

Amplification Honeypot Project

Aleksandra Shuianova¹, Artem Rodimov², and Nikula
Miro-Markus³

¹Grenoble INP, Ensimag, France

²Grenoble INP, Ensimag, France

³Aalto University, Finland

June 1, 2022

1 Introduction

A distributed denial-of-service (DDoS) attack occurs when multiple systems flood the bandwidth or resources of a targeted system, usually one or more web servers. A DDoS attack uses more than one unique IP address or machines, often from thousands of hosts infected with malware. A distributed denial of service attack typically involves more than around 3–5 nodes on different networks.

Amplification attacks are used to magnify the bandwidth that is sent to a victim. This is typically done through publicly accessible DNS servers that are used to cause congestion on the target system using DNS response traffic. Many services can be exploited to act as reflectors, some harder to block than others. DNS amplification attacks involve a new mechanism that increased the amplification effect, using a much larger list of DNS servers than seen earlier. The process typically involves an attacker sending a DNS name look up request to a public DNS server, spoofing the source IP address of the targeted victim. The attacker tries to request as much information as possible, thus amplifying the DNS response that is sent to the targeted victim. Since the size of the request is significantly smaller than the response, the attacker is easily able to increase the amount of traffic directed at the target. SNMP and NTP can also be exploited as reflector in an amplification attack.

It is very difficult to defend against these types of attacks because the response data is coming from legitimate servers. These attack requests are also sent through UDP, which does not require a connection to the server. This means that the source IP is not verified when a request is received by the server. In order to bring awareness of these vulnerabilities, campaigns have been started that are dedicated to finding amplification vectors which has led to people fixing their resolvers or having the resolvers shut down completely.

2 Project Goals

To implement the Amplification Honeypot project, the following goals were set:

1. Rent servers in 5 countries using the DigitalOcean service;
2. On each server deploy an AmpPot honeypot specifically designed to monitor amplification DDoS attacks;
3. Collect data about attacks and analyze them: identify patterns, identify technologies used by attackers (protocol type, request length), identify anomalies;
4. Visualize the received statistics.

3 AmpPot

This section begins with background information on what AmpPot is and what it is used for. This is followed by a description of the installation and configuration of AmpPot on the droplet. And finally, we briefly describe what AmpPot should output.

3.1 Background

AmpPot honeypot is specifically designed to monitor amplification DDoS attacks. AmpPot is able to mimic one or more services that are known to be vulnerable to amplification attacks, such as DNS and NTP. AmpPot mimics services having amplification attack vectors by listening on UDP ports that are likely to be abused. To make them attractive to attackers, it sends back legitimate responses. Attackers will try to abuse your honeypot as an amplifier, which allows you to observe ongoing attacks, their victims, and the DDoS techniques.

3.2 Setting up AmpPot

Before setting up AmpPot we checked that our system is with:

- a public static IPv4 address
- root access (raw sockets)
- python3.6 or higher
- a few GB of disk space for logs (usually between 0.5 and 1.5 GB/day)

Then, we used the following command to install AmpPot:

```
python3 -m pip install --upgrade --extra-index-url
https://YOUR_KEY@projects.cispa.saarland/api/v4/
projects/964/packages/pypi/simple amppot
```

The above command installs the AmpPot honeypot, as well as a configuration utility.

After that, we used a command:

```
setup-amppot
```

to create the initial configuration and service file for the honeypot.

In order for AmpPot to work in the background, you need to run the command:

```
systemctl enable amppot
```

This command will allow you to run AmpPot at system startup.

Finally, we edited a *config.cfg* file: disabled feed back to the servers of Saarland University by default, set 'attackMonitorServer'/'homeServer' to '127.0.0.1'.

3.3 Output of AmpPot

As a result, AmpPot generates the following folders: *db*, *log*, *pktdumps*, *unknown_req*. The *db* folder contains sqlite files by day, where all the information about requests collected by AmpPot on a particular day is recorded as it shown in Figure (1).














 2022-04-03-ddos-honeypot.db	04.04.2022 2:15	Data Base File	28 KB
 2022-04-04-ddos-honeypot.db	05.04.2022 2:00	Data Base File	12 620 KB
 2022-04-05-ddos-honeypot.db	06.04.2022 2:00	Data Base File	15 232 KB
 2022-04-06-ddos-honeypot.db	07.04.2022 2:04	Data Base File	13 156 KB
 2022-04-07-ddos-honeypot.db	08.04.2022 2:00	Data Base File	7 708 KB
 2022-04-08-ddos-honeypot.db	09.04.2022 2:00	Data Base File	20 020 KB
 2022-04-09-ddos-honeypot.db	10.04.2022 2:00	Data Base File	25 764 KB
 2022-04-10-ddos-honeypot.db	11.04.2022 2:00	Data Base File	33 144 KB
 2022-04-11-ddos-honeypot.db	12.04.2022 2:00	Data Base File	38 208 KB
 2022-04-12-ddos-honeypot.db	13.04.2022 2:00	Data Base File	38 128 KB
 2022-04-13-ddos-honeypot.db	14.04.2022 2:00	Data Base File	33 504 KB
 2022-04-14-ddos-honeypot.db	15.04.2022 2:00	Data Base File	32 524 KB
 2022-04-15-ddos-honeypot.db	16.04.2022 2:00	Data Base File	49 128 KB

Figure 1: List of .db files by day generated by AmpPot

Each *.db* file contains 3 tables: *honeypot_runs*, *requests*, *responses*. In Figures (2)(3), a *honeypot_runs* table is illustrated which contains a basic information about each honeypot session launched on a particular day.

	Honeypot_id	start_time	end_time	monitoredips	max_requests	wait_time	reset_time
	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	ensimag2022_ind	2022-05-01 00:00:00	2022-05-02 00:00:00	159.89.163.150	3	60	3600

Figure 2: Honeypot runs table, part 1

From this table, we can conclude that on May 1, 2022, AmpPot was launched once and received requests all day.

In Figures (4)(5), a *requests* table is illustrated which contains an information about each request on a particular day.

