FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

# Assignment of bachelor's thesis

| | |
|---|---|
| **Title:** | Botnets and Distributed Denial-of-Service attacks |
| **Student:** | Jan Lukáš |
| **Supervisor:** | Ing. Alexandru Moucha, Ph.D. |
| **Study program:** | Informatics |
| **Branch / specialization:** | Computer Security and Information technology |
| **Department:** | Department of Computer Systems |
| **Validity:** | until the end of summer semester 2023/2024 |

## Instructions

Get closer to the problematic of Distributed Denial-of-Service attacks (DDoS) and their connection to botnets.
Focus on DDoS attacks and document how they can be categorised and what differences are between them. What resources they need to operate, what is their target, how they are controlled and coordinated and how it all connects to the problematic of botnets.
Describe what botnets are and explain how they operate in a simplified manner.
Choose a few DDoS attacks (e.g. SYN flood) and describe their functioning and possible protections against them. Next, design a model in which selected attacks could be executed, their effectiveness measured, and protections against them tested. Make a testing network by implementing this model either by using a simulation, building a real network, or both.
Analyse how the attacks are influencing your network. Deploy previously discussed protections and repeat the measurements.
Compare the results of the measurements and discuss their implications.

*Electronically approved by prof. Ing. Pavel Tvrdík, CSc. on 12 November 2022 in Prague.*

Bachelor's thesis

# BOTNETS AND DISTRIBUTED DENIAL-OF-SERVICE ATTACKS

**Jan Lukáš**

Faculty of Information Technology
Computer Security and Information technology
Supervisor: Ing. Alexandru Moucha, Ph.D.
May 10, 2023

Citation of this thesis: Lukáš Jan. *Botnets and Distributed Denial-of-Service attacks*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

# Contents

# List of Figures

# List of Tables

# List of code listings

*I would like to express my heartfelt gratitude to Ing. Alexandru Moucha, Ph.D., the supervisor of this thesis, for his invaluable inspiration and guidance that enabled me to complete this work. His lectures have been a great source of motivation and have introduced me to the fascinating world of computer networking. I am also deeply grateful to my friends whose unwavering support has been instrumental in my three-year journey at FIT CTU. Lastly, I extend my sincere appreciation to my parents and sister, whose support and assistance have been invaluable beyond measure.*

# Declaration

## Abstract

This bachelor's thesis aims to investigate and describe various types of DDoS attacks. The literature review begins by introducing fundamental concepts and progresses to provide a thorough definition of botnets, DoS attacks, and their corresponding countermeasures. Leading to specific examples of DDoS attacks, which are classified based on the pre-established classification. The thesis also presents potential countermeasures for mitigating the identified attacks. Notably, the work emphasizes the possibility of selecting countermeasures based on the attack classification, which is a significant outcome of this research.

Following the literature review, the thesis describes a test model used to illustrate DDoS attacks. This involves deploying a botnet and executing DDoS attacks on a web page. The experiment is then repeated after implementing a chosen countermeasure, and the results are analyzed and compared.

Overall, this thesis provides a comprehensive understanding of DDoS attacks, including their objectives, vulnerabilities, and practical examples. Readers will gain valuable insights into the types of DDoS attacks and what a basic DDoS attack may look like.

**Keywords** DoS & DDoS attacks, DDoS & DoS countermeasures, Botnet, DDoS classification

## Abstrakt

Cílem této bakalářské práce je výzkum a popis vybraných útoků typu DDoS. Literární rešerše začíná od základních pojmů a postupně se přez detailní definici botnetu, DoS útoků a jejich protiopatření dostavá ke konkrétním případům DDoS útoků. Vybrané útoky řadí podle dříve definované klasifikace a uvádí možná řešení na jejích mitigaci. Jedním z hlavních výstupů práce je následná úvaha nad otázkou výběru protiopatření podle klasifikace útoku.

Následně je popsán testovací model, který je použit na představení DDoS útoků nasazením botnetu a provedení DDoS útoků na webovou stránku. Měření jsou následně opětovně provedena po nasazení vybraného protiopatření a výsledky porovnány.

