COMP90073 Security Analytics

Project 1: Detecting cyberattacks in network traffic data

Name: Chen-An Fan
Student ID: 1087032
User Name: chenanf

Introduction

In this project, a pcap file of network traffic of a victim network under cyberattacks is given for analysis. In this report, a list of attack, method for identifying attacks, and consequences of attacks will be discussed. Also, the countermeasure to address each attack will also be provided.

Summary

202.166.84.165 is the IP address of the compromised machine. This machine was controlled as a bot to send SPAM and conduct ClickFraud by clicking advertisement on a specific website. The attacker control this machine through 17 IRC servers. We could monitor the unusual traffic, cut off the connection between bot and servers, or block the bot directly in order to mitigate this cyberattack.

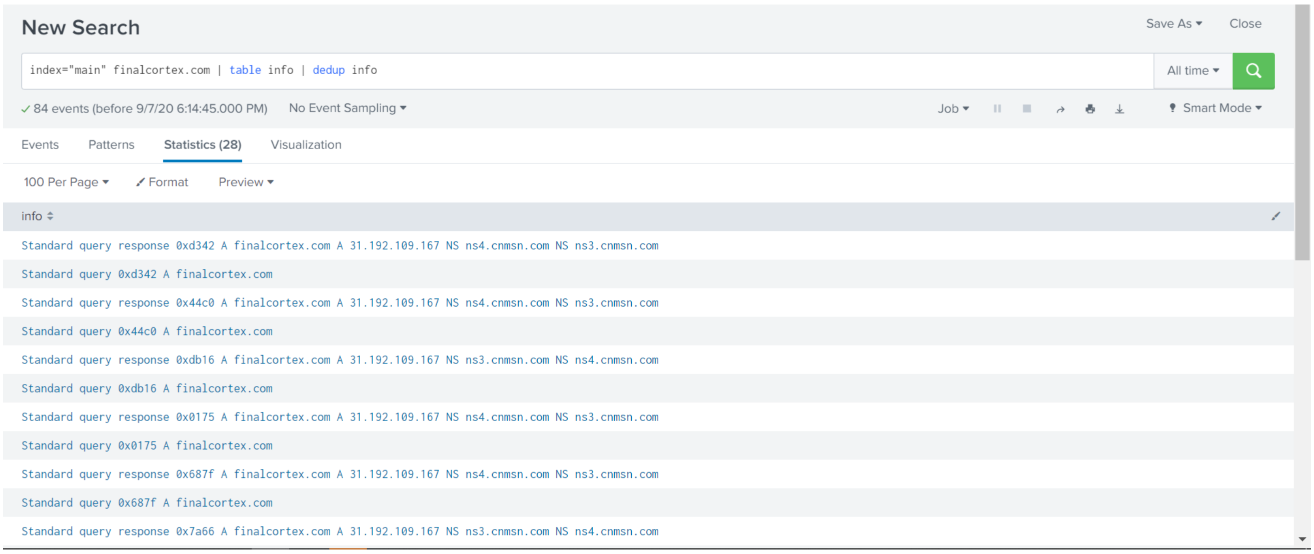
Attacks

Botnet Command & Control (C2)

Pattern: src_ip, dst_ip

To identify the Botnet Command & Control attack, first we need to find the IP address of the C2 server *"finalcortex.com"*. Use SPL command to find the information of *finalcortex.com*.

```
index="main" finalcortex.com | table info | dedup info
```



```
index="main" finalcortex.com | table info | dedup info
```

✓ 84 events (before 9/7/20 6:14:45.000 PM)

