

Yamin Mousselli  
CS 8803: Cybersecurity Operations

## Snort Write Up

Some Snort Resources I used:

- [http://commons.oreilly.com/wiki/index.php/Snort\\_Cookbook/Rules\\_and\\_Signatures](http://commons.oreilly.com/wiki/index.php/Snort_Cookbook/Rules_and_Signatures)
- [http://snort.datanerds.net/writing\\_snort\\_rules.htm](http://snort.datanerds.net/writing_snort_rules.htm)
- <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node27.html>

Snort is light-weight and it's used to detect or drop specific byte sequences within a packet. Instead of investigating the content of every packet in Wireshark, IDS signatures are quick to search a ton of packets in a very short period of time. Moreover, IDS signatures are useful for things such as alerting a system when something weird pops up in a packet or dropping the packet(s) all-together if malicious strings are detected within packets. If Snort drops a packet, then it's preventing something from happening and thus an IPS. Otherwise, Snort is detecting and thus an IDS. Snort is technically an IDS/IPS. More specifically, Snort is either an IDS or IPS based on its configuration within an organization's network.

Most Snort lines are one-liners. Snort rules are divided into two logical sections: the rule header and the rule options. The rule header contains the rule's action (alert or drop), protocol (tcp, ip, udp), source and destination IP addresses, and the source and destinations ports. The rule header is everything before the parenthesis. The rule options are not required by any rule, are contained in the parenthesis, and are used to create tighter definitions for the rule to alert, collect, or drop packets.

There is a rules file and within that, there are #includes. This keyword is similar to the one in C where you will include the contents of the files that you have included. Another similarity to programming is that you can define variables and use those variables. Variables are written in all-uppercase similar to when writing constants in Java or C. For example, you can define these variables to be IP addresses.

- You would use the include keyword like this: include: <include file path/name>
- Make sure you put a semi-colon at the end of the signature revision.

Note: For user-defined rules, the signature id (sid) needs to be greater than 1,000,000. The sid is unique identifies the signature. Revisions keep track of signature modifications.

### Things to know for VIM

- To go back:
  - Press Escape
  - Press :
  - Type q! (! forces it)
- To insert, press i.
- To move at the beginning or end of file, use Shift ^ or Shift \$, respectively.
- To save, use : w.
- You can enter wq! Which would save and quit in that order.

1)

