

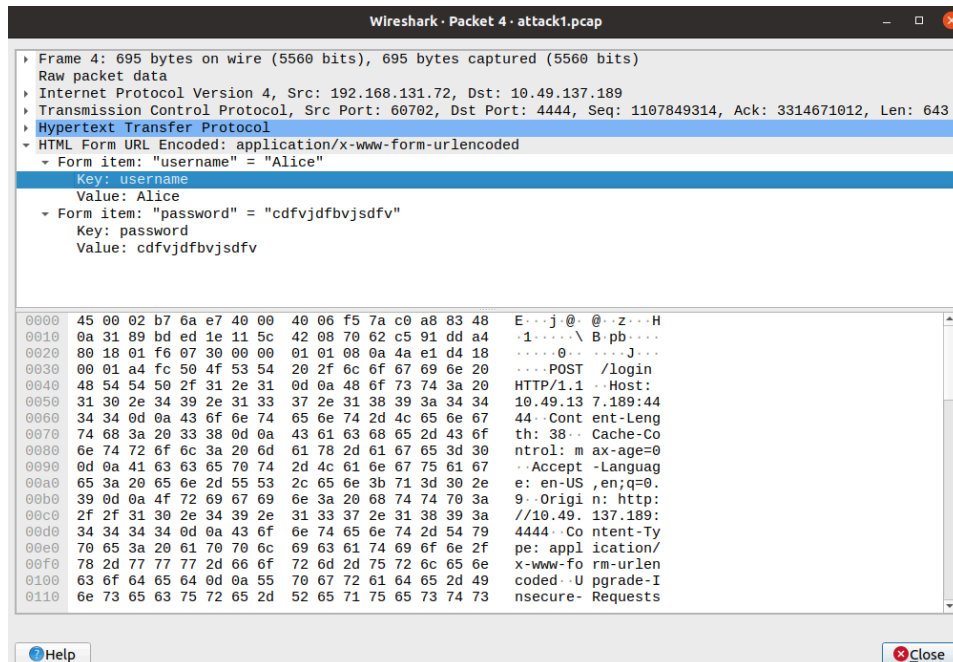
# Attack 1

Analysing attack1.pcap, we can see that there are many POST /login requests from 192.168.131.72 to 10.49.137.189.

The image below shows the attack1.pcap in Wireshark with the “http” filter.

No.	Time	Source	Destination	Protocol	Length	Info
4	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	695	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
10	2025-11-23 09:4...	10.49.137.189	192.168.131.72	HTTP	944	HTTP/1.0 200 OK (text/html)
16	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	689	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
22	2025-11-23 09:4...	10.49.137.189	192.168.131.72	HTTP	944	HTTP/1.0 200 OK (text/html)
28	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	689	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
34	2025-11-23 09:4...	10.49.137.189	192.168.131.72	HTTP	944	HTTP/1.0 200 OK (text/html)
40	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	689	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
46	2025-11-23 09:4...	10.49.137.189	192.168.131.72	HTTP	944	HTTP/1.0 200 OK (text/html)
52	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	688	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
58	2025-11-23 09:4...	10.49.137.189	192.168.131.72	HTTP	944	HTTP/1.0 200 OK (text/html)
64	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	687	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
70	2025-11-23 09:4...	10.49.137.189	192.168.131.72	HTTP	944	HTTP/1.0 200 OK (text/html)
76	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	687	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
82	2025-11-23 09:4...	10.49.137.189	192.168.131.72	HTTP	944	HTTP/1.0 200 OK (text/html)
88	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	689	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
94	2025-11-23 09:4...	10.49.137.189	192.168.131.72	HTTP	944	HTTP/1.0 200 OK (text/html)
100	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	687	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
108	2025-11-23 09:4...	10.49.137.189	192.168.131.72	HTTP	277	HTTP/1.0 303 SEE OTHER (text/html)
114	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	687	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
120	2025-11-23 09:4...	10.49.137.189	192.168.131.72	HTTP	944	HTTP/1.0 200 OK (text/html)

Upon taking a closer look at each of the POST requests, we notice that the suspicious user is trying to brute force the password for Alice.



Based on the HTTP packets, we notice that the packet length for a failed password is 944, with HTTP 200 OK. As such, we can use Wireshark to filter the packets with “http and frame.len != 944” rule and the result is shown below.

http and frame.len != 944							
No.	Time	Source	Destination	Protocol	Length	Info	
4	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	695	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
16	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	689	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
28	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	689	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
40	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	689	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
52	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	688	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
64	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	687	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
76	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	687	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
88	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	689	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
100	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	687	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
108	2025-11-23 09:4...	10.49.137.189	192.168.131.72	HTTP	277	HTTP/1.0 303 SEE OTHER	(text/html)
114	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	687	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
126	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	688	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
138	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	688	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
150	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	687	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
162	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	687	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
174	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	687	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
186	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	689	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
198	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	687	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
210	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	689	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)
222	2025-11-23 09:4...	192.168.131.72	10.49.137.189	HTTP	690	POST /login	HTTP/1.1 (application/x-www-form-urlencoded)

We can see that there is only 1 successful reply, which is captured in packet 108 with the 303 status code. As such, we know that the attacker successfully logged in to the system and gained information.

We know that packet 108 is the result of the HTTP login request on packet 100, so looking at packet 100, we know that the username is Alice and the password is monkey as shown below.