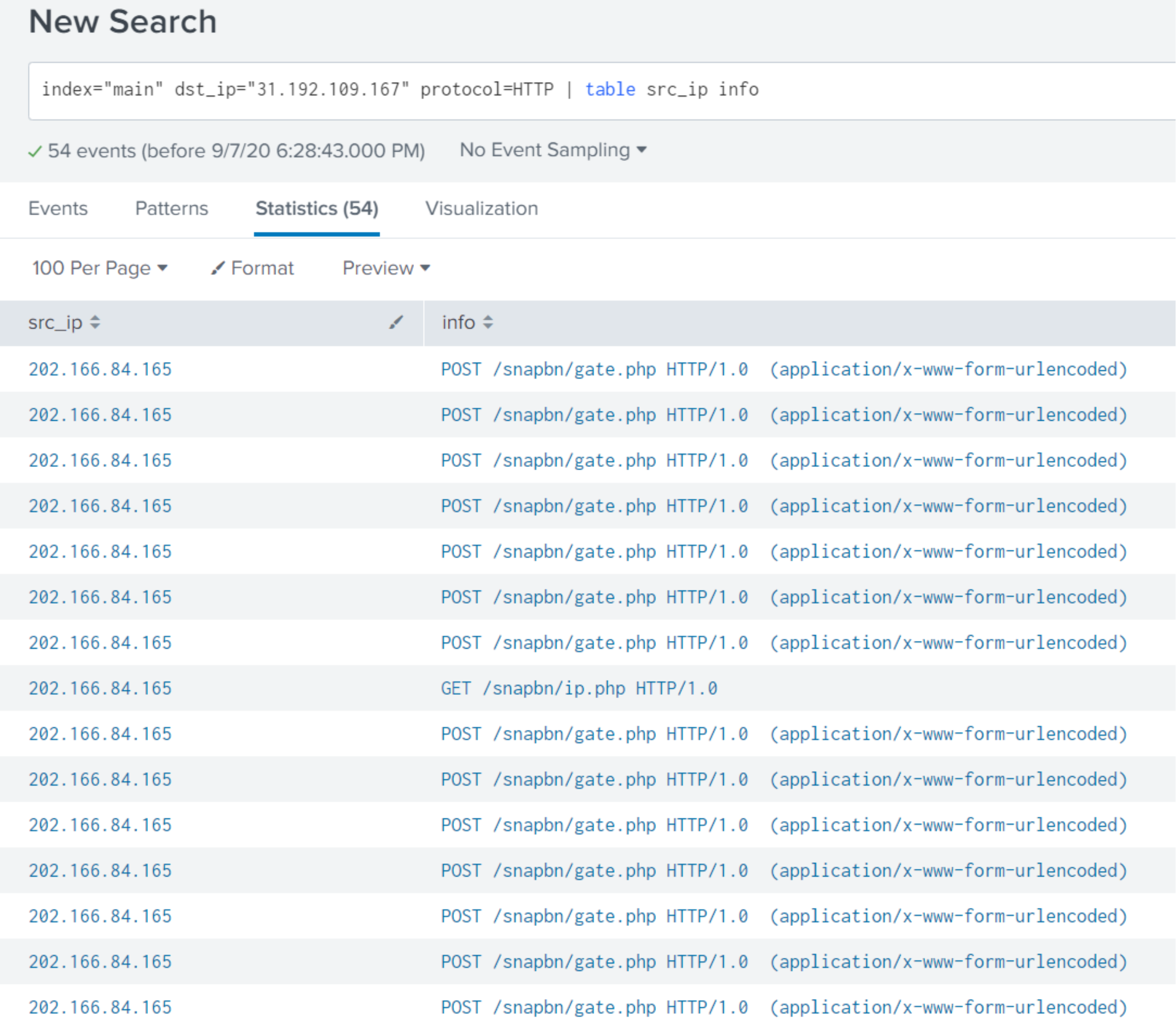
```
info
Standard query response 0xd342 A finalcortex.com A 31.192.109.167 NS ns4.cnmsn.com NS ns3.cnmsn.com
Standard query 0xd342 A finalcortex.com
Standard query response 0x44c0 A finalcortex.com A 31.192.109.167 NS ns4.cnmsn.com NS ns3.cnmsn.com
Standard query 0x44c0 A finalcortex.com
Standard query response 0xdb16 A finalcortex.com A 31.192.109.167 NS ns3.cnmsn.com NS ns4.cnmsn.com
Standard query 0xdb16 A finalcortex.com
Standard query response 0x0175 A finalcortex.com A 31.192.109.167 NS ns4.cnmsn.com NS ns3.cnmsn.com
Standard query 0x0175 A finalcortex.com
Standard query response 0x687f A finalcortex.com A 31.192.109.167 NS ns4.cnmsn.com NS ns3.cnmsn.com
Standard query 0x687f A finalcortex.com
Standard query response 0x7a66 A finalcortex.com A 31.192.109.167 NS ns3.cnmsn.com NS ns4.cnmsn.com
```

Figure 1 - Information of finalcortex.com (partial)

As we can see in the figure 1, the IP address of finalcortex.com is **`31.192.109.167`.**

In a botnet, backdoors are installed on compromised machine (bots). The backdoor will first test its connectivity to the C2 server with an initial *HTTP GET* request. After that the backdoor will start upload files to the C2 server through the *HTTP POST* request [1]. Therefore, we could use SPL command to examine the traffic to the C2 server *31.192.109.167*.

```
index="main" dst_ip="31.192.109.167" protocol=HTTP | table src_ip info
```



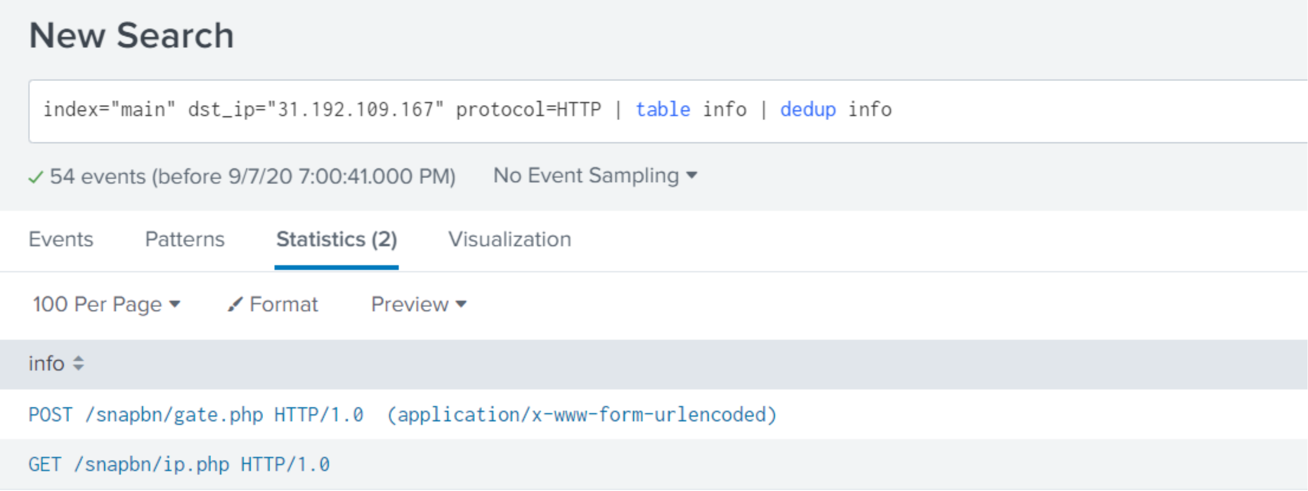
```
index="main" dst_ip="31.192.109.167" protocol=HTTP | table src_ip info
```