**Signature:**

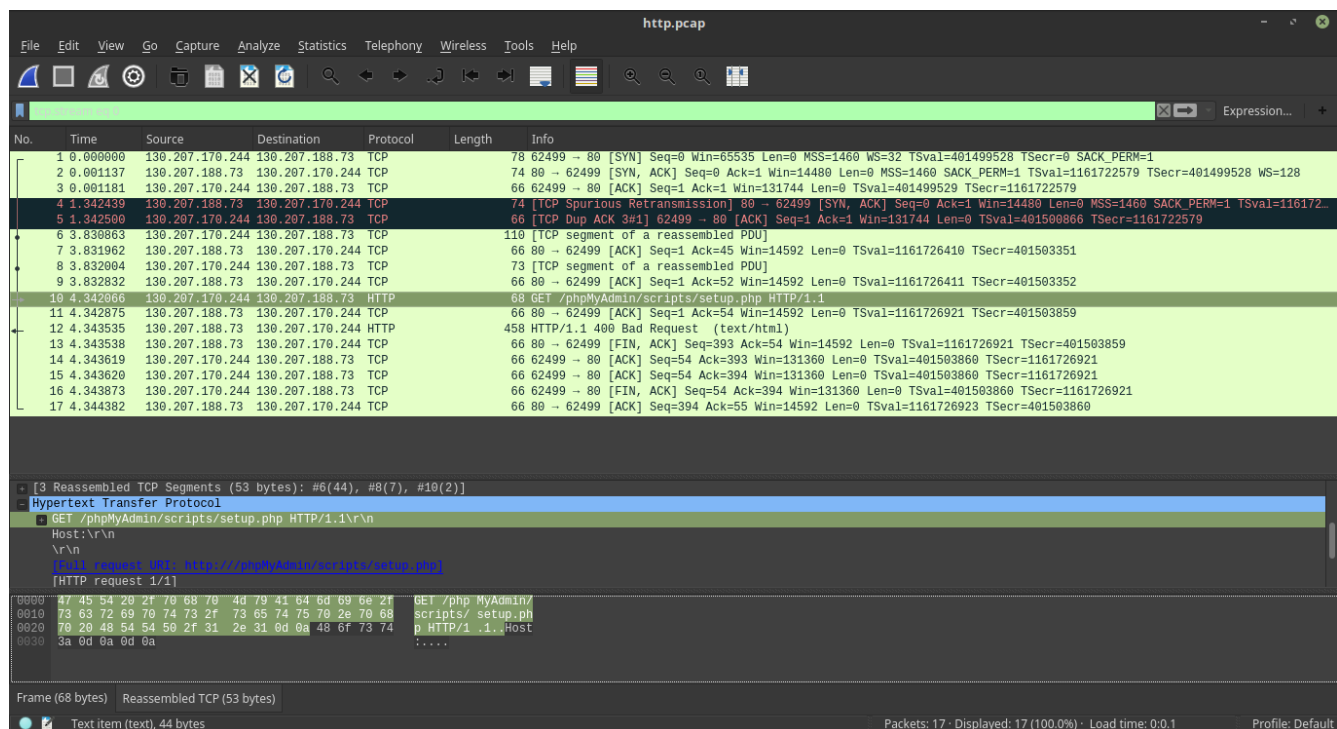
alert tcp 10.10.27.1 any → \$HOME\_NET \$HTTP\_PORTS (msg:"This is the malicious actor targeting our web services. "; sid:1000001; rev:1;)

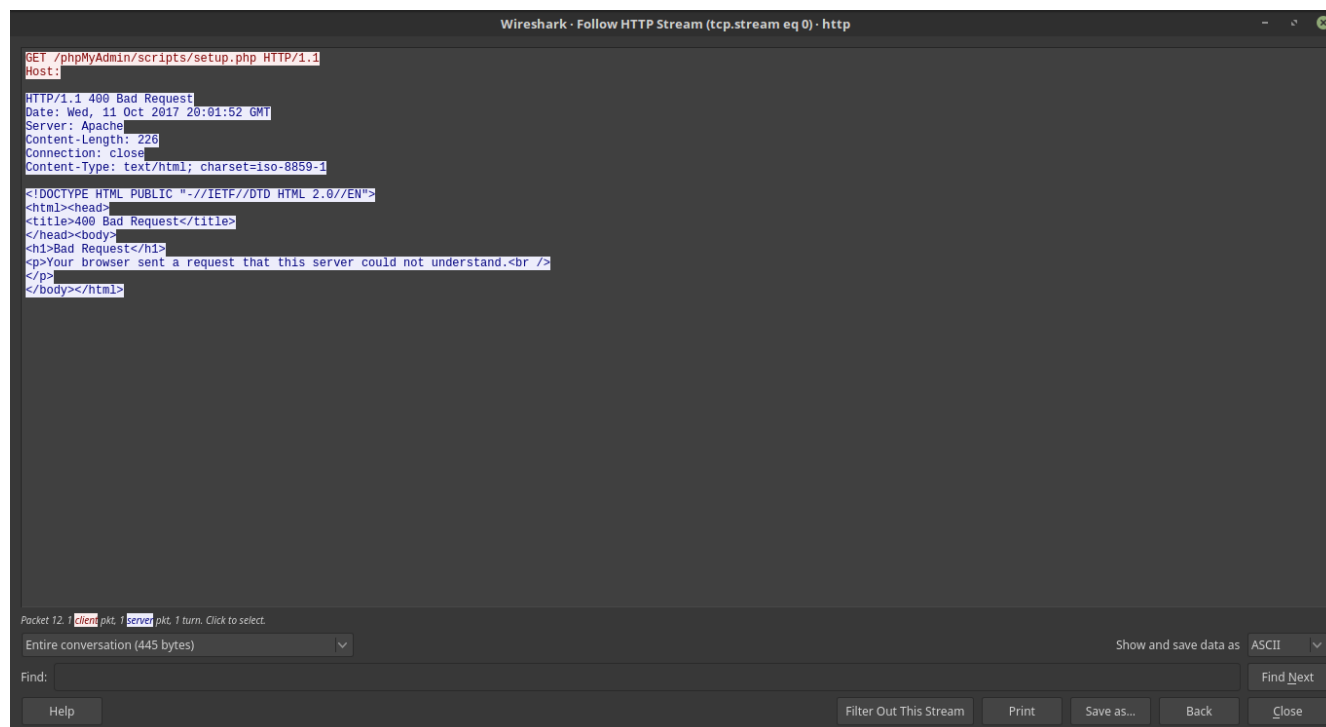
2)