Wireshark · Packet 100 · attack1.pcap

Frame 100: 687 bytes on wire (5496 bits), 687 bytes captured (5496 bits)

Raw packet data

Internet Protocol Version 4, Src: 192.168.131.72, Dst: 10.49.137.189

Transmission Control Protocol, Src Port: 48144, Dst Port: 4444, Seq: 4206679595, Ack: 2815036089, Len: 635

Hypertext Transfer Protocol

HTML Form URL Encoded: application/x-www-form-urlencoded

- Form item: "username" = "Alice"
  - Key: username
  - Value: Alice
- Form item: "password" = "monkey"
  - Key: password
  - Value: monkey

```

0000 45 00 02 af 4e 3e 40 00 40 06 12 2c c0 a8 83 48  E...N>@. @.,...H
0010 0a 31 89 bd bc 10 11 5c fa bc d6 2b a7 ca 0a b9  .1.....\...+...
0020 80 18 01 f6 82 ff 00 00 01 01 08 0a 4a e1 e1 0b  .001080a4ae1e10b
0030 00 01 b2 36 50 4f 53 54 20 2f 6c 6f 67 69 6e 20  ...6POST /login
0040 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20  HTTP/1.1 .Host:
0050 31 30 2e 34 39 2e 31 33 37 2e 31 38 39 3a 34 34  10.49.137.189:44
0060 34 34 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67  44..Content-Leng
0070 74 68 3a 20 33 30 0d 0a 43 61 63 68 65 2d 43 6f  th: 30..Cache-Co
0080 6e 74 72 6f 6c 3a 20 6d 61 78 2d 61 67 65 3d 30  ntrol: max-age=0
0090 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 61 67  ..Accept -Languag
00a0 65 3a 20 65 6e 2d 55 53 2c 65 6e 3b 71 3d 30 2e  e: en-US ,en;q=0.
00b0 39 0d 0a 4f 72 69 67 69 6e 3a 20 68 74 74 70 3a  9..Origin: http:
00c0 2f 2f 31 30 2e 34 39 2e 31 33 37 2e 31 38 39 3a  //10.49.137.189:
00d0 34 34 34 34 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79  4444..Content-Ty
00e0 70 65 3a 20 61 70 70 6c 69 63 61 74 69 6f 6e 2f  pe: appl ication/
00f0 78 2d 77 77 77 2d 66 6f 72 6d 2d 75 72 6c 65 6e  x-www-fo rm-urle
0100 63 6f 64 65 64 0d 0a 55 70 67 72 61 64 65 2d 49  coded..U pgrade-I
0110 6e 73 65 63 75 72 65 2d 52 65 71 75 65 73 74 73  nsecure- Requests

```

# Attack 2

From the image below, attack2.pcap shows many TCP SYN packets being sent from the same IP to the same destination, but to different ports. This resembles reconnaissance through port scanning, to find out which ports are open.

No.	Time	Source	Destination	Protocol	Length	Info
31	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63988 → 445 [SYN] Seq=1564659995 Win=1024 Len=0 MSS=1460
32	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63988 → 993 [SYN] Seq=1564659995 Win=1024 Len=0 MSS=1460
33	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63988 → 111 [SYN] Seq=1564659995 Win=1024 Len=0 MSS=1460
34	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63988 → 995 [SYN] Seq=1564659995 Win=1024 Len=0 MSS=1460
35	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63988 → 8080 [SYN] Seq=1564659995 Win=1024 Len=0 MSS=1460
36	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63988 → 8888 [SYN] Seq=1564659995 Win=1024 Len=0 MSS=1460
37	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63988 → 587 [SYN] Seq=1564659995 Win=1024 Len=0 MSS=1460
38	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63988 → 23 [SYN] Seq=1564659995 Win=1024 Len=0 MSS=1460
39	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63988 → 143 [SYN] Seq=1564659995 Win=1024 Len=0 MSS=1460
40	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63988 → 113 [SYN] Seq=1564659995 Win=1024 Len=0 MSS=1460
41	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63988 → 25 [SYN] Seq=1564659995 Win=1024 Len=0 MSS=1460
42	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63988 → 3389 [SYN] Seq=1564659995 Win=1024 Len=0 MSS=1460
43	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63988 → 256 [SYN] Seq=1564659995 Win=1024 Len=0 MSS=1460
44	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63988 → 5900 [SYN] Seq=1564659995 Win=1024 Len=0 MSS=1460
45	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63988 → 199 [SYN] Seq=1564659995 Win=1024 Len=0 MSS=1460
46	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63988 → 443 [SYN] Seq=1564528921 Win=1024 Len=0 MSS=1460
47	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63986 → 22 [SYN] Seq=1564528921 Win=1024 Len=0 MSS=1460
48	2025-11-24 05:3...	192.168.131.72	10.49.161.151	TCP	44	63986 → 135 [SYN] Seq=1564528921 Win=1024 Len=0 MSS=1460
49	2025-11-24 05:3...	10.49.161.151	192.168.131.72	ICMP	72	Destination unreachable (Host administratively prohibited)
50	2025-11-24 05:3...	10.49.161.151	192.168.131.72	TCP	44	22 → 63986 [SYN, ACK] Seq=2889644744 Ack=1564528922 Win=26883...