QOTD_enabled	CharGen_enabled	SSDP_enabled	NTP_enabled	DNS_enabled	GenProxy_enabled
Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
true	true	true	true	true	false

Figure 3: Honeypot runs table, part 2

	Honeypot_id	Version	HL	TOS	Lenght	Id	TTL	Protocol	IP_Checksum	Source_Addr	Dest_Addr	Source_Port	Dest_Port
	Фильтр	Фильтр	Фи...	Фи...	Фильтр	Фил...	Фи...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	ensimag2022_ind	4	20	0	79	57892	244	17	62513	5.32.168.55	159.89.163.150	9852	389
2	ensimag2022_ind	4	20	0	36	39980	243	17	12359	23.224.160.133	159.89.163.150	3934	123
3	ensimag2022_ind	4	20	0	36	61167	244	17	56451	23.224.160.133	159.89.163.150	5001	123
4	ensimag2022_ind	4	20	0	79	27646	244	17	27224	5.32.168.55	159.89.163.150	25235	389
5	ensimag2022_ind	4	20	0	36	45824	241	17	552	122.114.87.62	159.89.163.150	46573	123
6	ensimag2022_ind	4	20	0	79	18375	244	17	36495	5.32.168.55	159.89.163.150	25235	389
7	ensimag2022_ind	4	20	0	79	7013	244	17	47857	5.32.168.55	159.89.163.150	9852	389
8	ensimag2022_ind	4	20	0	36	6462	244	17	2025	45.207.52.227	159.89.163.150	4215	123
9	ensimag2022_ind	4	20	0	79	56054	244	17	64351	5.32.168.55	159.89.163.150	25235	389
10	ensimag2022_ind	4	20	0	79	52735	244	17	2135	5.32.168.55	159.89.163.150	25235	389

Figure 4: Requests table, part 1

Dest_Port	Lenght_UDP_Header	UDP_Checksum	Request_md5	Request_metadata	Request_time
Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
389		59	0 09fc5870063009da22860d2fe24b26a5	LDAP Search request	2022-05-01 00:00:00.338
123		16	0 4bbbd4470b8551d0b04839c052e32c60	NTP monlist request. Length: 8	2022-05-01 00:00:00.406
123		16	0 4bbbd4470b8551d0b04839c052e32c60	NTP monlist request. Length: 8	2022-05-01 00:00:00.408
389		59	0 09fc5870063009da22860d2fe24b26a5	LDAP Search request	2022-05-01 00:00:00.529
123		16	0 4bbbd4470b8551d0b04839c052e32c60	NTP monlist request. Length: 8	2022-05-01 00:00:00.603
389		59	0 09fc5870063009da22860d2fe24b26a5	LDAP Search request	2022-05-01 00:00:00.736
389		59	0 09fc5870063009da22860d2fe24b26a5	LDAP Search request	2022-05-01 00:00:00.746
123		16	0 4bbbd4470b8551d0b04839c052e32c60	NTP monlist request. Length: 8	2022-05-01 00:00:01.186
389		59	0 09fc5870063009da22860d2fe24b26a5	LDAP Search request	2022-05-01 00:00:01.418
389		59	0 09fc5870063009da22860d2fe24b26a5	LDAP Search request	2022-05-01 00:00:01.420

Figure 5: Requests table, part 2

Based on the data from this table, we will further analyze the requests, investigate anomalies and technologies used by the attackers.

4 Digital Ocean

Digital ocean is one of many cloud service providers. In general, Digital Ocean doesn't provide as many services and options as for example Google cloud, but it is easy to use and, in some cases, more affordable. Setting up virtual machines (in Digital Ocean, VM's are called droplets) is very easy. Only operating system image, location and a plan have to be selected. We chose to deploy five different droplets in five countries: Russia, Netherlands, U.S, Singapore and India. The goal was to scatter the servers as widely around the world as possible using the available locations. The droplets ran Ubuntu 20.04 LTS and had 1GB memory, 25GB disk and transfer limit of 1TB. The droplets can be accessed via SSH or from the terminal of the Digital Ocean GUI. A useful feature that we utilized is team feature. Digital Ocean allows to create teams for projects, which allow every member of the team to easily access project resources.

5 Data processing

The data that AmpPot outputs must be downloaded to a local computer. We used SCP program that allows to download files via SSH from our remote servers. To do this, you need to add the client's public key to the droplets. A Python script was developed to automate routine tasks. It allows to download data from servers, unpack the received archives and convert sqlite files into csv format. In Figure (6), you can see the main menu of the script.

```
Select option:
1) Download logs
2) Extract logs
3) Convert sqlite to csv
0) Exit
```

Figure 6: Main menu

At first, we need to download data from our remote servers. As mentioned above, the SCP utility was used for this. The list of servers is stored in a JSON

file, which can be seen in Figure (7). An IP address and an identifier are stored for each server. The script creates an archive with files on remote server to reduce the size and number of downloaded files. After downloading created archives are removing. The name of archives contains the date and the server ID. To store archives, a folder is created with the upload date in the name. The output of the script downloading the archives can be seen in Figure (8).

```
{
  "servers": [
    {
      "ip": "45.135.233.2",
      "name": "ru"
    },
    {
      "ip": "64.227.69.245",
      "name": "nl"
    },
    {
      "ip": "159.89.163.150",
      "name": "in"
    },
    {
      "ip": "139.59.98.134",
      "name": "sg"
    },
    {
      "ip": "143.198.142.20",
      "name": "us"
    }
  ]
}
```

Figure 7: JSON file containing servers

The next feature of the script is to unpack all downloaded archives. The program will ask you to select a folder with downloaded archives. After that, the process of unpacking all files inside the selected directory will begin. The output of the script is shown in Figure (9).

```
1
file - amppot_20220507_ru.tar.gz downloaded
file - amppot_20220507_nl.tar.gz downloaded
file - amppot_20220507_in.tar.gz downloaded
file - amppot_20220507_sg.tar.gz downloaded
file - amppot_20220507_us.tar.gz downloaded
Select option:
1) Download logs
2) Extract logs
3) Convert sqlite to csv
0) Exit
```

Figure 8: Downloaded archives

For the analyzing the collected data, it is necessary to convert them it CSV format. This action is also automated using a script. We need to select the folder with the unpacked data and the conversion process will begin. The conversion process is shown in Figure (10).