Note: You can inspect the user agent within the network tab in the developer tools of a web browser. A user agent will normally be set when you use a web browser to surf the Internet. You can access the Internet by using a script in which the user agent will not be set.

Note: Word applications have different arrows and quotation marks than Vim.

When I open Wireshark to inspect http.pcap, I see the GET request with the URI “/php/myMyAdmin/scripts/setup/php” and when I follow the HTTP stream, I do not see a user agent. I also see bad request. Additionally, there isn’t a host for the GET packet with that URI. This tells me something is wrong. If there isn’t a host defined for the packet, then it’s a bad request. The image below reflect my findings.





Additionally, we are going to alert on our web servers so the right hand side of the signature needs HOME\_NET on all HTTP ports since we saw the URI was associated with a GET request. We need to alert on the URI and GET requests that don't have a user agent set. Therefore, we need to use the http\_uri and http\_header.

Note that the exclamation point goes outside the quotation marks when defining the content for http\_header in the rules header.

### Signature:

```
alert tcp $EXTERNAL_NET any → $HOME_NET $HTTP_PORTS (msg:"malicious actors are scanning the uri that we are looking for that aren't associated with a user agent"; content:"/phpMyAdmin/scripts/setup.php"; http_uri; content:!"User-Agent"; http_header; sid:1000002; rev:1;)
```

I saved this signature in my local.rules file. If we execute `snort --help` to look at our options, we can see that we need to use "-r http.pcap" when executing the signature on the pcap file (the -r option is for the tcpdump file). Additionally, we need to use the -k flag because we don't want any checksum validation so we use "-k none". We also need to use the -c option to specify the configuration files (rules file) with our signature. The configuration file will look for the rules file.

```
"snort -c /etc/snort.conf -k none -r http.pcap"
```

