22:11:34.286597
```

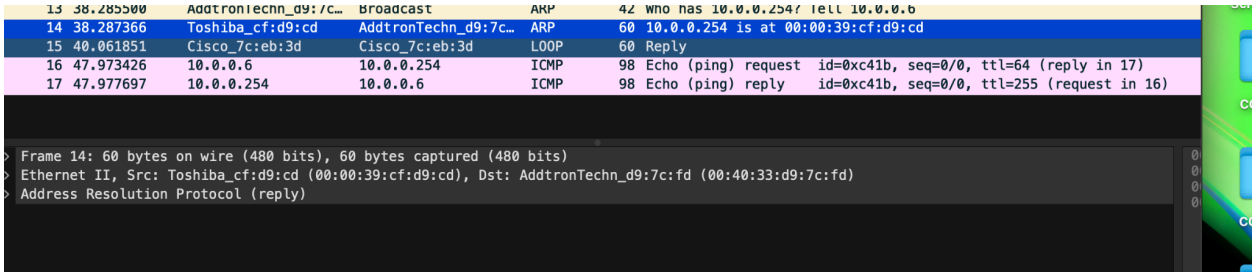
Alert based on my snort.conf file edit was **Attempted ARP cache overwrite attack**. This is an arp spoof attack, which is when an attacker sends a spoofed mac address to a local network to associate with the host ip. The host thinks it's legitimate and uses that mac address, which causes the traffic to be sent to the attacker and not the actual host itself..




The host **10.0.0.6** gets asked repeatedly who has the mac address of **10.0.0.254**




The “who has” packet shows the malicious mac address of **00:40:33:d9:7c:fd**, and its malicious because the “who has” keeps popping up.




Finally the last ARP packet has the correct reply with the legitimate mac address of **00:00:39:cf:d9:cd**

The **Malicious Payload** is the spoofed mac address of **00:40:33:d9:7c:fd**. It is a lie told to the public because the host thinks that **00:40:33:d9:7c:fd**

**Group 4 Captures - Writing Signatures (complete 2 of 2) - 40pts each The two captures in this group are non-malicious traffic, but the company wants to create alerts when certain events happen. The following descriptions provide details on how to write the custom signatures to trigger on these events which occur in specific packets noted in the capture names. Please be mindful of ensuring that all conditions are met for an alert and that the correct number of events trigger alerts. capture-06-p01.pcap (packet #1 in this capture is what you need to match on) For this capture, the company wishes to see when a particular file is written to a location using the TFTP protocol. Your signature should filter on the following with respect to packet 1 of the capture:**

**1. Protocol: UDP**
**2. Destination IP of the TFTP server: 192.168.0.13**
**3. Destination port: 69**
**4. Alert text: “Capture 6 Write Requested”**
**5. Payload content: (HEX encoded)0x0002(ASCII encoded)rfc1350.txt**
**6. SID: 10005450**
**7. REV: 1**

**Submission criteria:**

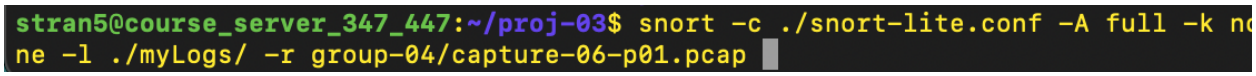
1. Text of the signature you created

```
# include $SO_RULE_PATH/server-other.rules
# include $SO_RULE_PATH/server-webapp.rules
include $HOME_DIR/proj-03/myRules/personal-rulesPG.rules
# Event thresholding or suppression commands. See threshold.conf
include /etc/snort/threshold.conf
"snort-lite.conf" 720L, 30667B                                718,1         Bot
```

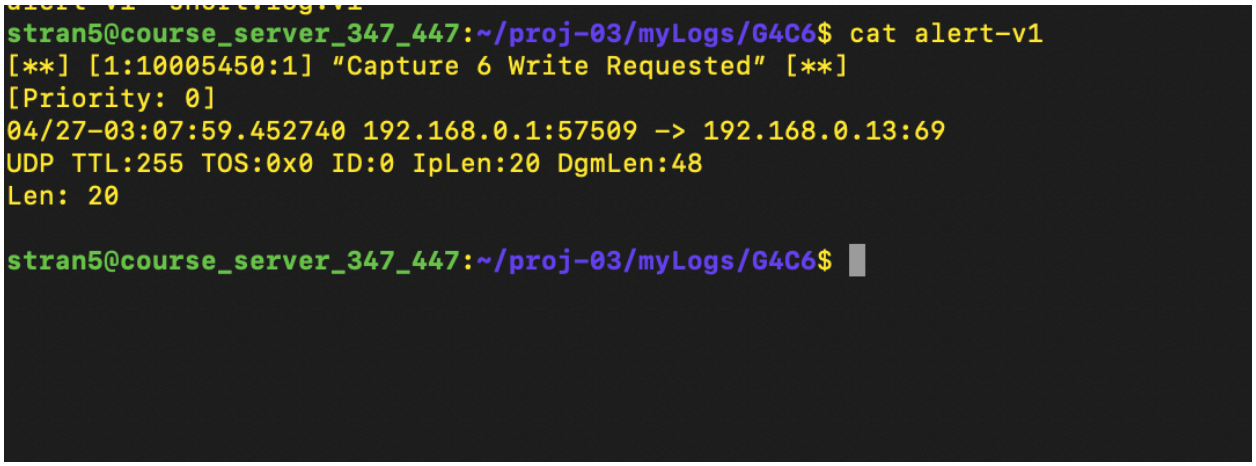
Had to make sure the snort-lite.conf file had my custom rules activated


This is my text signature based on the signature criteria given in the question.

2. Text of the alert from snort




Ran snort-lite on **Capture-06**




This is my text alert from snort based on the rules I made.

**capture-07-p08.pcap (packet #8 in this capture is what you need to match on) For this capture the company is interested in identifying when their backup**

**authentication service ID is used to successfully log into one of their systems. Since they use RADIUS authentication for all logins, we have a capture of events related to authentication. Your signature should trigger once for this capture, if it is configured as follows:**

**1. Protocol: UDP**
**2. Source port: 1812**
**3. Alert text: “Capture 7 RADIUS Auth Passed”**
**4. Payload content (1): (HEX encoded) 0x02 located in position 1 of the packet payload (indicating passed authentication)**
**5. Payload content (2): (ASCII encoded) “steve” which represents the username of the backup account**
**6. SID: 10005452**
**7. REV: 1**

**Submission criteria:**

1. Text of the signature you created

```
alert udp any 1812 -> any any \
( msg: "Capture 7 RADIUS Auth Passed"; \
  content: "|02|"; \
  content: "steve"; \
  sid: 10005452; rev: 1; )
~
~
~
~
~
~
~
~
~
~
~
~
"myRules/personal-rulesPG.rules" 11L, 319B                    8,42-38       All
```

This is my text signature based on the signature criteria given in the question.

2. Text of the alert from snort

```
stran5@course_server_347_447:~/proj-03/myLogs/G4C7$ cat alert-v1
[**] [1:10005452:1] "Capture 7 RADIUS Auth Passed" content: "|02|" [**]
[Priority: 0]
08/24-14:23:59.948233 127.0.0.1:1812 -> 127.0.0.1:65443
UDP TTL:64 TOS:0x0 ID:64337 IpLen:20 DgmLen:130
Len: 102

stran5@course_server_347_447:~/proj-03/myLogs/G4C7$
```

This is my text alert from snort based on the rules I made.