The script is developed using the libraries "tarfile", "sqlite3" and "csv", which are build-in the current version of the Python interpreter and do not require the installation of additional dependencies. To work with SSH and SCP, the commands available from the terminal were used. To work correctly, we need to install an SSH client and add the path to the executable files to the environment variables. Thus, the script turned out to be cross-platform.

```

2
Select directory:
0) .git
1) .idea
2) archives_20220502
3) archives_20220507
4) venv
5) __pycache__
3
file ./archives_20220507/amppot_20220507_in.tar.gz extracted!
file ./archives_20220507/amppot_20220507_nl.tar.gz extracted!
file ./archives_20220507/amppot_20220507_ru.tar.gz extracted!
file ./archives_20220507/amppot_20220507_sg.tar.gz extracted!
file ./archives_20220507/amppot_20220507_us.tar.gz extracted!
Select option:
1) Download logs
2) Extract logs
3) Convert sqlite to csv
0) Exit

```

Figure 9: Archives unpacking

```

Select option:
1) Download logs
2) Extract logs
3) Convert sqlite to csv
0) Exit
3
Select directory:
0) .git
1) .idea
2) archives_20220502
3) archives_20220507
4) venv
5) __pycache__
3
Exported: ./archives_20220507/amppot_20220507_in/db/2022-05-02-ddos-honeypot.db
Lines exported: 315279
Exported: ./archives_20220507/amppot_20220507_in/db/2022-05-03-ddos-honeypot.db
Lines exported: 611231
Exported: ./archives_20220507/amppot_20220507_in/db/2022-05-04-ddos-honeypot.db
Lines exported: 939593

```

Figure 10: The process of converting to CSV

6 Amplification DDoS attack analysis using Python

6.1 Attack definition

Attacks consist of individual requests. An attack was defined as requests from the same source address with no more than 60 minutes between consecutive requests. In addition, attacks with less than 100 requests were left out to filter scanners and normal internet traffic. These factors to define an attack are the same as used in [1]. The difference is that in this paper, a single attack can use multiple ports and protocols instead of just one.

We used Pandas-library when working with tabular csv-files. Pandas offers a groupby-method which allows us to group the individual requests by their source address. The tens of millions of individual requests were combined into under 240 000 attacks. We also extracted the honeypot id's, numbers of requests, time-to-live values, durations, lengths, protocols and ports as information describing the attacks.

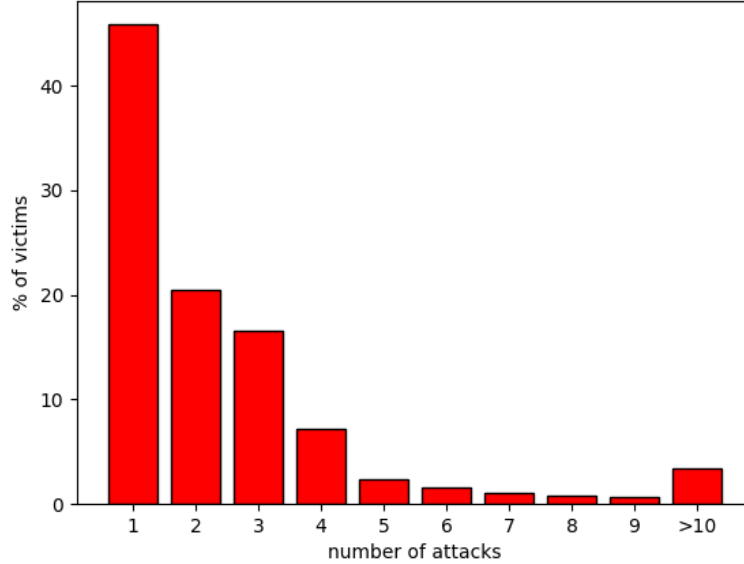


Figure 11: Number of attacks per victim

6.2 Attacks per victim

In the observed timespan there were in total 237794 attacks towards 77114 individual victims in all the five honeypots combined. Figure (11) shows the percentage of victims that were attacked respective number of times. It can be seen that about 45 % of victims only experienced one attack and only over 3 % of victims were target to 10 or more attacks. Almost half of the victims experienced only one attack and therefore most of the attacks can be classified as one-off attacks. However, there are a few victims that gathered hundreds of attacks, with the largest number being 796 attacks.

The proportion of victims attacked only once is significant. In this case it should also be kept in mind that an attack is defined as requests using the same source IP-address. Not as requests to the same /16 network. With a few of the victims to the most attacks being located in the US and in Europe, by far the most attacks were targeted to China. Especially Chinese technology giants Tencent and Alibaba were the most common internet service providers of the attacked networks.

Matplotlib.pyplot-library offers different types of diagrams to be used like for example the barplot above.

6.3 Attack duration

Duration of the attacks was measured by the time between the first and the last request of an attack. Majority of the attacks were short-lived with over 70 % of the attacks lasting less than five minutes. However, there were again

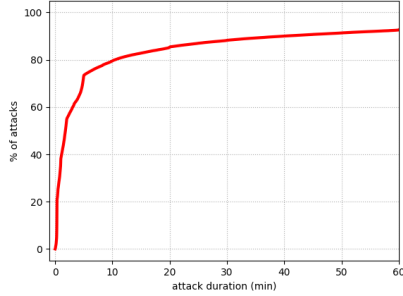


Figure 12: Cumulative distribution of attack durations.

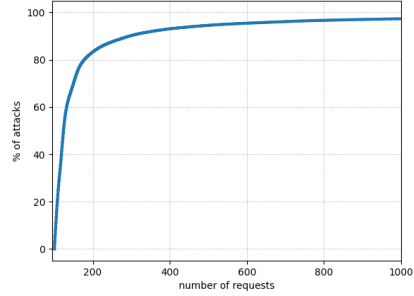


Figure 13: Cumulative distribution of number of requests.

some values differing from the majority. The longest attack lasted for almost 215 hours. Cumulative distribution of attack durations is shown in figure (12).

Some of the best-known DDoS attacks have lasted for days and even for months [2]. For that reason, the short duration of most of the attacks was a little surprising. However, an attack of just a couple of minutes could be enough to disrupt a service so that a customer switches to a competitor's service. Like for example with some online stores. When locating the victims of the longest lasting attacks, it was found that most of them are in Hong Kong and are again IP-addresses provided by Tencent.

The cumulative distribution function can be constructed using Pyplot together with Numpy-library's sort and linspace method. Limits :

```
plt.plot(np.sort(durations), np.linspace(0, 100, len(durations)))
```

6.4 Requests per attack

The minimum number of requests to be considered as an attack is 100 and thus, most of the attacks consisted of a bit over 100 requests. Figure (13) portrays the cumulative distribution of number of requests per attack. 85 % of attacks had under 200 requests and 90 % of attacks under 300. The vast majority of attacks were quite small in terms of number of requests, but the largest ones had over 300 000 requests. On the contrary to the victims of most or longest attacks, the victims of the largest attacks were more diverse and located mainly in different parts of Europe and Asia. Noteworthy is also that the targets of 2 of the 3 largest attacks are in Ukraine.