Čtenář po přečtení získá znalosti z oblasti DDoS útoků od cílů a slabých míst různých typů DDoS útoků po praktické znalosti, jak může jednoduchý DDoS útok vypadat.

**Klíčová slova** DoS a DDoS útoky, Ochrany proti DDoS a DoS, Botnet, klasifikace DDoS

# List of abbreviations

| | |
|---|---|
| ATP | Advanced Threat Protection |
| C&C | Command and Control |
| CPU | Central Processing Unit |
| DDoS | Distributed Denial of Service |
| DMZ | Demilitarized Zone |
| DNS | Domain Name System |
| FTP | File Transfer Protocol |
| HTT | Hypertext Transfer Protocol |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| ICMP | Internet Control Message Protocol |
| IDS | Intrusion Detection System |
| IP | Internet Protocol |
| IRC | Internet Relay Chat |
| ISO | International Organization for Standardization |
| LTE | Long-Term Evolution |
| MIB | Management Information Base |
| MTA | Mail Transfer Agent |
| MTU | Maximum Transmission Unit |
| OS | Operating System |
| OSI | Open Systems Interconnection |
| P2P | Peer-to-Peer |
| PID | Process Identifier |
| PPS | Packets Per Second |
| SIP | Session Initiation Protocol |
| SMTP | Simple Mail Transfer Protocol |
| SSL | Secure Sockets Layer |
| TCP | Transmission Control Protocol |
| TCP RFC | Transmission Control Protocol Request for Comments |
| TLS | Transport Layer Security |
| UDP | User Datagram Protocol |
| URL | Uniform Resource Locator |
| VLAN | Virtual Local Area Network |
| VoIP | Voice over Internet Protocol |
| WAF | Web Application Firewall |

# Introduction

In today's world, we are surrounded by online services like online banking, online shops, and online school systems. State-funded online services are also becoming more and more frequent. Providing convenient ways to communicate with state authorities. Every such service is part of a bigger system, that's responsible for keeping the service available for its clients. With how effective and convenient these services are some providers shifted their services only to this online form. And so they became a crucial means of income for providers with a natural need for high security against dangerous attacks like DoS attacks. By targeting vulnerable parts of providers' systems a DoS attack makes the service unresponsive or even takes it down.

As targets of these attacks are whole corporations or states, attackers are likewise backed by corporations or states resulting in corporate warfare or cyber warfare. Meaning both the attack and defense sides are well funded, resulting in every attack being properly researched making it unique often targeting different resources with different methods. With a diverse range of DoS attacks comes a myriad of different approaches and defensive strategies to prevent them. Proper defense mechanisms can be anything from applying a patch to redesigning a part of the company network so it can handle the pressure from a DoS attack. This means, that taking proper measures to counter DoS attacks is a difficult task.

This thesis aims at providing security researchers with a broad knowledge of how different DoS attacks work and what are the possible defense mechanisms. Making it easier to choose proper strategies to avoid and negate them.

We will start with a basic explanation of DoS followed by an in-depth classification of DoS, providing us with means to detect parts of our system, that DoS attacks may target. Later we will focus on the inner operations of a special attack network called botnets responsible for launching many DoS attacks. After that, we will focus on the classification of DoS defense mechanisms followed by an analysis of the more common DoS attack with our established taxonomies and specific countermeasures. The last part of this thesis will give a few examples by preparing a testing environment and measuring simple DoS attacks.

Chapter 1

# Introduction to DDoS and DoS problematics

*The first chapter introduces the problematics of DoS and DDoS attacks and why companies and states must be prepared and build defenses against them. Because botnets are a necessity for launching DDoS attacks, it suggests a basic explanation of their structure and functioning. Each explanation is on a basic level, which will be further developed in the following chapters.*

## 1.1 Dangers of Denial of Service

Special services like online transactions provided by online banking systems can be considered critical since denial of such service can mean a massive monetary loss for the service provider. Taking down a critical service may also lead to even more severe consequences than monetary loss as attacks targeting the healthcare sector or the army are not uncommon. To properly defend and negate DoS attacks we must first understand their basics.