When opening Wireshark, there was only packet with the specified URI. Therefore, we only needed 1 alert. Below is a picture of the alert. This signature is successful.

```

root@yaman-Aspire-E5-571 /home/yaman/Documents/The_Works/Georgia_Tech/cs/cs8803/hws/hw5
File Edit View Search Terminal Help

IP6 Disc: 0 ( 0.000%)
TCP Disc: 0 ( 0.000%)
UDP Disc: 0 ( 0.000%)
ICMP Disc: 0 ( 0.000%)
All Discard: 0 ( 0.000%)
Other: 0 ( 0.000%)
Bad Chk Sum: 0 ( 0.000%)
Bad TTL: 0 ( 0.000%)
SS G 1: 0 ( 0.000%)
SS G 2: 0 ( 0.000%)
Total: 17

Action Stats:
Alerts: 1 ( 5.882%)
Logged: 1 ( 5.882%)
Passed: 0 ( 0.000%)

Limits:
Match: 0
Queue: 0
Log: 0
Event: 0
Alert: 0

Verdicts:
Allow: 17 (100.000%)
Block: 0 ( 0.000%)
Replace: 0 ( 0.000%)
Whitelist: 0 ( 0.000%)
Blacklist: 0 ( 0.000%)
Ignore: 0 ( 0.000%)
Retry: 0 ( 0.000%)

Frag3 statistics:
Total Fragments: 0
Frgs Reassembled: 0
Discards: 0
Memory Faults: 0
Timeouts: 0
Overlaps: 0
Anomalies: 0
Alerts: 0
Drops: 0
FragTrackers Added: 0
FragTrackers Dumped: 0
FragTrackers Auto Freed: 0
Frag Nodes Inserted: 0

Additionally, we are going to alert on our web servers to the right hand side of the signature
needs to HOME_NET on all HTTP ports since we saw the URI was associated with a GET request. We
will need to alert on the URI and the fact that the GET request doesn't have a user agent. Therefore, we
will use the following rule:

None that the replacement point goes outside the quotation marks when defining the content for
http_header in the rules header.

***Make sure that you don't copy signatures from a word editor into vim because the quotation marks
are different and also arrow

2) alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"malicious action are
scanning the uri that we are looking for that aren't associated with a user agent",
content:"phpMyAdminscript=script.php", http_uri: content:"User-Agent", http_header: uid:1000002,
rev:1)

I have saved this signature in my local rules file. If we go to smart-help, we can see that we need to use
the "i http_req" when executing the signature on the pcap file (-i is for reading file). Additionally,
checksum validation as we use "-k none" for no
checksum. We also need to use "-c" to specify the configuration files (rules file) with our signature. The
config file will look for the rules file.

"smart -c /etc/snort.conf -k none -i http_req"

When opening Wireshark, there was only packet with the specified URI. Therefore, we only needed 1
alert. Below is a picture of the alert. This signature is a success.

```

3)