6.5 Protocols

Different protocols have different amplification rates. Amplification rate means the factor by which the response from server is larger than the corresponding request. At least one of seven different internet protocols were used in 99.9 % of the attacks. The usage of these protocols is shown in table (1). By far the most common protocol was NTP, with 73.55 % of the attacks using it. NTP is a very

Protocol	Usage (%)
NTP	73.55
LDAP	16.47
SSDP	8.78
DNS	0.56
CharGen	0.52
QOTD	0.04
Memcached	0.04

Table 1: Protocols used in attacks.

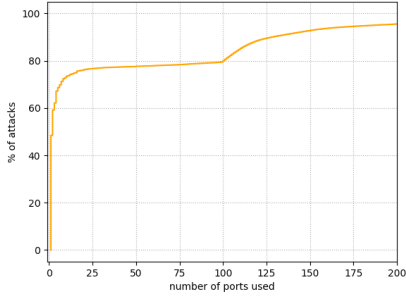


Figure 14: Cumulative distribution of number of ports used.

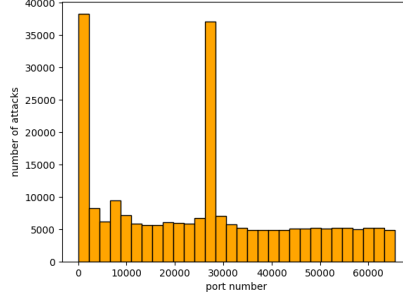


Figure 15: Histogram of port number distribution.

popular protocol to be used in amplification attacks mainly due to its Monlist-request property. Monlist requests information about the latest machines that the server has interacted with. This way, NTP responses can amplify the request length by a factor from 557 up to 4670 [3]. Following in popularity were LDAP and SSDP which can amplify traffic by factors of 55 and 76 respectively [3][4]. From the list of used protocols, the one with the highest amplification rate is Memcached, which has been reported to amplify a request by a factor of up to 50 000. This number is in the class of its own when it comes to amplification in contrast to the other protocols. It is interesting that a protocol with this kind of amplification capability was not used in more attacks. The reason most likely is that it is not as well-known as some of the other amplifiers. Although, Memcached-amplifier was introduced almost 15 years ago, its capabilities have only been discovered quite recently. In addition, most networks didn't even have the bandwidth required to handle traffic as large as produced by Memcached 10 years ago. [5]

As it was much more common for attacks to use only one protocol, there were also some attacks using multiple protocols. 97 % of attacks stuck to exploiting only one internet protocol as an amplifier and 3 % of the attacks used multiple protocols.

6.6 Ports

Figure (14) shows the cumulative distribution of number of ports used. There is a lot of variation in the number of used source ports per attack. Half of the attacks targeted only one port of the victim and about 15 % targeted between 100 and 200 different ports. Intuitively, attacking more ports is a more surefire way to attack as it is rather easy for the victim to deny traffic from single ports.

Figure (15) shows that port distribution is quite even for the most parts, except for two ports. The most common source ports used are port number 27015 with 12.6 % -, and port 80 with 9.6 % of the attacks' targets. Port 80 is no surprise as it is one of the most used ports in normal internet traffic. Port 27015 is typically used by Steam-gaming servers. The attacks targeting port 27015 were also scattered quite evenly for all the five honeypots.

6.7 Attack sources

Denial of service attacks can be sent from single or multiple sources like for example a botnet. As attacks are using the IP-addresses of targets as the source address for the requests, it is impossible to determine attackers based on source addresses. Instead, time-to-live (TTL)-tag was used to find out if an attack comes from an individual source or a network of attackers. Time-to-live is an integer value that decreases by one every time an IP-packet gets forwarded in a network. This way of separating the attack sources is not completely accurate, as the network structure can change during attacks and also the attackers can manually modify the TTL-value. However, this way we can get a good intuition of the proportions of single- and multi-source attacks. In [1], threshold of 2 distinct TTL-values is used to distinct an attack from single source and therefore was also used in this paper. This threshold allows one change in the TTL-value of a single-source attacks and thus provides some flexibility. It was found that 9.5 % of the attacks came from multiple sources whereas 90.5 % stemmed from a single source. The maximum number of different TTL-values for an attack was 21, which means that there were no attacks from very large botnets.

6.8 Conclusion

72 million requests were combined into almost 240 000 attacks. The majority of the attacks were short lived with 70 % lasting less than five minutes. In some cases, short attacks seem to be enough to disrupt a service enough for a user to switch to another service. Attacks with number of requests close to 100 were the most common. Almost half of the victims were attacked only once and only about 3 % of the victims were target to 10 or more attacks. This suggests that one attack is enough most times for the attacker. Victims of most and longest attacks are in China and the largest attacks in terms of request quantity were targeted to Ukraine. Most port numbers were used evenly except for port 80 and port 27015 which combined into over 20 % of the attacks. Half of the attacks used more than one port. Over 90 % of attacks stem from a single source, and therefore seem to pose a greater threat in terms on amplification DDoS attacks.

7 Data analysis using Elastic Stack

Elastic Stack allows to take data from any source, in any format, then search, analyze, and visualize it in real time.

Elasticsearch is a distributed, RESTful search and analytics engine capable of addressing a growing number of use cases.

Kibana is a free and open user interface that lets us visualize our Elasticsearch data and navigate the Elastic Stack.

Logstash is a free and open server-side data processing pipeline that ingests data from a multitude of sources, transforms it, and then sends it storage.

Elastic Stack (Elasticsearch, Kibana and Logstash) is launched using docker. Official docker images are used for this. Each application runs in a separate container, which are connected by a virtual network.

Data analysis is a resource-intensive task that cannot be run on our remote servers. To do this, a virtual machine was created with ubuntu installed. But this approach does not allow other participants to connect to Kibana. An SSH tunnel is used for this. The user sends a request to the droplet, which will be redirected through the tunnel to the VM, and then to the docket container. The response to the request comes along the same path, only in reverse order. The request diagram can be seen in Figure (16).