▶ **Definition 1.1.** *(Denial of Service) Denial of Service (DoS) is a state describing that a service is unavailable to its users.*

If DoS is a result of malicious actions its purpose is essentially causing monetary loss to the provider every time the service would be used or providing some other benefits to the attacking party. To create a DoS effect the Denial of Service attacks need to target vulnerable parts of the systems, which are responsible for keeping the service available.

The definition of DoS attacks is often simplified to attacks on websites resulting in the website being nonoperational for some time. This is because websites providing customer service(banking systems, online shops) are the most common targets of DoS attacks. In this thesis, we assume that our DoS attacks are carried on over the internet as it's the most common practice.

IT Security experts often use broader definitions as shown in the following citations, because DDoS encompasses many different attacks with different targets, techniques, and goals.

*"A Denial of Service attack is an attack that attempts to block access to and use of a resource. It is a violation of availability. DOS (or DoS) attacks include flooding, connection exhaustion, and resource demand."* [1]

*"A Denial of Service attack is an attack resulting in the prevention of authorized access to resources or the delaying of time-critical operations. (Time-critical may be milliseconds or it may be hours, depending upon the service provided.)"* [2]

DoS attacks can originate from one or more points. These points of origin are devices generating internet traffic directly responsible for causing the DoS effect. Measuring the number of these devices is the simplest classification of DoS. Single point-of-origin DoS attacks are not very common, because they mostly require a presence of an exploitable bug on the targeted system. On the other hand, Distributed Denial of Service (DDoS) attacks, which are DoS attacks with multiple attack sources are much more frequent.

DDoS is a subcategory of DoS and as such will be properly defined later in DoS classification. Essentially it's a DoS attack, where the attacker operates an attack network (botnet) so the source of the attack is a network, not one device. Once again the reader can use definitions cited from the Committee on National Security Systems Glossary and the Cybersecurity glossary of terms for better understanding.

*"Distributed Denial of Service is a Denial of Service technique that uses numerous hosts to perform the attack"* [2]

*"Distributed Denial of Service attack is an attack that attempts to block access to and use of a resource. It is a violation of availability. DDoS is a variation of the DoS attack and can include flooding attacks, connection exhaustion, and resource demand. The distinction between DDoS and DoS is that the attack traffic may originate from numerous sources or is reflected or bounced off of numerous intermediary systems. The purpose of a DDoS attack is to significantly amplify the level of the attack beyond that which can be generated by a single attack system in order to overload larger and more protected victims. DDoS attacks are often waged using botnets."* [1]

As stated in the citation flooding attacks, connection exhaustion attacks(bandwidth attacks), and resource depletion attacks can be considered the main categories of DDoS attacks and will be disgusted later.

To conclude, the main danger of DoS for companies is a monetary loss, and for states, it's losing a critical part of the infrastructure. And so proper defense is needed on all state and corporate levels.

## 1.2 Introduction to botnets

The prerequisite for performing a DDoS attack is having multiple devices (points of origin) taking part in the DoS attack. Such a device (bot) is created by infecting a vulnerable device with a malicious code called bot binary.

▶ **Definition 1.2.** *(Botnet) A botnet is a special network composed of infected devices (bots) whose numbers can even approach millions of individual devices.*

A botnet by itself is just a powerful tool with high computational powers. Its purpose and actions are determined by a person called a botmaster. DDoS attacks are just one of the possible actions botnets can perform. Communication between the C&C server and bots is needed to instruct the bots on what actions should they perform. This communication is called Command & Control (C&C) communication. Botmaster can command the botnet from a C&C server or one of the nodes of the botnet. This depends on what kind of structure the botnet has. In most cases, devices comprising the botnet can be separated into bots, C&C servers, and stepping-stones, which are devices between them often with special functions. Botnets are also using a myriad of evasion techniques to stay undetected.

The reader can once again compare the author's explanation with definitions from other sources.