There is only one TCP stream within smtp.pcap because there is only 1 email. When we follow tcp stream, we can see the link that was sent to the victim. The image below shows the email contents.

```

Wireshark - Follow TCP Stream (tcp.stream eq 0) - smtp

220 mx.us.email.fireeyecloud.com IAD17 ESMTP server ready
EHLO tulkas.localdomain
250-mx.us.email.fireeyecloud.com hello [130.207.170.244], pleased to meet you
250-HELP
250-SIZE 1000000000
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-STARTTLS
250 OK
MAIL FROM:<chris@tulkas.localdomain> SIZE=1815
250 2.1.0 <chris@tulkas.localdomain> sender ok
RCPT TO:<christopher.craig@security.gatech.edu>
250 2.1.5 <christopher.craig@security.gatech.edu> recipient ok
DATA
354 OK
Received: by tulkas.localdomain (Postfix, from userid 501)
id 225892595CE4; Wed, 11 Oct 2017 15:58:38 -0400 (EDT)
From: Christopher Craig <christopher.craig@security.gatech.edu>
Content-Type: multipart/alternative;
boundary="Apple-Mail=EDFB57B5-12B0-4D34-9115-7EF6D0181E3E"
Mime-Version: 1.0 (Mac OS X Mail 10.3 (3273))
Subject: test
X-Universally-Unique-Identifier: 6926FE59-98E5-4070-950C-789740FBA49C
Message-Id: <F3FB5EB8-DEAD-4C70-80F9-73228057A9ED@security.gatech.edu>
Date: Wed, 11 Oct 2017 15:49:10 -0400
To: Christopher Craig <christopher.craig@security.gatech.edu>

--Apple-Mail=EDFB57B5-12B0-4D34-9115-7EF6D0181E3E
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain;
charset=us-ascii

Click here <https://accounts.google.com/o/oauth2/auth> to have bad =
things happen=20

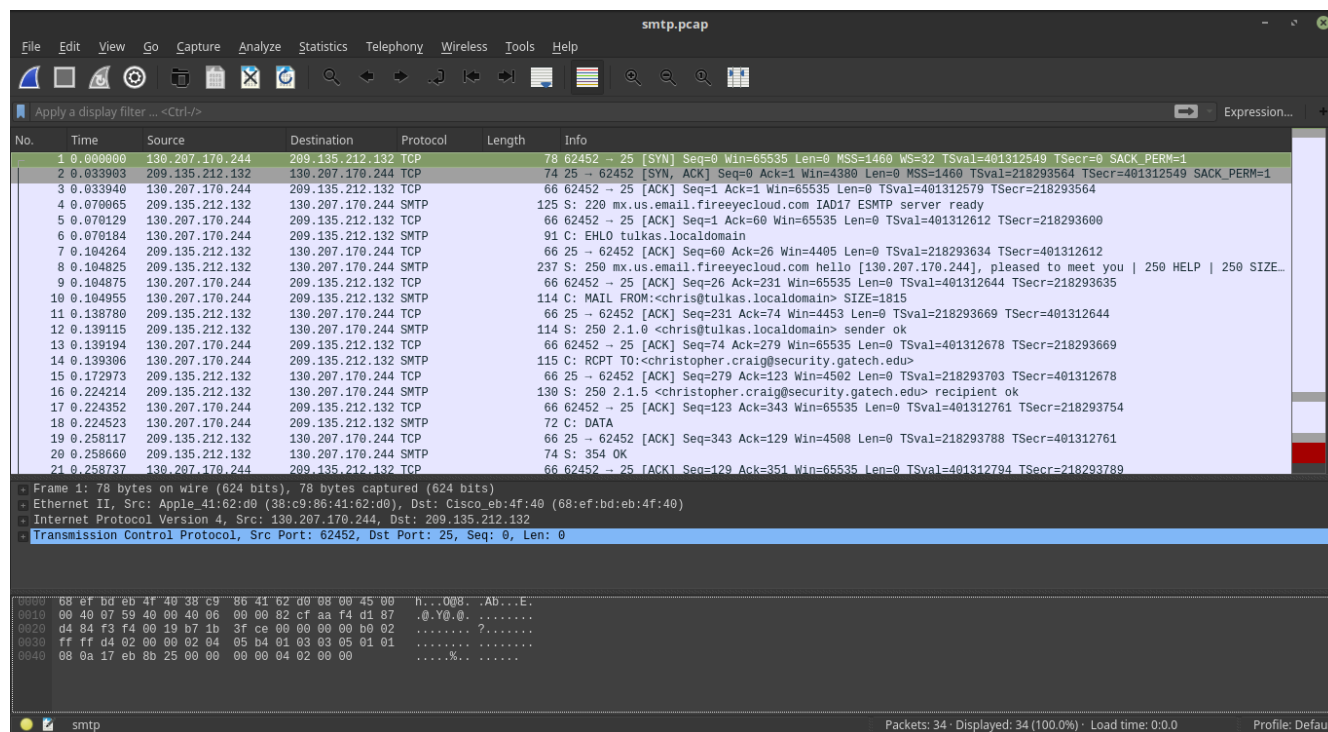
--Apple-Mail=EDFB57B5-12B0-4D34-9115-7EF6D0181E3E

Packet 22. 10 client pkts, 8 server pkts, 15 turns. Click to select.
Entire conversation (2427 bytes)
Show and save data as ASCII
Find:
Filter Out This Stream Print Save as... Back Close

```

The content we are looking for within the pcap can be found when we follow TCP stream. We only need to put the link in the content. Also SMTP doesn't have a header specific for it. We don't need to specify anything like http\_header or http\_uri options because we are searching the content of the email.

Moreover, we need to see which ports to watch both ingress and egress traffic on. Always go to the first packet in Wireshark to see both source and destination ports (the source port will be the client). The SMTP server uses the well-known port 25. The image below reflects my finding. The EXTERNAL\_NET variable defines malicious actors because they are the outbound connections and the HOME\_NET variable is the server because it is what we are protecting (Think of it as us monitoring inbound connections).