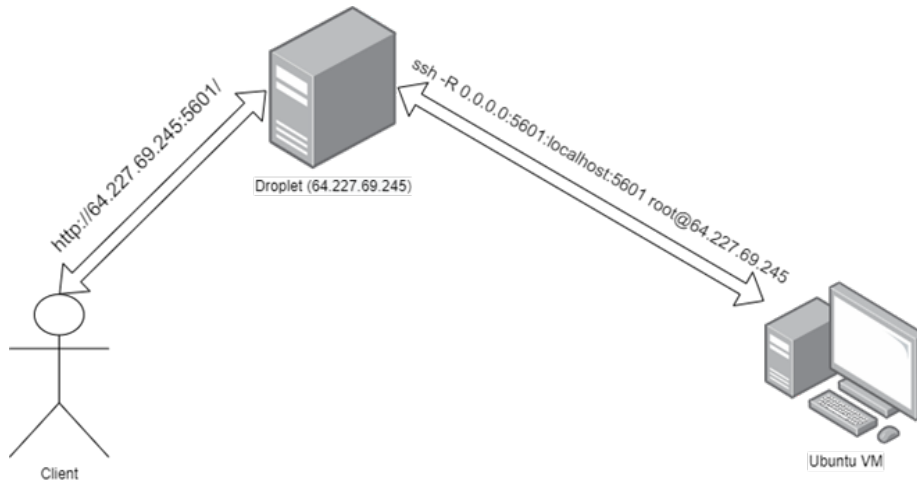


Figure 16: Request through droplet

The collected data in the form of CSV is processed using Logstash. In order for Logstash to read all the collected files, it is necessary to specify the path template to them as in Figure (17).

```

input {
  file {
    path => "/usr/share/logstash/amppot/**/*.*.csv"
    start_position => "beginning"
  }
}
  
```

Figure 17: Logstash input

The read files must be processed in the filter section. It is necessary to specify the format of the read file as CSV and the names of the columns to be saved. The date section contains a field that contains information about the document date, date format, and timezone. Also, each numeric field must be converted to the "integer" type. Next, you can delete unnecessary embedded fields. The collected data contains the IP addresses of the victims, Logstash can use this value to determine the country, city and other information. There is a "geoip" section for this. The described part of the configuration file is shown on Figure (18).

We need to save filtered data. In output section we write that our data should store in Elasticsearch and specify node, index credentials. This part of configuration shown on Figure (19).

The collected data is stored in Elasticsearch under the index "amppot". As a result of the work of the servers, 74224469 requests for 13.9 GB were collected. This information is shown on Figure (20).

```

filter {
  csv {
    separator => ","
    autodetect_column_names => true
    skip_header => true
    columns => ["Honeypot_id", "Version", "IHL", "TOS",
    "Lenght", "Id", "TTL", "Protocol", "IP_Checksum", "Source_Addr",
    "Dest_Addr", "Source_Port", "Dest_Port", "Lenght_UDP_Header",
    "UDP_Checksum", "Request_md5", "Request_metadata", "Request_time"]
  }
  date {
    match => ["Request_time", "yyyy-MM-dd HH:mm:ss.SSS"]
    target => "@timestamp"
    timezone => "Europe/Paris"
  }
  mutate {convert => ["Version", "integer"]}
  mutate {convert => ["IHL", "integer"]}
  mutate {convert => ["TOS", "integer"]}
  mutate {convert => ["Lenght", "integer"]}
  mutate {convert => ["Id", "integer"]}
  mutate {convert => ["TTL", "integer"]}
  mutate {convert => ["Protocol", "integer"]}
  mutate {convert => ["Source_Port", "integer"]}
  mutate {convert => ["Dest_Port", "integer"]}
  mutate {convert => ["Lenght_UDP_Header", "integer"]}
  mutate {convert => ["UDP_Checksum", "integer"]}
  mutate {
    remove_field => ["[event][original]", "message", "Request_time", "[host][name]", "@version"]
  }
  geoip {
    source => "Source_Addr"
    target => "Source_Addr_Geo"
  }
}

```

Figure 18: Logstash filter

```

output {
  elasticsearch {
    hosts => ["https://172.18.0.2:9200"]
    ssl => true
    cacert => '/usr/share/logstash/certs/http_ca.crt'
    api_key => "8MrdgYABbYf6fIp8M6vQ:5zFd9dKVSciPYS_ymYWkA"
    index => "ampspot"
  }
}

```

Figure 19: Logstash output

<input type="checkbox"/> Name	Health	Status	Primaries	Replicas	Docs count	Storage size	Data stream
<input type="checkbox"/> .items-default-000001	● yellow	open	1	1	0	225b	
<input type="checkbox"/> .lists-default-000001	● yellow	open	1	1	0	225b	
<input type="checkbox"/> amppot	● yellow	open	1	1	74224469	13.92gb	

Figure 20: AmpPot index

Figure (21) shows a diagram containing the ratio of collected requests by each server separately. The Russian server collected the most requests (27.25%), followed by a server from the Netherlands (23.94%), a server from the USA (23.27%), from Singapore (14.21%) and India (11.33%).

Figure (22) shows a diagram containing the ratio of collected requests by victims. The most attacks are made on servers in the USA (27.75%), China (18.84%), France (4.33%) and Russia (4.13%). Other countries make up 11.8%, which indicates a wide variety of countries to be attacked and their percentage is less than 1.