✓ 54 events (before 9/7/20 6:28:43.000 PM)

src_ip	info
202.166.84.165	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
202.166.84.165	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
202.166.84.165	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
202.166.84.165	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
202.166.84.165	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
202.166.84.165	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
202.166.84.165	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
202.166.84.165	GET /snapbn/ip.php HTTP/1.0
202.166.84.165	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
202.166.84.165	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
202.166.84.165	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
202.166.84.165	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
202.166.84.165	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
202.166.84.165	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
202.166.84.165	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)

Figure 2 **-** A Compromised Machine Communicate with C2 Server (partial)

As we can see in the figure 2, there are **54** HTTP C2 requests and the compromised machine's IP is `202.166.84.165`. Moreover, we find the URI strings used in this attack are: **`POST /snapbn/ip.php`** and **`GET /snapbn/gate.php`**. (Figure 3)



```
index="main" dst_ip="31.192.109.167" protocol=HTTP | table info | dedup info
```

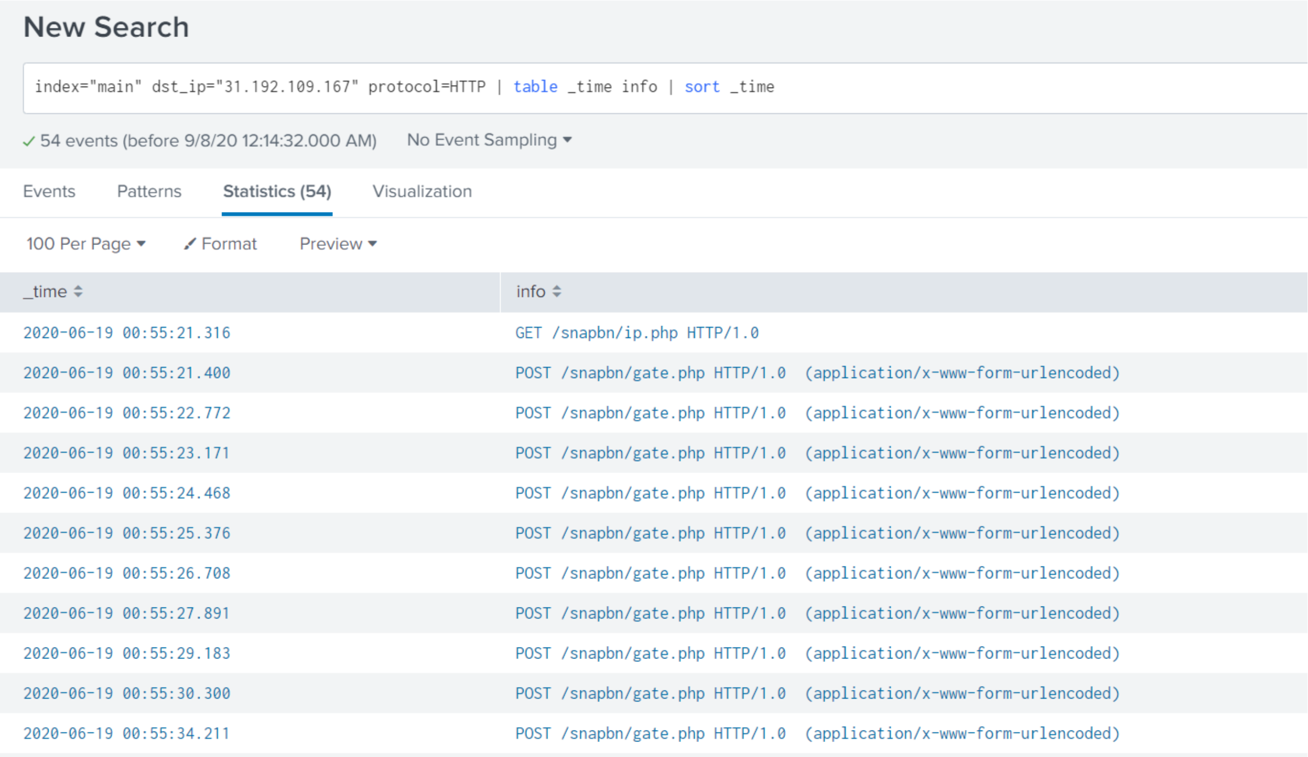
✓ 54 events (before 9/7/20 7:00:41.000 PM)

info
POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
GET /snapbn/ip.php HTTP/1.0