As such, I created a rule as shown below to detect TCP SYN port scanning. It checks for SYN flags and will trigger alerts based on threshold, tracked by source. The threshold that I have set is 1000 SYN packets within 5 seconds.

```
# Port Scanning
alert tcp any any -> any any (msg:"Possible TCP SYN port scanning"; flags:S; threshold:type
threshold, track by_src, count 1000, seconds 5; classtype: network-scan; sid:1000021; rev:1;)
```

After sending attack2.pcap to suricata, we can see that there is an alert trigger in the logs.

```
root@1006859:/home/seed/Desktop/IDS/PCAPS# cat /var/log/suricata/eve.json |
grep -a "Possible TCP SYN port scanning"
{"timestamp":"2025-11-24T05:33:15.962767-0500","flow_id":1038831234063638,"
pcap_cnt":1084,"event_type":"alert","src_ip":"192.168.131.72","src_port":63
986,"dest_ip":"10.49.161.151","dest_port":2608,"proto":"TCP","ip_v":4,"pkt
src":"wire/pcap","alert":{"action":"allowed","gid":1,"signature_id":1000021
,"rev":1,"signature":"Possible TCP SYN port scanning","category":"Detection
of a Network Scan","severity":3},"direction":"to_server","flow":{"pkts_to_s
erver":1,"pkts_to_client":0,"bytes_to_server":44,"bytes_to_client":0,"start":"
2025-11-24T05:33:15.962767-0500","src_ip":"192.168.131.72","dest_ip":"10.49
.161.151","src_port":63986,"dest_port":2608}}
```

Through Wireshark, we know that the attacker is performing SYN port scanning.

# Attack 3

From the image below, attack3.pcap contains POST requests with suspicious field content. Upon taking a look at the POST request to /login, I notice that there seems to be a UNION SQL injection, where the attacker uses “union select” to extract additional information from the database.

The image shows a Wireshark packet capture of an HTTP POST request to /login. The packet is 601 bytes long and is captured on the interface 10.49.137.189. The payload is application/x-www-form-urlencoded. The form data contains a 'username' field with the value 'union select 'Alice','' --' and a 'password' field with the value '27+union+select+ %27Alice %27%2C%2 2%22+--&'. The packet is captured at 2025-11-23 09:47:42.237379-0500.

As such, I created a rule to detect HTTP inputs that contain “union”, followed by “select”, case insensitive as shown below.

```
# UNION SQL Injection
alert http any any -> any any (msg:"Possible UNION SQL injection"; flow:established,to_server;
content:"union"; nocase; content:"select"; nocase; distance:0; classtype:web-application-
attack; sid:1000022; rev:1;)
```

After sending attack3.pcap to suricata, we can see that there is an alert trigger in the logs.

```
root@1006859:/home/seed/Desktop/IDS/PCAPS# cat /var/log/suricata/eve.json |
grep -a "Possible UNION SQL injection"
{"timestamp":"2025-11-23T09:47:42.237379-0500","flow_id":1798199931839800,"
pcap_cnt":2209,"event_type":"alert","src_ip":"192.168.131.72","src_port":59
274,"dest_ip":"10.49.137.189","dest_port":4444,"proto":"TCP","ip_v":4,"pkt_
src":"wire/pcap","alert":{"action":"allowed","gid":1,"signature_id":1000022
,"rev":1,"signature":"Possible UNION SQL injection","category":"Web Applica
tion Attack","severity":1},"app_proto":"http","direction":"to_server","flow
":{"pkts_toserver":4,"pkts_toclient":3,"bytes_toserver":765,"bytes_toclient
":188,"start":"2025-11-23T09:47:42.090996-0500","src_ip":"192.168.131.72","
dest_ip":"10.49.137.189","src_port":59274,"dest_port":4444}}
```



Through Wireshark, and observing packet 2206, we know that the attacker is performing SQL injection with UNION.

## Attack 4

From the image below, attack4.pcap contains HTTP GET requests on different pages. However, we observe that there is a suspicious GET request that seems to inject a command on the server to read /etc/passwd.

http and ip.src_host==192.168.131.72						
No.	Time	Source	Destination	Protocol	Length	Info
4	2025-11-23 09:5...	192.168.131.72	10.49.137.189	HTTP	563	GET /home HTTP/1.1
24	2025-11-23 09:5...	192.168.131.72	10.49.137.189	HTTP	563	GET /about HTTP/1.1
40	2025-11-23 09:5...	192.168.131.72	10.49.137.189	HTTP	566	GET /contact HTTP/1.1
58	2025-11-23 09:5...	192.168.131.72	10.49.137.189	HTTP	565	GET /home HTTP/1.1
78	2025-11-23 09:5...	192.168.131.72	10.49.137.189	HTTP	563	GET /image HTTP/1.1
96	2025-11-23 09:5...	192.168.131.72	10.49.137.189	HTTP	565	GET /img/18 HTTP/1.1
106	2025-11-23 09:5...	192.168.131.72	10.49.137.189	HTTP	544	GET /favicon.ico HTTP/1.1
120	2025-11-23 09:5...	192.168.131.72	10.49.137.189	HTTP	563	GET /home HTTP/1.1
140	2025-11-23 09:5...	192.168.131.72	10.49.137.189	HTTP	593	GET /website?u=http://10.49.137.189:4444 HTTP/1.1
160	2025-11-23 09:5...	192.168.131.72	10.49.137.189	HTTP	545	GET /website?u=cat%20/etc/passwd HTTP/1.1
172	2025-11-23 09:5...	192.168.131.72	10.49.137.189	HTTP	595	GET /upload HTTP/1.1
182	2025-11-23 09:5...	192.168.131.72	10.49.137.189	HTTP	593	GET /news HTTP/1.1
194	2025-11-23 09:5...	192.168.131.72	10.49.137.189	HTTP	635	GET /redirect?url=http://10.49.137.189:4444/contact HTTP/1.1
204	2025-11-23 09:5...	192.168.131.72	10.49.137.189	HTTP	596	GET /contact HTTP/1.1

As such, I created a rule to detect common command injection operators like “;”, “|”, “||”, “&&” as shown below.

```
# Command Injection
alert http any any -> any any {msg:"Possible Command Injection"; flow:established,to_server;
http.uri; pcre:"/(\\;|\\||\\&&)/"; classtype:web-application-attack; sid:1000023; rev:1; }
```