### Signature:

alert tcp \$EXTERNAL\_NET any → \$HOME\_NET 25 (msg:"google account users are adding malicious services"; content:"<https://accounts.google.com/o/oauth2/auth>"; sid:1000003; rev:1;)

The signature alerted and logged three packets (there's a difference). The picture below reflects my findings.

```

root@yaman-Aspire-E5-571 /home/yaman/Documents/The_Works/Georgia_Tech/cs/cs8803/hws/hw5
File Edit View Search Terminal Help
-----
Total: 35
Action Stats:
  Alerts: 3 ( 8.571%)
  Logged: 3 ( 8.571%)
  Passed: 0 ( 0.000%)
Limits:
  Match: 0
  Queue: 0
  Log: 0
  Event: 0
  Alert: 0
Verdicts:
  Allow: 34 (100.000%)
  Block: 0 ( 0.000%)
  Replace: 0 ( 0.000%)
  Whitelist: 0 ( 0.000%)
  Blacklist: 0 ( 0.000%)
  Ignore: 0 ( 0.000%)
  Retry: 0 ( 0.000%)
-----
Frag3 statistics:
  Total Fragments: 0
  Frags Reassembled: 0
  Discards: 0
  Memory Faults: 0
  Timeouts: 0
  Overlaps: 0
  Anomalies: 0
  Alerts: 0
  Drops: 0
  FragTrackers Added: 0
  FragTrackers Dumped: 0
  FragTrackers Auto Freed: 0
  Frag Nodes Inserted: 0
  Frag Nodes Deleted: 0
-----
Stream statistics:
  Total sessions: 1
  TCP sessions: 1
  UDP sessions: 0
  ICMP sessions: 0
  IP sessions: 0
  TCP Prunes: 0

```

**Correction:** The arrow in this signature should be bidirectional because you need to check both *inbound* and outbound mail. I had only checked for outbound mail.

4)

I see that the client IP is 143.215.17.133 and the server IP is 66.196.114.194 within ymsg.pcap in Wireshark. I also see that the destination port is 5050. The image below reflects my findings.

The image shows a Wireshark packet capture of ymsg2.pcap. The packet list pane displays several packets, with packet 4 selected. The packet details pane shows the structure of the selected packet, which is a YMSG Authentication packet. The packet bytes pane shows the raw data of the packet.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.225944	143.215.17.133	66.196.114.194	YMSG	96	Authentication (status=Default)
5	0.273568	66.196.114.194	143.215.17.133	YMSG	172	Authentication (status=Server Ack)
6	0.647180	143.215.17.133	66.196.114.194	TCP	64	57033 -> 5050 [ACK] Seq=39 Ack=115 Win=65612 Len=0
7	1.934354	143.215.17.133	66.196.114.194	YMSG	816	Authentication Response (status=Unknown Status: 12)
8	1.974342	66.196.114.194	143.215.17.133	YMSG	337	List (status=Default)

Packet 4 details:

- Frame 4: 96 bytes on wire (768 bits), 96 bytes captured (768 bits)
- Ethernet II, Src: Cisco-eb:49:80 (68:ef:bd:eb:49:80), Dst: JuniperN-d0:29:c0 (dc:38:e1:d0:29:c0)
- 802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 2922
- Internet Protocol Version 4, Src: 143.215.17.133, Dst: 66.196.114.194
- Transmission Control Protocol, Src Port: 57033, Dst Port: 5050, Seq: 1, Ack: 1, Len: 38
- Yahoo YMSG Messenger Protocol (Authentication)
  - Version: 19
  - Vendor ID: 0
  - Packet Length: 18
  - Service: Authentication (87)
  - Status: Default (0)
  - Session ID: 0x00000000
  - Content: 31c08073757265647265346261626573c080