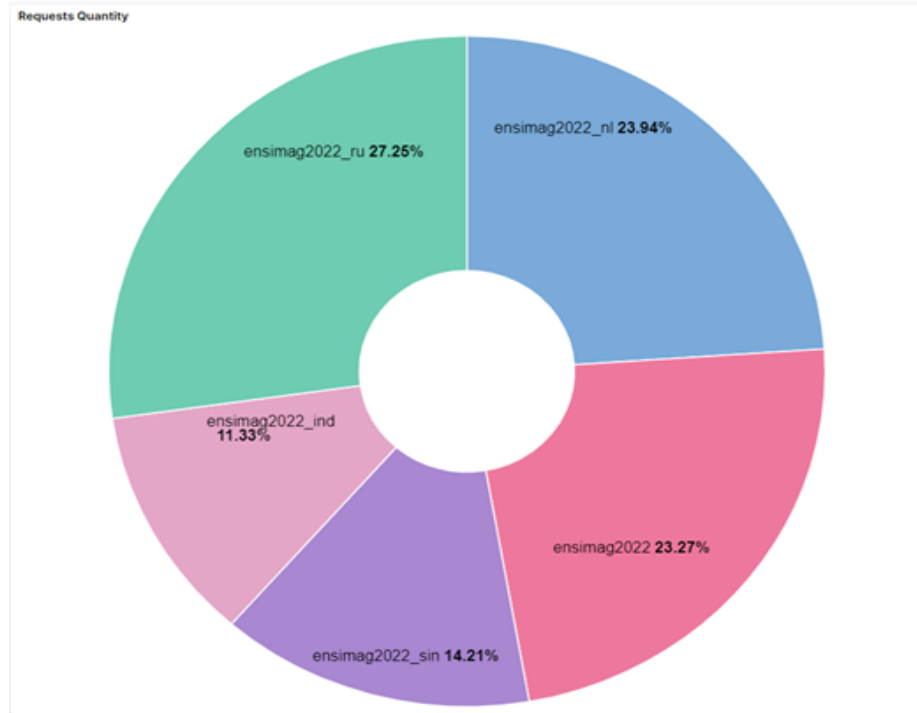


Figure 21: Requests collected by honeypots

Figure (23) shows a diagram of requests by day. From April 4 to April 22, all 5 servers were working, so it was during this period that the most requests were collected. We can also notice an increase in the number of requests every day after launch. This is due to the detection of the server by scanners. Every day more and more scanners discover a new server. The number of requests also depends on the bandwidth of the servers. The server from Russia was running on the “ruvds” platform, which provided wider bandwidth compared to droplets. The Dutch server also has wider bandwidth than the rest of the droplets. The server from the USA worked the longest, so it collected almost as much as the server from Amsterdam.

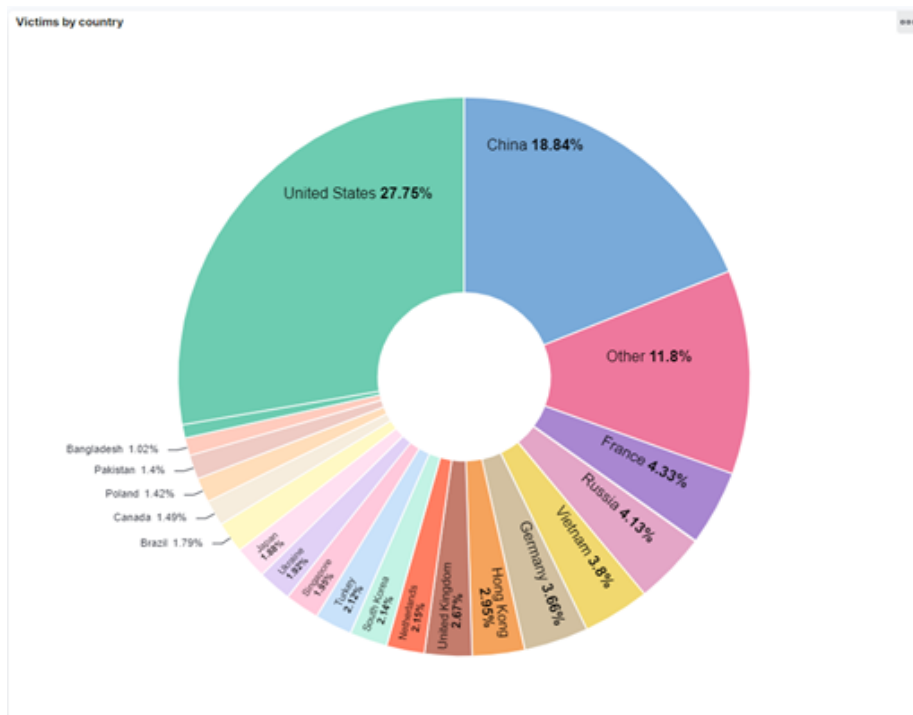


Figure 22: Requests collected by victims

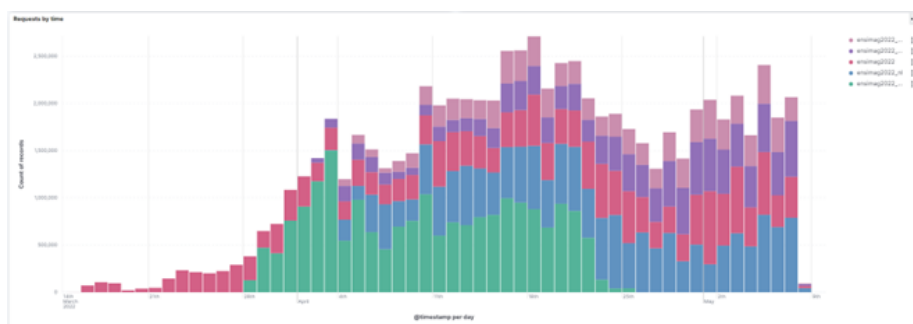


Figure 23: Requests by days

Figure (24) shows a diagram of requests by protocols. The most commonly used are NTP (78.81%), LDAP (9.91%) and SSDP (9.07%).

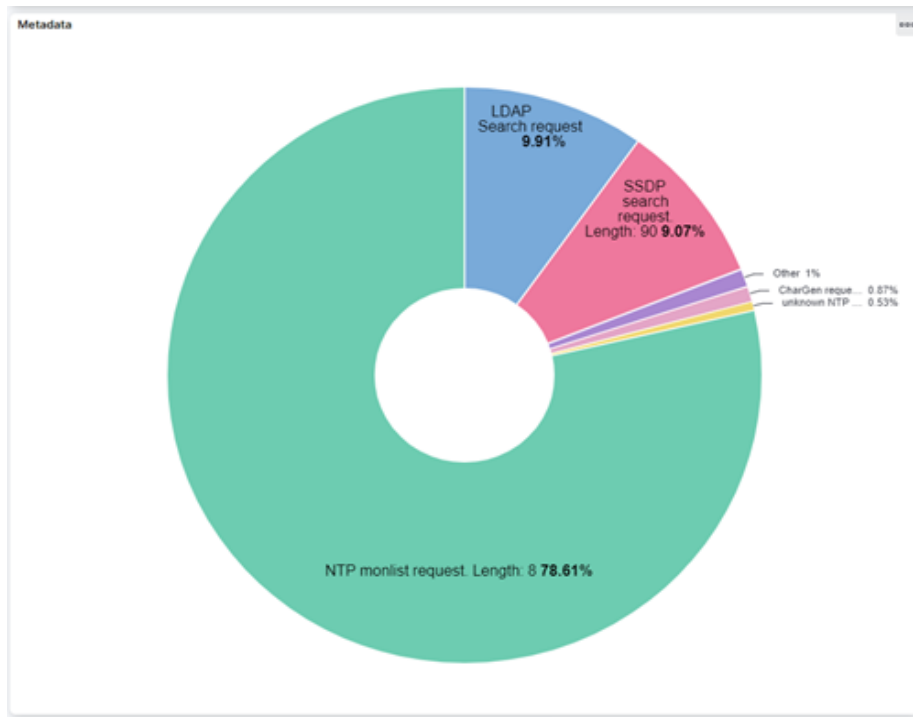


Figure 24: Requests by protocols

Kibana contains unsupervised learning technics for anomaly detection in data. For our dataset we utilized multi-metrics and population methods.