After sending attack4.pcap to suricata, we observe that there is an alert trigger in the logs.

```
root@1006859:/home/seed/Desktop/IDS/PCAPS# cat /var/log/suricata/eve.json |
grep -a "Possible Command Injection"
{"timestamp":"2025-11-23T09:50:59.896698-0500","flow_id":979295079778229,"p
cap_cnt":163,"event_type":"alert","src_ip":"192.168.131.72","src_port":5660
6,"dest_ip":"10.49.137.189","dest_port":4444,"proto":"TCP","ip_v":4,"pkt_sr
c":"wire/pcap","tx_id":0,"alert":{"action":"allowed","gid":1,"signature_id"
:1000023,"rev":1,"signature":"Possible Command Injection","category":"Web A
pplication Attack","severity":1},"ts_progress":"request_complete","tc_progr
ess":"response_headers","http":{"hostname":"10.49.137.189","http_port":4444
,"url":"/website?u=cat%20/etc/passwd","http_user_agent":"Mozilla/5.0 (X11;
Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0","http_method":"GET",
"protocol":"HTTP/1.1","status":200,"length":0},"app_proto":"http","directio
n":"to_server","flow":{"pkts_toserver":4,"pkts_toclient":3,"bytes_toserver"
:709,"bytes_toclient":181,"start":"2025-11-23T09:50:59.752297-0500","src_ip
":"192.168.131.72","dest_ip":"10.49.137.189","src_port":56606,"dest_port":4
444}}
```

Through Wireshark, we know that the attacker is performing command injection through a GET parameter, u, at /website.

With this attack, the attacker gained read access to /etc/passwd, as shown below.

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 1817
Server: Werkzeug/0.12.2 Python/3.4.5
Date: Sun, 23 Nov 2025 14:50:59 GMT

root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-bus-proxy:x:999:998:systemd Bus Proxy:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:998:997:User for polkitd:/:/sbin/nologin
tss:x:59:59:Account used by the trousers package to sandbox the tcsd daemon:/dev/null:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/ssh:/sbin/nologin
postfix:x:89:89:/var/spool/postfix:/sbin/nologin
chrony:x:997:995:/var/lib/chrony:/sbin/nologin
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
```

## Attack 5

From the image below, attack5.pcap contains various POST requests. The highlighted packet 5165 contained a suspicious POST request, which had “test” as its user-agent field.

http.request.method==POST

No.	Time	Source	Destination	Protocol	Length	Info
3370	2018-10-31 11:3...	10.100.9.107	173.171.132.82	HTTP	405	POST /sat91/HEADLESS-PC_W617601.88AD32764B61024C64A0DFEC972C6...
3533	2018-10-31 11:3...	10.100.9.107	173.171.132.82	HTTP	340	POST /sat91/HEADLESS-PC_W617601.88AD32764B61024C64A0DFEC972C6...
5165	2018-10-31 11:3...	10.100.9.107	173.171.132.82	HTTP	280	POST /sat91/HEADLESS-PC_W617601.88AD32764B61024C64A0DFEC972C6...

Wireshark · Packet 5165 · attack5.pcap

Internet Protocol Version 4, Src: 10.100.9.107, Dst: 173.171.132.82

Transmission Control Protocol, Src Port: 49220, Dst Port: 8082, Seq: 903774242, Ack: ...

[5 Reassembled TCP Segments (4834 bytes): #5158(228), #5160(1460), #5162(1460), #5164...

Hypertext Transfer Protocol

POST /sat91/HEADLESS-PC\_W617601.88AD32764B61024C64A0DFEC972C64A8/90 HTTP/1.1\r\n

Content-Type: multipart/form-data; boundary=Arasfjasu7\r\n

User-Agent: test\r\n

0000 00 02 7d e1 59 04 00 06 8b 2a 96 0a 08 00 45 00 .Y..P ..\*...E.

0010 01 0a 06 63 40 00 06 ad be 0a 64 09 6b ad ab ...c@... ..d.k..

0020 84 52 c0 44 1f 92 35 de 80 22 0b 11 66 43 50 18 R.D..5. ....fCP.

0030 fa f0 5c 32 00 00 77 65 65 6e 6a 6f 62 2d 44 43 ..\2..we enjob-DC

0040 2e 68 61 6c 6c 6f 77 65 65 6e 6a 6f 62 2e 63 6f .hallowe enjob.co

0050 6d 0d 0a 46 6f 72 65 73 74 20 74 72 65 65 73 3a m..Fores t trees:

0060 0d 0a 09 31 20 68 61 6c 6c 6f 77 65 65 6e 6a 6f ...1 hal loweenjo

0070 62 2e 63 6f 6d 0d 0a 0d 0a 0d 0a 0d 0a 4c 69 73 b.com... ..Lis

Frame (280 bytes)

Reassembled TCP (4834 bytes)

0000 00 02 7d e1 59 04

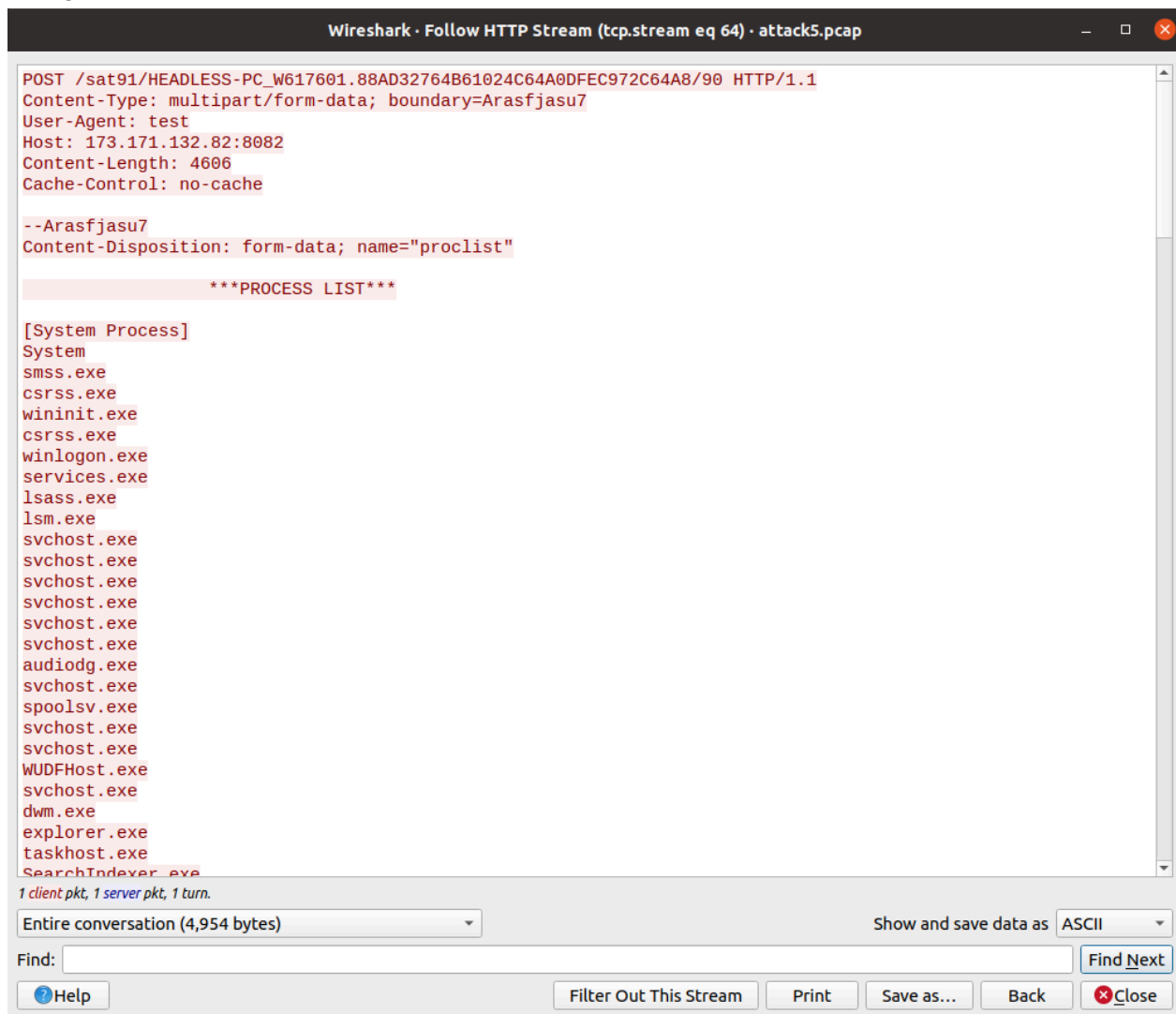
0010 01 0a 06 63 40 06

0020 84 52 c0 44 1f 92

0030 fa f0 5c 32 00 06

0040 2e 68 61 6c 6c 6f

Following the HTTP stream of packet 5165, we can see the result below. We can see the data that was being extracted from the infected client, including process list, system information, IP config, etc.



As such, I created a rule to detect HTTP POST requests that sets the User-Agent header to "test".