Figure 3 - URI Strings

To find the start time and end time of this attack, I built a table with information about event's arrival time, and then sort this table by the SPL command:

```
index="main" dst_ip="31.192.109.167" protocol=HTTP | table _time info | sort _time
```



```
index="main" dst_ip="31.192.109.167" protocol=HTTP | table _time info | sort _time
```

✓ 54 events (before 9/8/20 12:14:32.000 AM) No Event Sampling ▾

_time	info
2020-06-19 00:55:21.316	GET /snapbn/ip.php HTTP/1.0
2020-06-19 00:55:21.400	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
2020-06-19 00:55:22.772	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
2020-06-19 00:55:23.171	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
2020-06-19 00:55:24.468	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
2020-06-19 00:55:25.376	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
2020-06-19 00:55:26.708	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
2020-06-19 00:55:27.891	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
2020-06-19 00:55:29.183	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
2020-06-19 00:55:30.300	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)
2020-06-19 00:55:34.211	POST /snapbn/gate.php HTTP/1.0 (application/x-www-form-urlencoded)

Figure 4 - Botnet C2 Attack Event Time (partial)

As figure 4 shows in the chronological order, not surprised, the first event is the GET request. We can find the start time at the top and end time at the end of the table.

`Start time: 2020-06-19 00:55:21.316`

`End time: 2020-06-19 00:58:00.085`

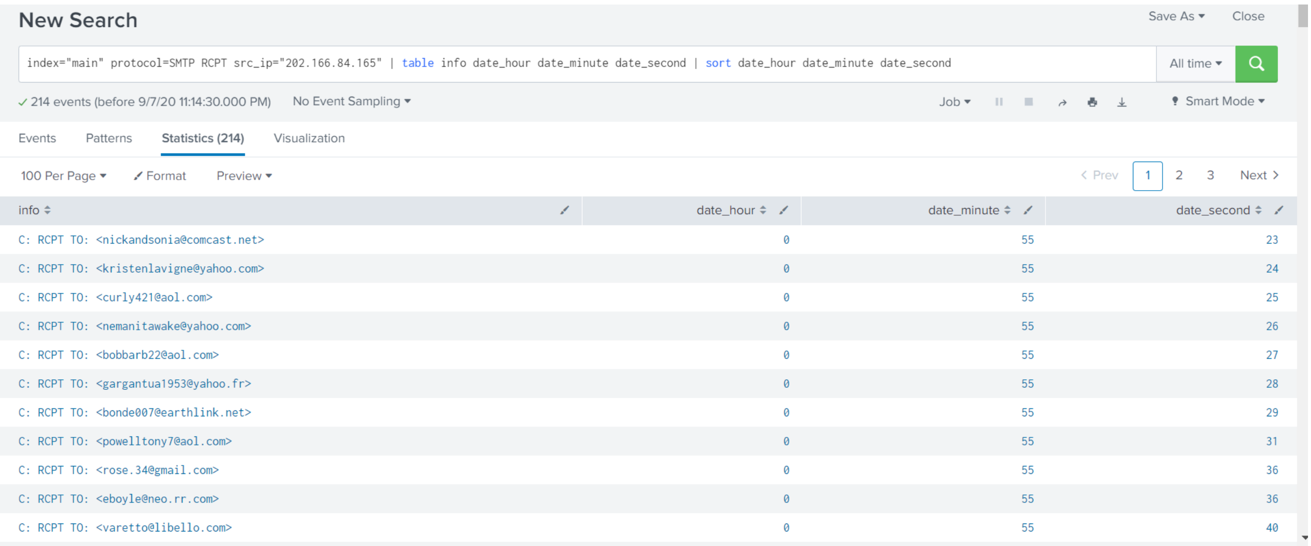
SPAM

Pattern: protocol=SMTP, RCPT command, src_ip="202.166.84.165"

The RCPT command is used to tell mail server who is the recipient of your message [2]. And emails are transfer by SMTP protocol. Moreover, we already know the compromised machine's IP is *202.166.84.165*, and we assume all the emails sent from this machine after the first botnet C2 attack and before the last botnet C2 attack is SPAM. Therefore, the email sent between 2020-06-19 00:55:21.316 and 2020-06-19 00:58:00.085 are SPAM.

I use the below command to find the traffic within specific time interval

```
index="main" protocol=SMTP RCPT src_ip="202.166.84.165" | eval botnetStartTime=strptime("2020-
06-19 00:55:21.316", "%Y-%m-%d %H:%M:%S") |eval botnetEndTime=strptime("2020-06-19
00:58:00.085", "%Y-%m-%d %H:%M:%S") | where(_time >= botnetStartTime AND _time <= botnetEndTime)
| table _time info | sort _time |dedup info
```



```
index="main" protocol=SMTP RCPT src_ip="202.166.84.165" | table info date_hour date_minute date_second | sort date_hour date_minute date_second
```