In multi-metric we tried to analyze requests count per day and influence of it by IP address and county of victim. The results of this method is shown on Figures (25-28). This method allows us to detect anomaly on 29 of March where 37.38% of requests were made with target country as Ukraine. (Shown on Figure (29))

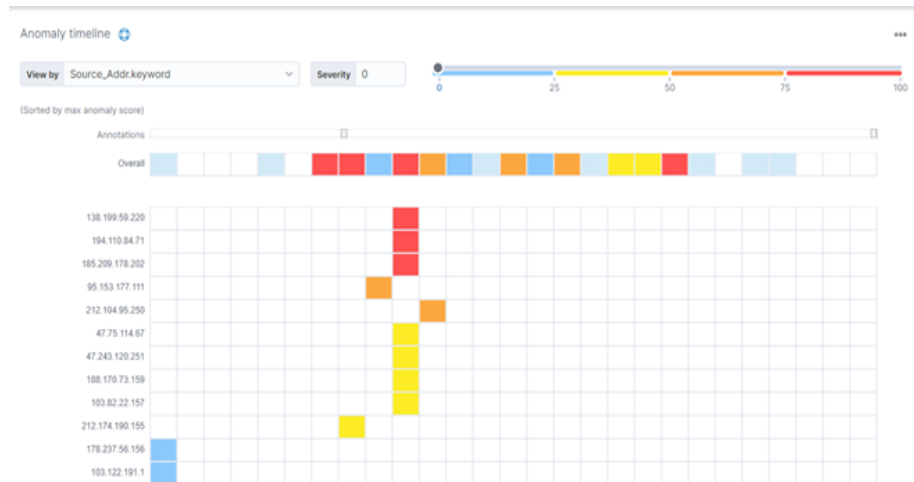


Figure 25: Anomalies timeline by multi-metrics method

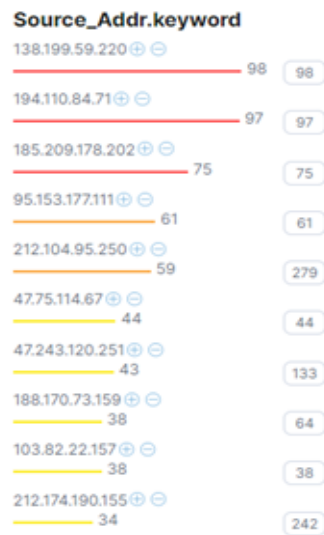


Figure 26: Top anomalies by IP by multi-metrics method

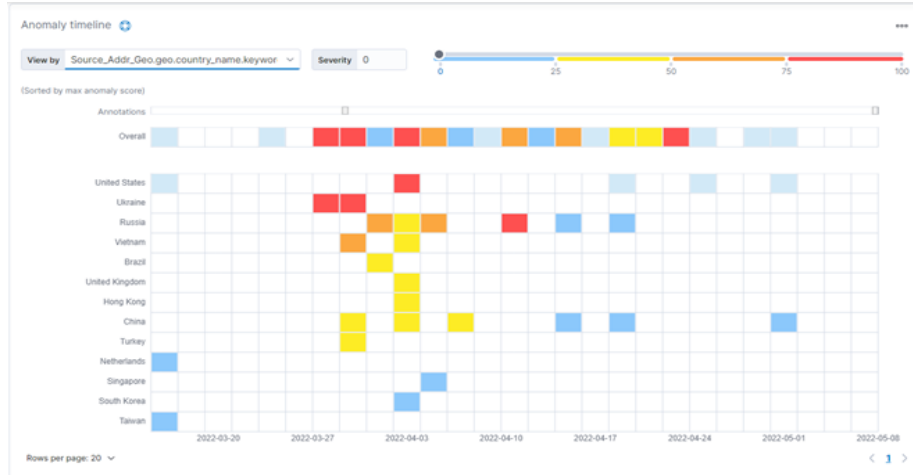


Figure 27: Anomalies timeline by country name by multi-metrics method

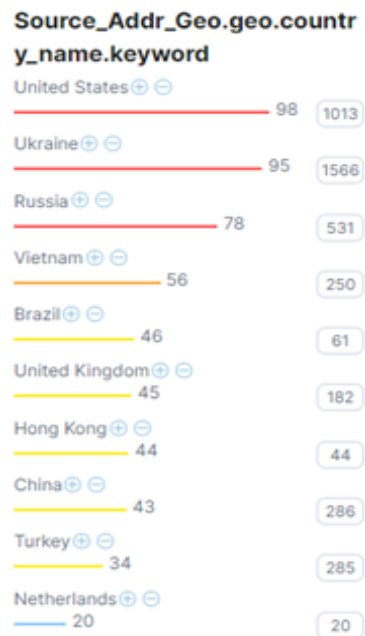


Figure 28: Top anomalies by country name by multi-metrics method

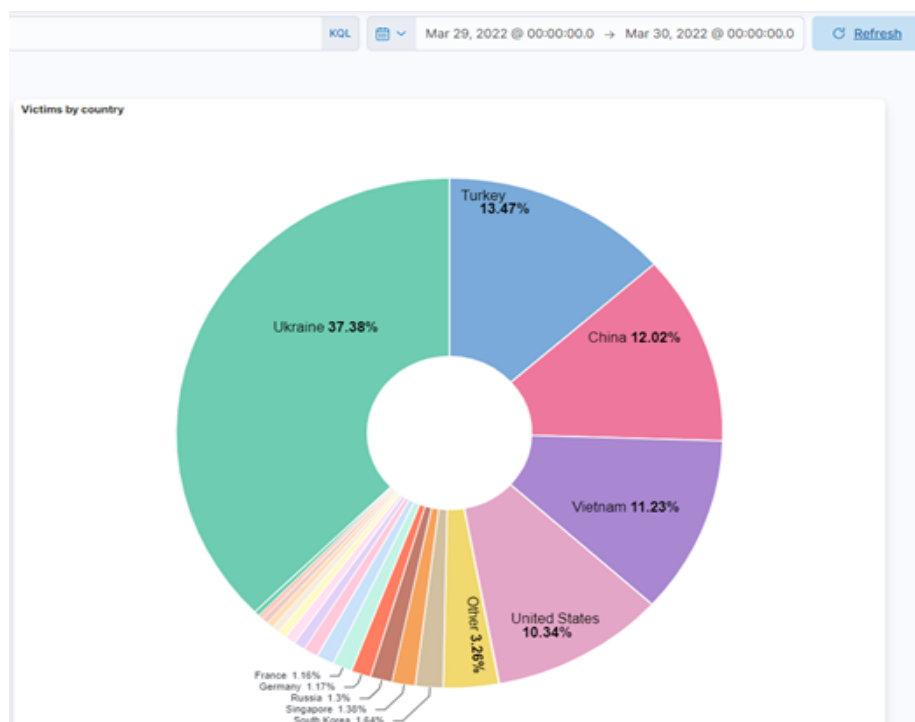


Figure 29: Attacks anomaly in Ukraine

We also tried to use population method where populations separate by country name and tried detect anomalies in counts per day in different populations (the same aim like in multi-metric). We create 2 jobs. The first analyze full data (Shown on figures (30 and 31)) and the second analyze data from 4.04 to 22.04 because in this period all 5 servers collection data (Shown on figures (32 and 33)).

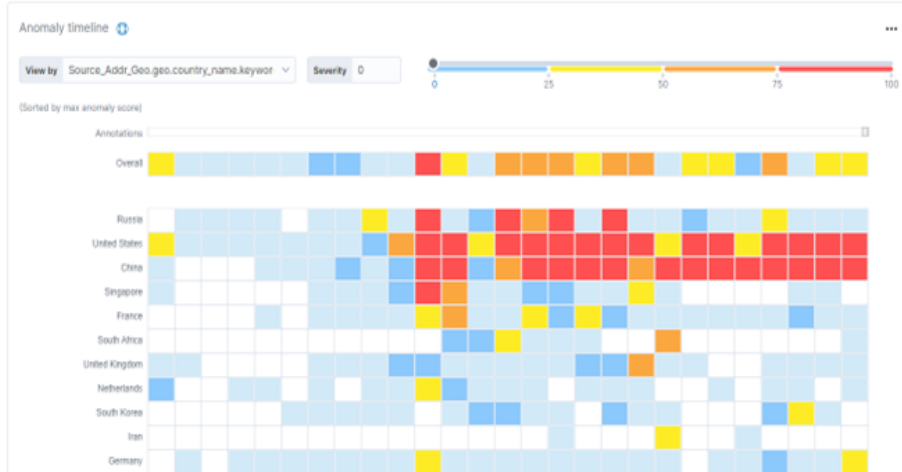


Figure 30: Anomalies timeline by population method on full data

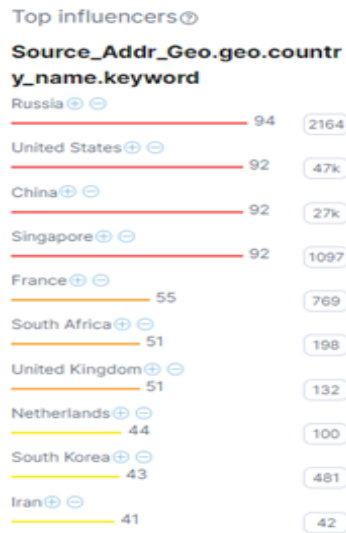


Figure 31: Top anomalies by population method on full data

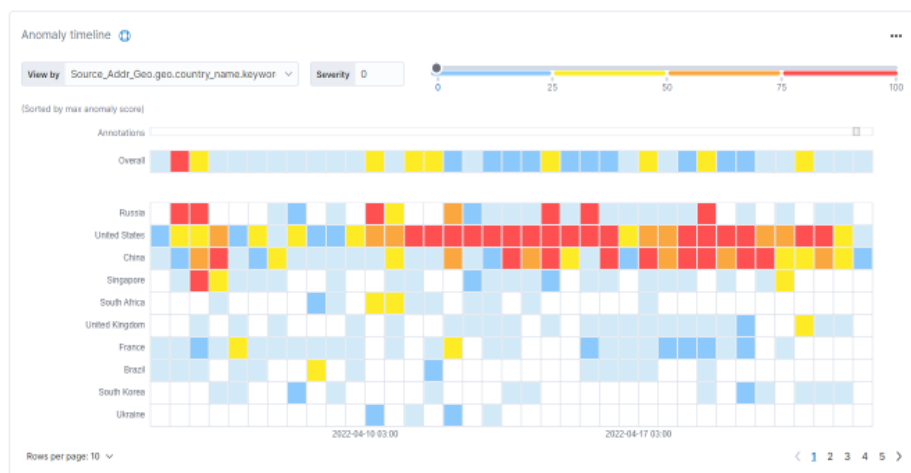


Figure 32: Anomalies timeline by population method



Figure 33: Top anomalies by population method