*"Botnet is a collection of innocent computers which have been compromised by malicious code in order to run a remote control agent granting an attacker the ability to remotely take*

*advantage of the system's resources in order to perform illicit or criminal actions. These actions include DoS flooding attacks, hosting false Web services, spoofing DNS, transmitting SPAM, eavesdropping on network communications, recording VOIP communications, and attempting to crack encryption or password hashes. Botnets can be comprised of dozens to over a million individual computers. The term botnet is a shortened form of robotic network."* [1]

*"A bot, originating from the term 'robot', is an application that can perform and repeat a particular task faster than a human. When a large number of bots spread to several computers and connect through the Internet, they form a group called a botnet, which is a network of bots. A botnet comes from three main elements - the bots, the command and control (C & C) servers, and the botmasters. A bot is designed to infect targets (e.g. computers or mobiles) and make them a part of a botnet without their owners' knowledge under the control of a person, known as the botmaster. The botmaster sends orders to all the bots on infected targets and controls the entire botnet through the Internet and the C & C servers. The botmasters try to get control of these targets and carry out their malicious activities."* [3]

The following figure shows an example of a botnet performing a DDoS attack on the victim's server. Botnet is divided into two parts with the left side using a centralized structure and the right decentralized structure. Botnet's structure, functioning, and individual parts will be discussed later in greater detail.

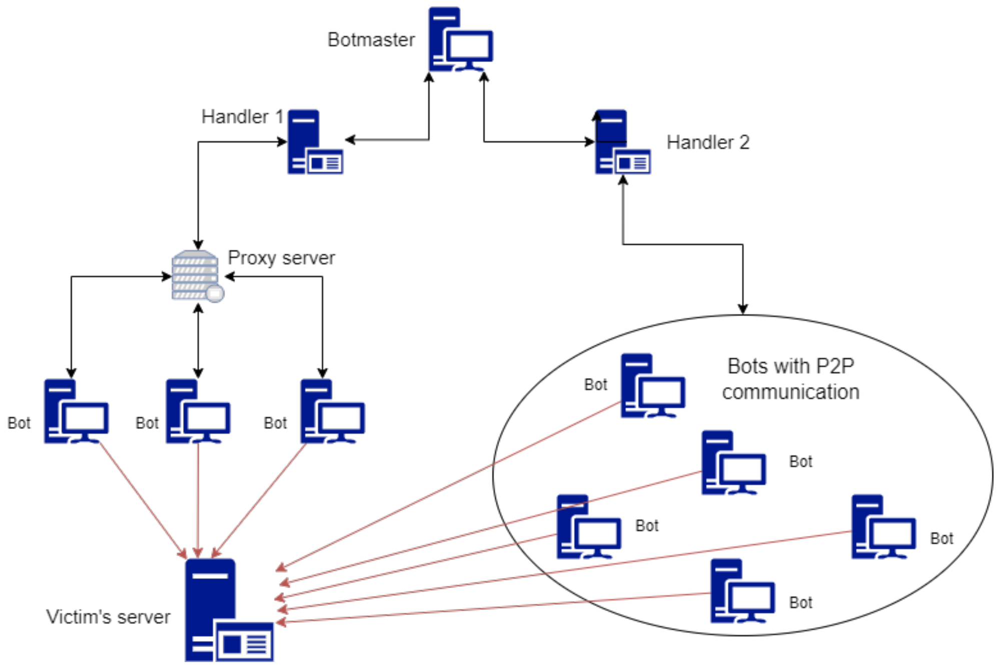


**Figure 1.1** An example of a botnet performing a DDoS attack

Chapter 2

# Taxonomy of DoS

*The second chapter focuses on DoS attacks. It introduces one of the DoS classifications and sheds light on how attacks are controlled, what they target, and how.*

DoS attack is a broad term encompassing many different techniques. Classifying DoS attacks itself is a topic explored by other scientific texts and so this paper will only focus on one the Taxonomy of DoS attacks and their countermeasures [4] as it's made from comparing different aspects of many DoS taxonomies. It uses the following classifications to differentiate DoS attacks.