✓ 214 events (before 9/7/20 11:14:30.000 PM)

info	date_hour	date_minute	date_second
C: RCPT TO: <nickandsonia@comcast.net>	0	55	23
C: RCPT TO: <kristenlavigne@yahoo.com>	0	55	24
C: RCPT TO: <curly421@aol.com>	0	55	25
C: RCPT TO: <nemanitawake@yahoo.com>	0	55	26
C: RCPT TO: <bobbarb22@aol.com>	0	55	27
C: RCPT TO: <gargantua1953@yahoo.fr>	0	55	28
C: RCPT TO: <bonde007@earthlink.net>	0	55	29
C: RCPT TO: <powelltony7@aol.com>	0	55	31
C: RCPT TO: <rose.34@gmail.com>	0	55	36
C: RCPT TO: <eboyle@neo.rr.com>	0	55	36
C: RCPT TO: <varetto@libello.com>	0	55	40

Figure 5 - Email Sending History (partial)

As figure 5 shows, there are 22 email addresses targeted by this spam.
At the top and the end of the table, we can found the:

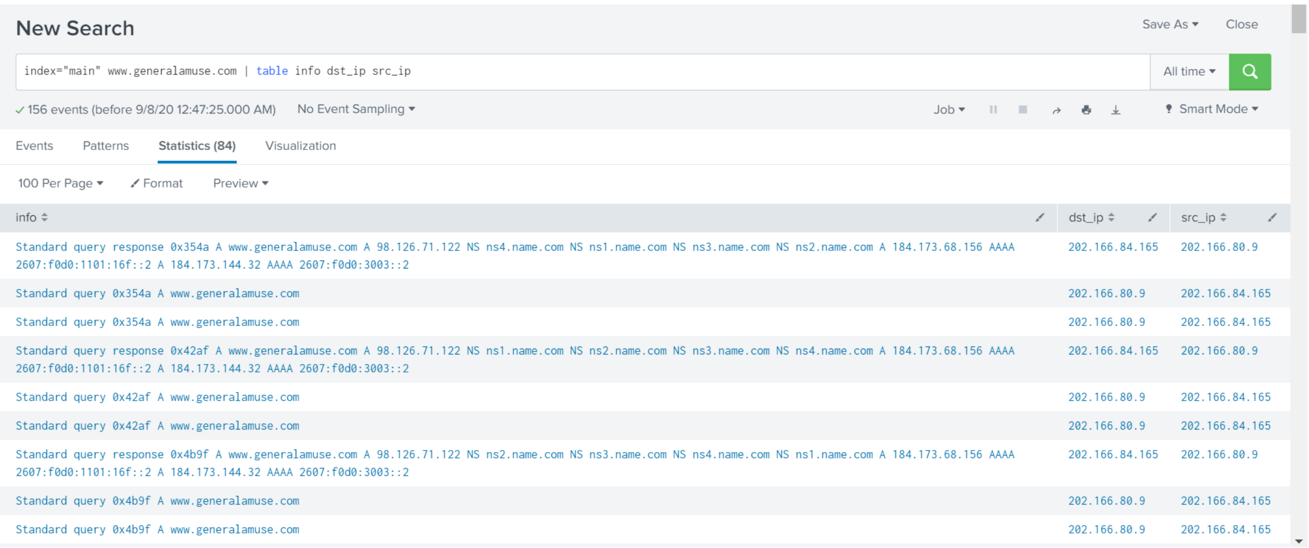
Email SPAM start time: 2020-06-19 00:55:23.278
First recipient: nickandsonia@comcast.net
Email SPAM end time: 2020-06-19 00:56:17.316
Last recipient: billmac@charter.net

ClickFraud

Pattern src_ip, dst_ip

To examine the ClickFraud, we need to identify the IP address of "www.generalamuse.com" first. We use below command to do that.

```
index="main" www.generalamuse.com | table info dst_ip src_ip
```



```
index="main" www.generalamuse.com | table info dst_ip src_ip
```

✓ 156 events (before 9/8/20 12:47:25.000 AM)