```
# Trickbot Trojan
alert http any any -> any any (msg:"Possible Trickbot Exfiltration - User-Agent test";
flow:established,to_server; content:"POST"; http_method; content:"User-Agent: test";
http_header; sid:1000024; rev:1;)
```

Passing attack5.pcap to suricata, we observe from the logs below that there is an alert trigger in the logs due to the “User-Agent: test” header present in a packet.

```
root@1006859:/home/seed/Desktop/IDS/PCAPS# cat /var/log/suricata/eve.json |
grep -a "Possible Trickbot Exfiltration - User-Agent test"
{"timestamp":"2018-10-31T11:36:11.105035-0400","flow_id":783581558290728,"p
cap_cnt":5160,"event_type":"alert","src_ip":"10.100.9.107","src_port":49220
,"dest_ip":"173.171.132.82","dest_port":8082,"proto":"TCP","ip_v":4,"pkt_sr
c":"wire/pcap","tx_id":0,"alert":{"action":"allowed","gid":1,"signature_id"
:1000024,"rev":1,"signature":"Possible Trickbot Exfiltration - User-Agent t
est","category":"","severity":3},"ts_progress":"request_body","tc_progress"
:"response_started","http":{"hostname":"173.171.132.82","http_port":8082,"u
rl":"/sat91/HEADLESS-PC_W617601.88AD32764B61024C64A0DFEC972C64A8/90","http
_user_agent":"test","http_method":"POST","protocol":"HTTP/1.1","length":0},"
app_proto":"http","direction":"to_server","flow":{"pkts_toserver":4,"pkts_t
oclient":2,"bytes_toserver":1916,"bytes_toclient":112,"start":"2018-10-31T1
1:36:10.837801-0400","src_ip":"10.100.9.107","dest_ip":"173.171.132.82","sr
c_port":49220,"dest_port":8082}}
```

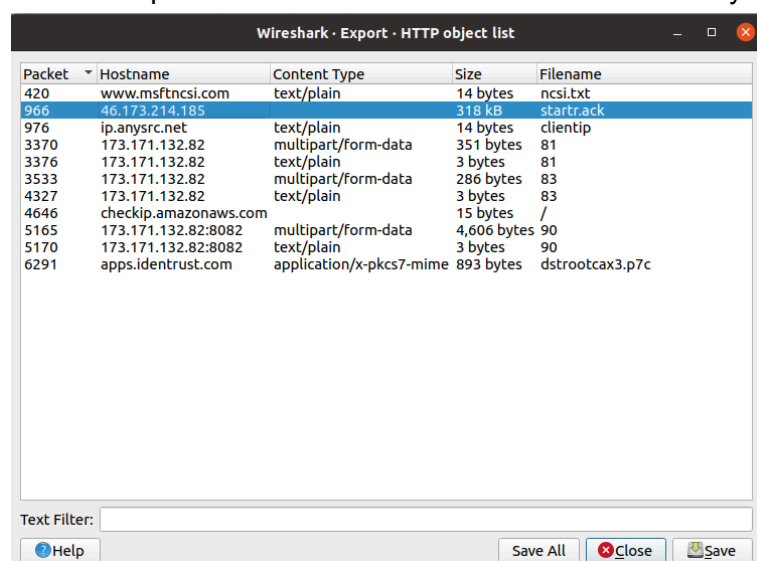
The IP address of the infected windows client is 10.100.9.107, which can be found through the source IP address of the POST request with “User-Agent: test”.

The domain name of the infected windows client is halloweenjob.com as shown below.