Packet 4 bytes:

```

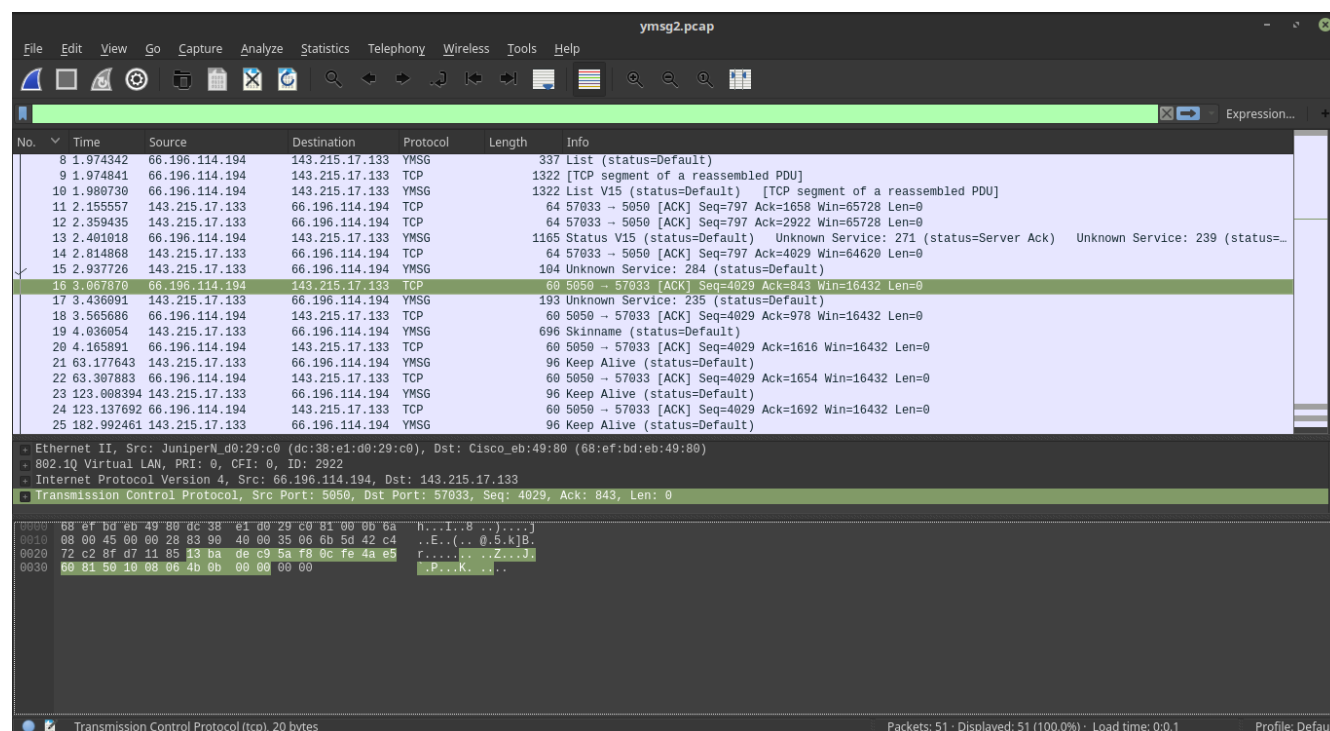
0000  dc 38 e1 d0 29 c0 68 ef bd eb 49 80 81 00 0b 6a  .8..).h. .I....]
0010  08 00 45 00 00 4e 02 a4 40 00 7e 06 a3 23 8f d7  ..E..N..@..#..
0020  11 85 42 c4 72 c2 0e c9 13 ba 4a e5 5d 37 5a f7  ..B.r...c.J]7Z.
0030  fd 42 50 18 40 39 bd 7f 00 00 59 4d 53 47 09 13  .BP.00...YMSG..
0040  00 00 00 12 60 57 00 00 00 00 00 00 00 31 c0  ....W.....1.
0050  80 73 75 72 65 64 72 65 34 62 61 62 65 73 c0 80  .suredre 4bases..