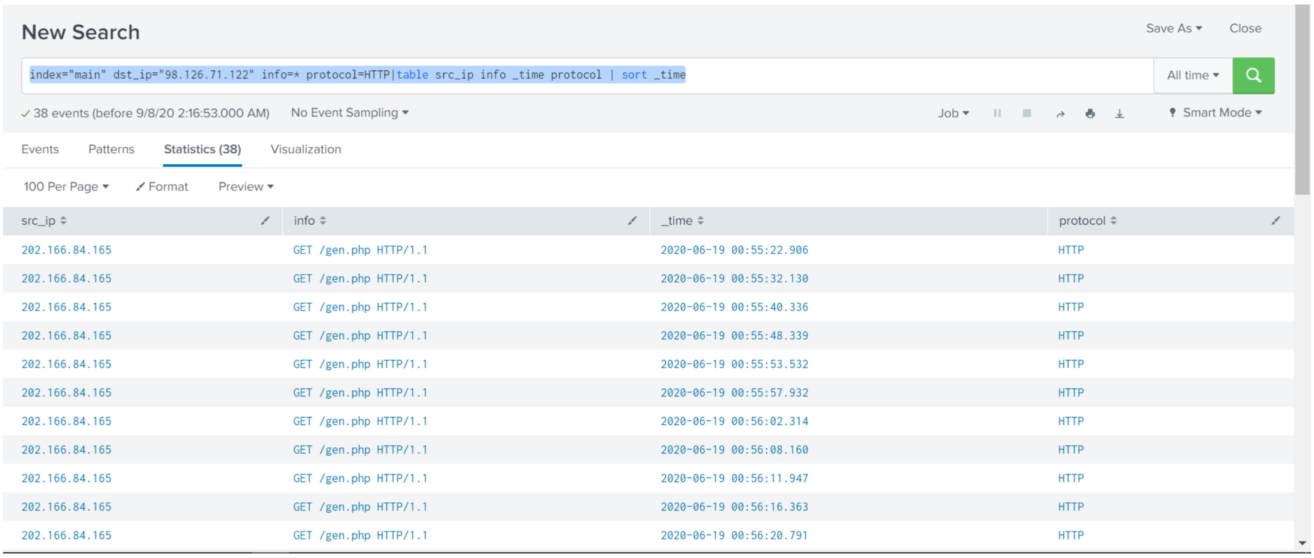
info	dst_ip	src_ip
Standard query response 0x354a A www.generalamuse.com A 98.126.71.122 NS ns4.name.com NS ns1.name.com NS ns3.name.com NS ns2.name.com A 184.173.68.156 AAAA 2607:f0d0:1101:16f::2 A 184.173.144.32 AAAA 2607:f0d0:3003::2	202.166.84.165	202.166.80.9
Standard query 0x354a A www.generalamuse.com	202.166.80.9	202.166.84.165
Standard query 0x354a A www.generalamuse.com	202.166.80.9	202.166.84.165
Standard query response 0x42af A www.generalamuse.com A 98.126.71.122 NS ns1.name.com NS ns2.name.com NS ns3.name.com NS ns4.name.com A 184.173.68.156 AAAA 2607:f0d0:1101:16f::2 A 184.173.144.32 AAAA 2607:f0d0:3003::2	202.166.84.165	202.166.80.9
Standard query 0x42af A www.generalamuse.com	202.166.80.9	202.166.84.165
Standard query 0x42af A www.generalamuse.com	202.166.80.9	202.166.84.165
Standard query response 0x4b9f A www.generalamuse.com A 98.126.71.122 NS ns2.name.com NS ns3.name.com NS ns4.name.com NS ns1.name.com A 184.173.68.156 AAAA 2607:f0d0:1101:16f::2 A 184.173.144.32 AAAA 2607:f0d0:3003::2	202.166.84.165	202.166.80.9
Standard query 0x4b9f A www.generalamuse.com	202.166.80.9	202.166.84.165
Standard query 0x4b9f A www.generalamuse.com	202.166.80.9	202.166.84.165

Figure 6 - Generalamuse.com IP (partial)

As we can see in the figure 6, the IP address of Generalamuse.com is `98.126.71.122`.

To find the number of ClickFraud requests, I set the protocol to "HTTP" to ignore TCP three-way handshakes.

```
index="main" dst_ip="98.126.71.122" info=* protocol=HTTP|table src_ip info _time protocol | sort _time
```


New Search

Save As ▾ Close

```
index="main" dst_ip="98.126.71.122" info=* protocol=HTTP|table src_ip info _time protocol | sort _time
```

All time ▾

✓ 38 events (before 9/8/20 2:16:53.000 AM) No Event Sampling ▾

Job ▾ Smart Mode ▾

Events Patterns Statistics (38) Visualization

100 Per Page ▾ Format Preview ▾

src_ip	info	_time	protocol
202.166.84.165	GET /gen.php HTTP/1.1	2020-06-19 00:55:22.906	HTTP
202.166.84.165	GET /gen.php HTTP/1.1	2020-06-19 00:55:32.130	HTTP
202.166.84.165	GET /gen.php HTTP/1.1	2020-06-19 00:55:40.336	HTTP
202.166.84.165	GET /gen.php HTTP/1.1	2020-06-19 00:55:48.339	HTTP
202.166.84.165	GET /gen.php HTTP/1.1	2020-06-19 00:55:53.532	HTTP
202.166.84.165	GET /gen.php HTTP/1.1	2020-06-19 00:55:57.932	HTTP
202.166.84.165	GET /gen.php HTTP/1.1	2020-06-19 00:56:02.314	HTTP
202.166.84.165	GET /gen.php HTTP/1.1	2020-06-19 00:56:08.160	HTTP
202.166.84.165	GET /gen.php HTTP/1.1	2020-06-19 00:56:11.947	HTTP
202.166.84.165	GET /gen.php HTTP/1.1	2020-06-19 00:56:16.363	HTTP
202.166.84.165	GET /gen.php HTTP/1.1	2020-06-19 00:56:20.791	HTTP

Figure 7 - ClickFraud Requests (partial)