It's possible to notice that population method detects more anomalies than multi-metrics and give them the higher score. This fact means that for example we can't detect anomaly in Ukraine which was shown before.

8 Conclusion

Amplification attacks continue to be a dangerous threat to millions of users. AmpPot assists in analyzing the amplification threats in more detail. We placed the AmpPot service on 5 servers in 5 countries around the world and analyzed the results collected by the service. As a result, we were able to identify anomalies in some countries where the number of requests on certain days increased sharply, identify the most frequently used protocol by hackers, and collect many other useful statistics.

References

- [1] Krämer et al. AmpPot: Monitoring and Defending Against Amplification DDoS Attacks, 2015. Available in: https://krebsonsecurity.com/wp-content/uploads/2016/10/amppot-raid2015.Brian_.pdf
- [2] Diego Asturias. The Most Notorious DDoS Attacks in History – 2021 Update. Cloudbric, 2021. Available in: <https://en.cloudbric.com/blog/2021/04/most-notorious-ddos-attacks-in-history-2021-update/>
- [3] C. Rossow. Amplification Hell: Revisiting Network Protocols for DDoS Abuse, 2014. Available in: <https://christian-rossow.de/publications/amplification-ndss2014.pdf>
- [4] Catalin Cimpanu. CLDAP Protocol Allows DDoS Attacks with 70x Amplification Factor. Bleepingcomputer, 2017. Available in: <https://www.bleepingcomputer.com/news/security/cldap-protocol-allows-ddos-attacks-with-70x-amplification-factor/>
- [5] Qrator labs. Understanding the facts of memcached amplification attacks. Medium, 2018. Available in: <https://qratorlabs.medium.com/understanding-the-facts-of-memcached-amplification-attacks-729611547244>