```
***LOCAL MACHINE DATA***

User name: CN=Ichabod Crane,CN=Users,DC=halloweenjob,DC=com
Computer name: CN=HEADLESS-PC,CN=Computers,DC=halloweenjob,DC=com
Site name: Default-First-Site-Name
Domain shortname: HALLOWEENJOB
Domain name: halloweenjob.com
Forest name: halloweenjob.com
Domain controller: Halloweenjob-DC.halloweenjob.com
Forest trees:
1 halloweenjob.com
```

To find out the name of the malware, I first exported HTTP objects from the PCAP file and we notice that packet 966 has a file startr.ack and it is a very large file of 318kB.



Packet	Hostname	Content Type	Size	Filename
420	www.msftncsi.com	text/plain	14 bytes	ncsi.txt
966	46.173.214.185		318 kB	startr.ack
976	ip.anysrc.net	text/plain	14 bytes	clientip
3370	173.171.132.82	multipart/form-data	351 bytes	81
3376	173.171.132.82	text/plain	3 bytes	81
3533	173.171.132.82	multipart/form-data	286 bytes	83
4327	173.171.132.82	text/plain	3 bytes	83
4646	checkip.amazonaws.com		15 bytes	/
5165	173.171.132.82:8082	multipart/form-data	4,606 bytes	90
5170	173.171.132.82:8082	text/plain	3 bytes	90
6291	apps.identrust.com	application/x-pkcs7-mime	893 bytes	dstrootcax3.p7c



Taking a closer look at packet 966, we can see that the data section starts with 4d 5a.



Searching for Windows Executable magic bytes/file signatures, we notice that it matches “0x4D 0x5A” for DOS Executable.

Executable Binaries	Mnemonic	Signature
DOS Executable	"MZ"	0x4D 0x5A

Hence, packet 966 is the packet that transferred the malware.  
Therefore, the name of the malware file is startr.ack.

## Attack 6

First, I exported the HTTP objects that were in attack6.pcap and I noticed that some packets were unusually huge. In particular, they were packets 142, 558, 616, and 2067.

The image shows the Wireshark 'Export - HTTP object list' window. It contains a table with the following columns: Packet, Hostname, Content Type, Size, and Filename. The table lists various HTTP objects, including text/plain, text/html, text/javascript, image/png, application/x-www-form-urlencoded, and application/x-x509-ca-cert. The sizes range from 14 bytes to 2,297 bytes. The filenames are mostly empty or contain partial paths.

Packet	Hostname	Content Type	Size	Filename
68	www.msftncsi.com	text/plain	14 bytes	ncsi.t
98	bv.truecompassdesigns.net	text/html	140 bytes	?0000
108	grandrapidsnonprofits.com	text/html	7,886 bytes	?0000
142	suburban-sanitation.com	text/javascript	27 kB	?0000
558	nailcountryandtan.com	image/png	384 kB	?0000
616	nailcountryandtan.com	image/png	45 kB	?0000
2067	nailcountryandtan.com	image/png	1,417 kB	?0000
2071	nailcountryandtan.com	image/png	2,297 bytes	?0000
2156	52.50.59.31	application/x-www-form-urlencoded	476 bytes	/
2158	52.50.59.31	text/html	185 bytes	/
2184	crt.comodoca.com	application/x-x509-ca-cert	1,400 bytes	COMC
2196	www.download.windowsupdate.com	application/x-x509-ca-cert	1,082 bytes	02FAF
2514	52.50.59.31	application/x-www-form-urlencoded	440 bytes	/
2517	52.50.59.31	text/html	185 bytes	/
2582			114 bytes	/
2589	77.225.141.195	application/x-www-form-urlencoded	436 bytes	/
2628	www.download.windowsupdate.com	application/x-x509-ca-cert	993 bytes	B51C
2738	cacerts.digicert.com	application/x-x509-ca-cert	1,176 bytes	DigiC
2911	52.50.59.31	application/x-www-form-urlencoded	464 bytes	/
2913	52.50.59.31	text/html	185 bytes	/
2981	77.225.141.195	application/x-www-form-urlencoded	464 bytes	/
3333	52.50.59.31	application/x-www-form-urlencoded	468 bytes	/
3336	52.50.59.31	text/html	185 bytes	/
3337	52.50.59.31	application/x-www-form-urlencoded	500 bytes	/

Taking a closer look at packets 558, 616, and 2067, I noticed that even though they had headers “Content-Type: image/png”, they all contained media type that had signatures “0x4d