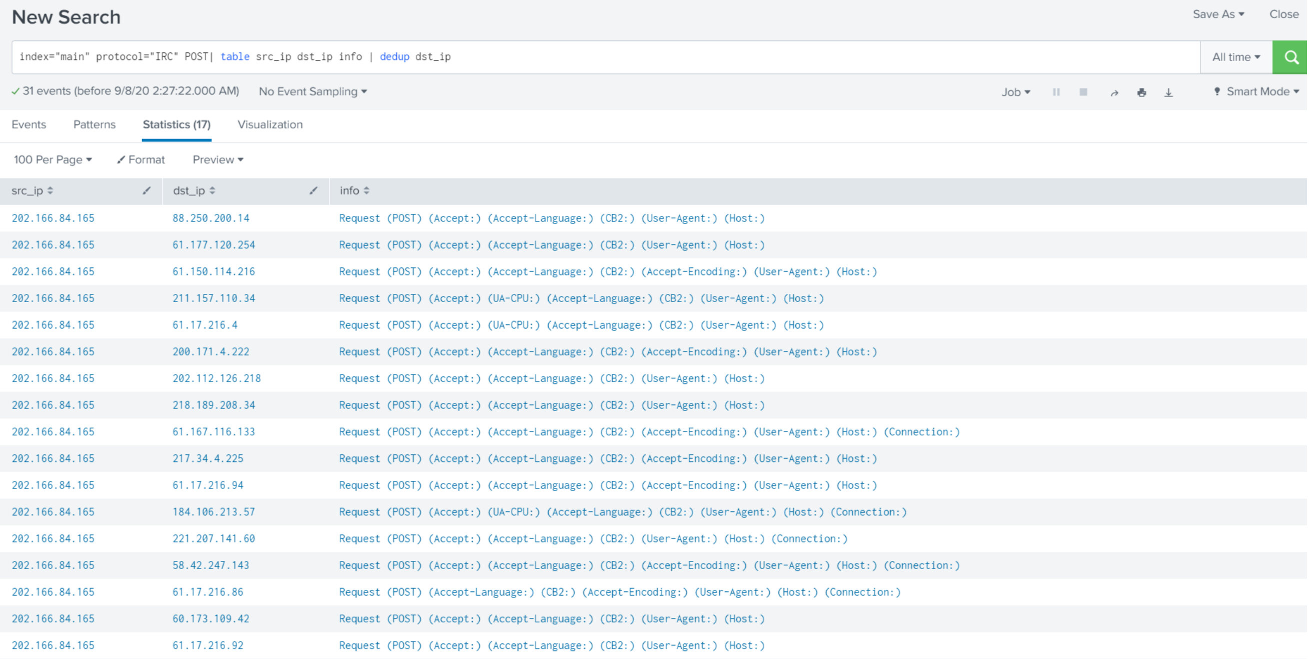
As we can see in figure 7, there are 38 ClickFraud requests with `GET /gen.php`URI string. And the start time is `2020-06-19 00:55:22.906`, the end time is `2020-06-19 01:01:08.208`.

IRC

Pattern: src_ip, protocol

To find the IRC server, we set the protocol to "IRC", search the key word "POST" and then delete the duplicate destination IP address by using the below command.

```
index="main" protocol="IRC" POST| table src_ip dst_ip info | dedup dst_ip
```


New Search

Save As ▾ Close

```
index="main" protocol="IRC" POST| table src_ip dst_ip info | dedup dst_ip
```

All time ▾

✓ 31 events (before 9/8/20 2:27:22.000 AM) No Event Sampling ▾

Job ▾ Smart Mode ▾

Events Patterns Statistics (17) Visualization

100 Per Page ▾ Format Preview ▾

src_ip	dst_ip	info
202.166.84.165	88.250.200.14	Request (POST) (Accept:) (Accept-Language:) (CB2:) (User-Agent:) (Host:)
202.166.84.165	61.177.120.254	Request (POST) (Accept:) (Accept-Language:) (CB2:) (User-Agent:) (Host:)
202.166.84.165	61.150.114.216	Request (POST) (Accept:) (Accept-Language:) (CB2:) (Accept-Encoding:) (User-Agent:) (Host:)
202.166.84.165	211.157.110.34	Request (POST) (Accept:) (UA-CPU:) (Accept-Language:) (CB2:) (User-Agent:) (Host:)
202.166.84.165	61.17.216.4	Request (POST) (Accept:) (UA-CPU:) (Accept-Language:) (CB2:) (User-Agent:) (Host:)
202.166.84.165	200.171.4.222	Request (POST) (Accept:) (Accept-Language:) (CB2:) (Accept-Encoding:) (User-Agent:) (Host:)
202.166.84.165	202.112.126.218	Request (POST) (Accept:) (Accept-Language:) (CB2:) (User-Agent:) (Host:)
202.166.84.165	218.189.208.34	Request (POST) (Accept:) (Accept-Language:) (CB2:) (User-Agent:) (Host:)
202.166.84.165	61.167.116.133	Request (POST) (Accept:) (Accept-Language:) (CB2:) (Accept-Encoding:) (User-Agent:) (Host:) (Connection:)
202.166.84.165	217.34.4.225	Request (POST) (Accept:) (Accept-Language:) (CB2:) (Accept-Encoding:) (User-Agent:) (Host:)
202.166.84.165	61.17.216.94	Request (POST) (Accept:) (Accept-Language:) (CB2:) (Accept-Encoding:) (User-Agent:) (Host:)
202.166.84.165	184.106.213.57	Request (POST) (Accept:) (UA-CPU:) (Accept-Language:) (CB2:) (User-Agent:) (Host:) (Connection:)
202.166.84.165	221.207.141.60	Request (POST) (Accept:) (Accept-Language:) (CB2:) (User-Agent:) (Host:) (Connection:)
202.166.84.165	58.42.247.143	Request (POST) (Accept:) (Accept-Language:) (CB2:) (Accept-Encoding:) (User-Agent:) (Host:) (Connection:)
202.166.84.165	61.17.216.86	Request (POST) (Accept-Language:) (CB2:) (Accept-Encoding:) (User-Agent:) (Host:) (Connection:)
202.166.84.165	60.173.109.42	Request (POST) (Accept:) (Accept-Language:) (CB2:) (User-Agent:) (Host:)
202.166.84.165	61.17.216.92	Request (POST) (Accept:) (Accept-Language:) (CB2:) (User-Agent:) (Host:)

Figure 8 - List of IRC Servers

Figure 8 lists all the IRC servers' IP addresses (in the dst_ip column). Totally, there are `17` IRC servers.