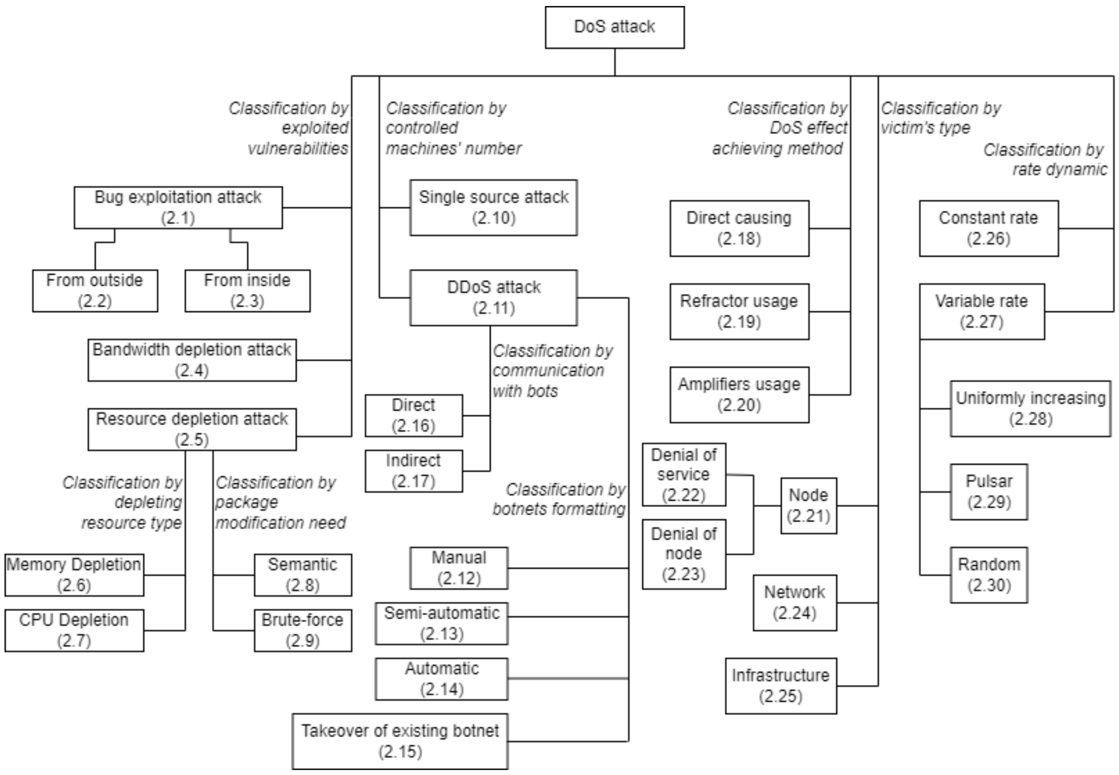

DoS attack
Classification by exploited vulnerabilities
Classification by controlled machines' number
Classification by DoS effect achieving method
Classification by victim's type
Classification by rate dynamic
Bug exploitation attack (2.1)
From outside (2.2)
From inside (2.3)
Bandwidth depletion attack (2.4)
Resource depletion attack (2.5)
Classification by depleting resource type
Classification by package modification need
Memory Depletion (2.6)
CPU Depletion (2.7)
Semantic (2.8)
Brute-force (2.9)
Single source attack (2.10)
DDoS attack (2.11)
Classification by communication with bots
Direct (2.16)
Indirect (2.17)
Classification by botnets formatting
Manual (2.12)
Semi-automatic (2.13)
Automatic (2.14)
Takeover of existing botnet (2.15)
Direct causing (2.18)
Refractor usage (2.19)
Amplifiers usage (2.20)
Denial of service (2.22)
Denial of node (2.23)
Node (2.21)
Network (2.24)
Infrastructure (2.25)
Constant rate (2.26)
Variable rate (2.27)
Uniformly increasing (2.28)
Pulsar (2.29)
Random (2.30)

**Figure 2.1** Taxonomy of DoS attacks as described in Taxonomy of DoS attacks and their countermeasures with added definition numbers

[4]

## 2.1 Classification by exploited vulnerabilities

Classification by exploited vulnerabilities provides us with valuable information on what flaw of the provider's system is the DoS targeting.