0x5a”, instead of “0x89 0x50 0x4e 0x47” (PNG magic bytes). This means that instead of PNG files, they contained Windows Executable files instead.

Packet 558:

Media Type	
Media type: image/png (384294 bytes)	
00000120	67 65 2f 70 6e 67 0d 0a 0d 0a 4d 5a 90 00 03 00 ge/png ..MZ....
00000130	00 00 04 00 00 00 ff ff 00 00 b8 00 00 00 00 .....
00000140	00 00 40 00 00 00 00 00 00 00 00 00 00 00 ..@.....
00000150	00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000160	00 00 00 00 00 00 80 00 00 00 0e 1f ba 0e 00 b4 .....
00000170	09 cd 21 b8 01 4c cd 21 54 68 69 73 20 70 72 6f ..!..L.! This pro
00000180	67 72 61 6d 20 63 61 6e 6e 6f 74 20 62 65 20 72 gram can not be r
00000190	75 6e 20 69 6e 20 44 4f 53 20 6d 6f 64 65 2e 0d un in DO S mode..
000001a0	0d 0a 24 00 00 00 00 00 00 00 50 45 00 00 4c 01 ..\$. ....PE..L.
000001b0	08 00 a1 e2 6a 3b 00 00 00 00 00 00 00 00 e0 00 ....j;.....
000001c0	0e 01 0b 01 02 17 00 30 01 00 00 12 06 00 00 e0 .....0 .....
000001d0	03 00 db 33 00 00 00 10 00 00 00 40 01 00 00 00 ...3.....@....
000001e0	40 00 00 10 00 00 00 02 00 00 04 00 00 01 00 @.....>....
000001f0	00 00 04 00 00 00 00 00 00 00 a0 3e 06 00 00 04 .....

Packet 616:

Media Type	
Media type: image/png (45056 bytes)	
0120	67 65 2f 70 6e 67 0d 0a 0d 0a 4d 5a 90 00 03 00 ge/png ..MZ....
0130	00 00 04 00 00 00 ff ff 00 00 b8 00 00 00 00 00 .....
0140	00 00 40 00 00 00 00 00 00 00 00 00 00 00 ..@.....
0150	00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0160	00 00 00 00 00 00 e8 00 00 00 0e 1f ba 0e 00 b4 .....
0170	09 cd 21 b8 01 4c cd 21 54 68 69 73 20 70 72 6f ..!..L.! This pro
0180	67 72 61 6d 20 63 61 6e 6e 6f 74 20 62 65 20 72 gram can not be r
0190	75 6e 20 69 6e 20 44 4f 53 20 6d 6f 64 65 2e 0d un in DO S mode..
01a0	0d 0a 24 00 00 00 00 00 00 00 98 b3 ad 85 dc d2 ..\$. ....
01b0	c3 d6 dc d2 c3 d6 dc d2 c3 d6 a7 ce cf d6 db d2 .....
01c0	c3 d6 5f ce cd d6 dd d2 c3 d6 5f da 9e d6 da d2 .._....._.....
01d0	c3 d6 dc d2 c2 d6 71 d2 c3 d6 34 cd c9 d6 d3 d2 .....q...4.....
01e0	c3 d6 64 d4 c5 d6 dd d2 c3 d6 34 cd c7 d6 da d2 ...d.....4.....
01f0	c3 d6 52 69 63 68 dc d2 c3 d6 00 00 00 00 00 00 ..Rich.....

Packet 2067:

Media Type	
Media type: image/png (1417216 bytes)	
00000130	70 6e 67 0d 0a 0d 0a 4d 5a 90 00 03 00 00 00 04 png....M Z.....
00000140	00 00 00 ff ff 00 00 b8 00 00 00 00 00 00 40 .....
00000150	00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000160	00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000170	00 00 00 f8 00 00 00 0e 1f ba 0e 00 b4 09 cd 21 .....
00000180	b8 01 4c cd 21 54 68 69 73 20 70 72 6f 67 72 61 ..L!Thi s progra
00000190	6d 20 63 61 6e 6e 6f 74 20 62 65 20 72 75 6e 20 m cannot be run
000001a0	69 6e 20 44 4f 53 20 6d 6f 64 65 2e 0d 0d 0a 24 in DOS m ode....\$
000001b0	00 00 00 00 00 00 00 ff 90 b5 04 bb f1 db 57 bb .....
000001c0	f1 db 57 bb f1 db 57 c0 ed d7 57 a5 f1 db 57 38 ..W...W...W...W8
000001d0	ed d5 57 bf f1 db 57 8d d7 d1 57 b0 f1 db 57 3c ..W...W...W...W<
000001e0	ed d9 57 94 f1 db 57 35 f9 84 57 be f1 db 57 38 ..W...W5...W...W8
000001f0	f9 86 57 b6 f1 db 57 bb f1 da 57 92 f0 db 57 53 ..W...W...W...WS
00000200	ee d1 57 ba f0 db 57 03 f7 dd 57 ba f1 db 57 53 ..W...W...W...WS

As such, I wrote a rule to detect Windows Executable that are disguised as PNG images. It also checks whether a downloaded file starts with “MZ”, which corresponds to “0x4d 0x5a”.

```
# Windows Malware Disguised as PNG
alert http any any -> any any (msg:"Windows Malware Disguised as PNG";
flow:established,to_client; content:"MZ"; distance:0; content:"Content-Type: image/png";
http_header; classtype:trojan-activity; sid:1000025; rev:1;)
```

Passing attack6.pcap to suricata, we can observe that there were 3 alert triggers from the logs as shown below, reflecting the 3 packets, 558, 616, and 2067.