To find out the total number of requests, use the below command.

```
index="main" protocol="IRC" POST| stats count(info)
```

We found that there are total `31` POST requests made by the infected machine.

By using the command:

```
index="main" protocol="IRC" POST| table _time info src_ip dst_ip | sort _time
```

We can find out the start time is `2020-06-19 00:55:21.813`, and end time is `2020-06-19 01:01:59.756`.

Attack Narratives

The main victim's IP address in this case is *"202.166.84.165"*. It was compromised and installed a backdoor program to become a bot of C2 server *"finalcortex.com"* since *2020-06-19 00:55:21.316* until *2020-06-19 00:58:00.085*. During this time interval, this compromised machine had sent 53 POST request to the C2 server. Furthermore, the C2 server also use this compromised machine to send SPAM mail. Moreover, this compromised also suffer from ClickFraud since *2020-06-19 00:55:22.906* to *2020-06-19 01:01:08.208*. Lastly, this infected machine also be controlled to send multiple requests to 17 IRC servers since *2020-06-19 00:55:21.813* to *2020-06-19 01:01:59.759.

Consequence

Botnet Command & Control

The backdoor in the victim machine (the bot) would damage the whole network which this bot belongs to. The other machines within the network may still trust this victim machine, therefore, the bot may able to get confidential information from other machines, even able to spread malware to others. This botnet C2 attack breaks the confidentiality, integrity, and availability at the same time.

SPAM

The infected machine was controlled to send SPAM to other machine within the network. The spam emails may contain malware to infect other machines. The malware may be able to delete or steal files on machines; this break the confidentiality and integrity. Moreover, if the malware infect other machines, making server malfunctioned, it would also break the availability.

ClickFraud

This compromised machine was controlled to click an online advertisement many times. This will generate charge per click to the advertiser. Therefore, it will harm the advertiser and benefit the host website. ClickFraud may damage the availability of the advertisement in the targeted network, because the large amount of clicks on one advertisement may cause the server unable to burden. This would cause people who want to see the advertisement unable to access it.

IRC

IRC are widely used in botnet due to the simple, low bandwidth communication characteristics of IRC networks [3]. IRC botnets would damage the confidentiality, integrity, and availability as we discussed above.

Attack Pattern

- Botnet C2: `src_ip + dst_ip`
  In botnet C2 attack, we could see abnormally large number of commands from one server to multiple bots. Also, we could see multiple bots reply to a single C2 server. Therefore, source IP, destination IP, and the abnormally large traffic are the pattern to distinguish botnet C2 attack.
- SPAM: `src_ip + protocol + RCPT command`

Emails are usually sent by POP3, SMTP, or IMAP protocol. If we observe an abnormal large number of email sent from single source IP with these 3 protocol, then it is highly possible a SPAM attack. Also, RCPT command is used in email transaction; therefore, it can also be used to help identify the SPAM attack.

- ClickFraud: `src_ip + dst_ip`
  If we see a numerous requests sent from single machines to single advertiser in a short period, it is likely a ClickFraud.
- IRC: `src_ip + protocol`

  IRC server could be identified by the IRC protocol and the source IP address connect to IRC server are usually an infected machine.

Countermeasures (Detection and Mitigation)

- Botnet C2 attack:
  Botnet C2 attack can be detected by monitoring abnormally large network traffic from single source IP. This single source IP often connects to numerous destinations (bots). If we can find the botnet server, we can block all the bots it connects with or cut the connection between bots and C2 servers, therefore mitigate the damage.
- SPAM:
  SPAM are usually contain exaggerated content and sent from unreliable source. Therefore, we could use the subject, content and source IP to filter the SPAM.

- ClickFraud:
  An advertiser could block the ClickFraud by blocking specific IP addresses who click the advertisement many times within a short period. Therefore, monitoring unusual network traffic could detect and resolve the ClickFraud.
- IRC:
  Attack through IRC could be detected and block by pre-defined protocol rule.

Reference

[1] Azeria-Labs. 2020. *Command And Control*. [online] Available at: https://azeria-labs.com/command-and-control/ [Accessed 7 September 2020].

[2] Users.cs.cf.ac.uk. 2020. *The RCPT Command*. [online] Available at: https://users.cs.cf.ac.uk/Dave.Marshall/PERL/node177.html [Accessed 7 September 2020].

[3] “Botnet,” *Wikipedia*, 03-Sep-2020. [Online]. Available: https://en.wikipedia.org/wiki/Botnet#IRC. [Accessed: 07-Sep-2020].