```

Through the link that was provided to us, I was able to figure out that the protocol number for the YAHOO\_SERVICE\_LIST is 0x55. The number encoding in Wireshark within the byte sequence is 00 55. We need to specify 2 bytes in our signature because the service number for the Yahoo messenger protocol is 2 bytes.

The offset is 10 bytes because in both the protocol and Wireshark, you can see the service number starts on the 11<sup>th</sup> byte. Therefore, you must start on the 11<sup>th</sup> byte. Offset always starts the beginning of the sequence. Depth specifies how far in the packet you should look. Depth should only be 2 bytes because the sequence number for the Yahoo messenger protocol is only 2 bytes long.

I also see that the server has an ACK which is weird. Usually, it's the client that sends the SYN but the server is asking the client for something which tells me the Yahoo Messenger Protocol is weird. This led me to have the bi-directional arrow in my signature because I need it to alert on both ends. The image below reflect my findings:



### Signature:

```
alert tcp $HOME_NET any <> $EXTERNAL_NET 5050 (msg:"This packet is a part of Yahoo-List Service"; content:"|00 55|"; offset:10; depth:2; sid:1000004; rev:1;)
```

My signature alerted on one packet. Below is a picture of my results:

```

root@yaman-Aspire-E5-571 /home/yaman/Documents/The_Works/Georgia_Tech/cs/cs8803/hws/hw5

File Edit View Search Terminal Help

55 G 2: 0 ( 0.000%)
Total: 51

-----
Action Stats:
Alerts: 1 ( 1.961%)
Logged: 1 ( 1.961%)
Passed: 0 ( 0.000%)

Limits:
Match: 0
Queue: 0
Log: 0
Event: 0
Alert: 0

Verdicts:
Allow: 51 (100.000%)
Block: 0 ( 0.000%)
Replace: 0 ( 0.000%)
Whitelist: 0 ( 0.000%)
Blacklist: 0 ( 0.000%)
Ignore: 0 ( 0.000%)
Retry: 0 ( 0.000%)

-----
Frag3 statistics:
Total Fragments: 0
Frag Reassembled: 0
Discards: 0
Memory Faults: 0
Timeouts: 0
Overlaps: 0
Anomalies: 0
Alerts: 0
Drops: 0
FragTrackers Added: 0
FragTrackers Dumped: 0
FragTrackers Auto Freed: 0
Frag Nodes Inserted: 0
Frag Nodes Deleted: 0

-----
Stream statistics:
Total sessions: 1
TCP sessions: 1
UDP sessions: 0
ICMP sessions: 0
IP sessions: 0

-----
Signature:
alert tcp $HOME_NET any <=> $EXTERNAL_NET 5050 (msg:"This packet is a part of Yahoo-List Service"; content:"[00 55]"; offset:10; depth:2; sid:1000004; rev:1;)

My signature alerted on one packet. Below is a picture of my results:

```

**Correction:** While the signature works, ports should be any b/c yahoo messenger could use other ports. Therefore, include content: "YMSG" within the signature

Below is a picture of all my signatures:

```

root@yaman-Aspire-E5-571 /home/yaman/Documents/The_Works/Georgia_Tech/cs/cs8803/hws/hw5

File Edit View Search Terminal Help

alert tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS (msg:"malicious actors are scanning the uri that we are looking for that aren't associated with a user agent"; content:"/phpMyAdmin/scripts/setup.php"; http_uri; content:! "User-Agent"; http_header; sid:1000002; rev:1;)

alert tcp $HOME_NET any -> $EXTERNAL_NET 25 (msg:"Google-account users are adding malicious services"; content:"https://accounts.google.com/o/oauth2/auth"; sid:1000003; rev:1;)

alert tcp $HOME_NET any <=> $EXTERNAL_NET 5050 (msg:"This packet is a part of Yahoo-List Service"; content:"[00 55]"; offset:10; depth:2; sid:1000004; rev:1;)

-----
Below is a picture of all my signatures:

```