```
root@1006859:/home/seed/Desktop/IDS/PCAPs# cat /var/log/suricata/eve.json | grep -a "Windows Malware Disguised as PNG"
{"timestamp": "2017-03-21T11:49:25.125123-0400", "flow_id": 1131043835728263, "pcap_cnt": 154, "event_type": "alert", "src_ip": "50.63.125.1", "src_port": 80, "dest_ip": "192.168.22.94", "dest_port": 49161, "proto": "TCP", "ip_v": 4, "pkt_src": "wire/pcap", "tx_id": 0, "alert": {"action": "allowed", "gid": 1, "signature_id": 1000025, "rev": 1, "signature": "Windows Malware Disguised as PNG", "category": "A Network Trojan was detected", "severity": 1}, "ts_progress": "request complete", "tc_progress": "response body", "http": {"hostname": "nailcountryandtan.com", "url": "/counter/?0000001MKqMadoTwsD8bMbWxfG2zHjraZnWghk2xY5rpyqa6RhRlo6U7zbn07DD8M0P1pZrllNTv383v8Y7CIMAzGZPifYdnKvrwm9Mm8G W0bGLE74JD74zik2n-N_qCHLo9TFUXHSRbMGL2", "http_user_agent": "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Win64; x64; Trident/4.0; .NET CLR 2.0.50727; SLCC2; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)", "http_content_type": "image/png", "http_method": "GET", "protocol": "HTTP/1.1", "status": 200, "length": 2622, "files": [{"filename": "a.png", "gaps": false, "state": "UNKNOWN", "stored": false, "size": 2622, "tx_id": 0}], "app_proto": "http", "direction": "to client", "flow": {"pkts_toserver": 4, "pkts_toclient": 5, "bytes_toserver": 705, "bytes_toclient": 6118, "start": "2017-03-21T11:49:24.984237-0400", "src_ip": "192.168.22.94", "dest_ip": "50.63.125.1", "src_port": 49161, "dest_port": 80}}
{"timestamp": "2017-03-21T11:49:25.474991-0400", "flow_id": 1131043835728263, "pcap_cnt": 564, "event_type": "alert", "src_ip": "50.63.125.1", "src_port": 80, "dest_ip": "192.168.22.94", "dest_port": 49161, "proto": "TCP", "ip_v": 4, "pkt_src": "wire/pcap", "tx_id": 1, "alert": {"action": "allowed", "gid": 1, "signature_id": 1000025, "rev": 1, "signature": "Windows Malware Disguised as PNG", "category": "A Network Trojan was detected", "severity": 1}, "ts_progress": "request complete", "tc_progress": "response body", "http": {"hostname": "nailcountryandtan.com", "url": "/counter/?0000001MKqMadoTwsD8bMbWxfG2zHjraZnWghk2xY5rpyqa6RhRlo6U7zbn07DD8M0P1pZrllNTv383v8Y7CIMAzGZPifYdnKvrwm9Mm8G W0bGLE74JD74zik2n-N_qCHLo9TFUXHSRbMGL3", "http_user_agent": "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Win64; x64; Trident/4.0; .NET CLR 2.0.50727; SLCC2; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)", "http_content_type": "image/png", "http_method": "GET", "protocol": "HTTP/1.1", "status": 200, "length": 2622, "files": [{"filename": "853.png", "gaps": false, "state": "UNKNOWN", "stored": false, "size": 2622, "tx_id": 1}], "app_proto": "http", "direction": "to client", "flow": {"pkts_toserver": 151, "pkts_toclient": 268, "bytes_toserver": 9984, "bytes_toclient": 403452, "start": "2017-03-21T11:49:24.984237-0400", "src_ip": "192.168.22.94", "dest_ip": "50.63.125.1", "src_port": 49161, "dest_port": 80}}
{"timestamp": "2017-03-21T11:49:25.613384-0400", "flow_id": 1131043835728263, "pcap_cnt": 623, "event_type": "alert", "src_ip": "50.63.125.1", "src_port": 80, "dest_ip": "192.168.22.94", "dest_port": 49161, "proto": "TCP", "ip_v": 4, "pkt_src": "wire/pcap", "tx_id": 2, "alert": {"action": "allowed", "gid": 1, "signature_id": 1000025, "rev": 1, "signature": "Windows Malware Disguised as PNG", "category": "A Network Trojan was detected", "severity": 1}, "ts_progress": "request complete", "tc_progress": "response body", "http": {"hostname": "nailcountryandtan.com", "url": "/counter/?0000001MKqMadoTwsD8bMbWxfG2zHjraZnWghk2xY5rpyqa6RhRlo6U7zbn07DD8M0P1pZrllNTv383v8Y7CIMAzGZPifYdnKvrwm9Mm8G W0bGLE74JD74zik2n-N_qCHLo9TFUXHSRbMGL4", "http_user_agent": "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Win64; x64; Trident/4.0; .NET CLR 2.0.50727; SLCC2; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)", "http_content_type": "image/png", "http_method": "GET", "protocol": "HTTP/1.1", "status": 200, "length": 4069, "files": [{"filename": "0c1d552cb1d4cb.png", "gaps": false, "state": "UNKNOWN", "stored": false, "size": 4069, "tx_id": 2}], "app_proto": "http", "direction": "to client", "flow": {"pkts_toserver": 176, "pkts_toclient": 302, "bytes_toserver": 11943, "bytes_toclient": 453562, "start": "2017-03-21T11:49:24.984237-0400", "src_ip": "192.168.22.94", "dest_ip": "50.63.125.1", "src_port": 49161, "dest_port": 80}}
```

From Wireshark, we can tell that the attacker used PNG image files to camouflage it as a normal file even though it was malware. The image below shows that “Content-Type: image/png” header was set, but the payload started with “MZ” instead, which was the file signature for Windows Executables.