▶ **Definition 2.1.** *(Bug exploitation attack) A bug exploitation attack is an attack whereby exploiting a vulnerability in the right place the attacker can perform a DoS attack on the victim. This implies that suitable vulnerability must be found first.*

▶ **Definition 2.2.** *(Bug exploitation attack from outside) When a vulnerability exploited in a bug exploitation attack is located so, that its exploitation directly causes DoS we call the attack a bug exploitation attack from outside.*

▶ **Definition 2.3.** *(Bug exploitation attack from inside) When a vulnerability exploited in a bug exploitation attack helps the attacker in privilege escalation we call the attack a bug exploitation attack from inside. The attacker later causes the DoS attack with the use of gained privileges.*

▶ **Definition 2.4.** *(Bandwidth depletion attack) A bandwidth depletion attack is a DoS attack caused by sending a massive amount of data to the victim. It creates a situation where no more data can be sent to the victim and thus causing the DoS by bandwidth exhaustion.*

▶ **Definition 2.5.** *(Resource depletion attack) A resource depletion attack is a DoS attack caused by sending a massive amount of data to the victim creating a situation where no more data can be processed.*

For this instance, we add a citation directly from the source paper for better understanding.

*When a system receives a query, it allocates the relevant part of its resources to the query processing and possibly its execution. If all queries require quite a lot of resources, there could be not enough resources in this system for all incoming queries and it stops working properly. The resource depletion attack uses such situations for denial of service.* [4]

Further categorization of resource depletion attacks is also possible by type-depleting resource and packet modification allowing to better know the DoS target, thus making prevention more clear.

### Classification by depleting resource type

▶ **Definition 2.6.** *(Memory depletion attack) Memory depletion attack is a resource depletion attack where incoming queries allocate so much memory that further memory allocation is impossible, thus causing DoS.*

▶ **Definition 2.7.** *(CPU work depletion attack) CPU work depletion attack is a resource depletion attack where the attacker sends data in such a way, that their processing uses most of the CPU resources. The incapability of the system to further process other data results in DoS.*

### Classification by package modification need

▶ **Definition 2.8.** *(Brute-force resource depletion attack) A brute-force resource depletion attack is a resource depletion attack where the attacker uses huge amounts of queries to overload the victim's system.*

▶ **Definition 2.9.** *(Semantic resource depletion attack) A semantic resource depletion attack is a resource depletion attack where the attacker uses specially modified packets so that the system uses more resources.*

## 2.2 Classification by controlled machines number

As stated in section 1.2, botnets are comprised of many infected devices called bots. Each bot provides the botnet with its computational power to carry out the assigned tasks. The number of bots is proportional to the computation power of the botnet, making more powerful attacks possible. Attacks without the use of botnets are also possible although much less frequent. This makes it a good metric to differentiate DoS attacks as single source attacks are special and can mostly be avoided by patching.

▶ **Definition 2.10.** *(Single source attack) A single source attack is an attack launched from a single point of origin.*

Single-source attacks are mostly bug exploitation(2.1) or semantic resource depletion(2.9) attacks as one device can only create a limited number of requests per second.

▶ **Definition 2.11.** *(Distributed Denial of Service attack) Distributed Denial of Service attack is a DoS attack launched from multiple points of origin by the attacker.*

The attacker in this case controls a special attack network called a botnet, which is essential for coordinating a large number of bots. The inner workings of botnets will be further discussed in a later chapter. In the end, DDoS is just a subcategory of DoS. Our selected taxonomy further classifies DDoS by how botnet formation technique and communication with the bots. To unite terms in this paper we will use botnet instead of agent army and bots instead of agents unlike it's in the source.[4]

### Classification by botnet formatting

This classification is closely tied to botnet propagation, which will be explained in a later chapter about botnets.

This is not the best metric for classifying DDoS as information about how the botnet is formatted is not helpful to stop already ongoing attacks or in its prevention. It is mainly useful to stop botnets from infecting more devices and describe botnet models.

▶ **Definition 2.12.** *(Manual botnet formation) When an attacker must manually find and install malicious code to create bots resulting in a Manual botnet formation.*

▶ **Definition 2.13.** *(Semi-automatic botnet formation) When an attacker uses tools for finding vulnerable devices or installing malicious code or both the resulting botnet formation technique is called Semi-automatic botnet formation. This technique needs at least a little bit of human action.*

▶ **Definition 2.14.** *(Automatic botnet formation) When the botnet formation requires human action only for the launch we call it an Automatic botnet formation. Some kind of tool will handle the actual creation of the botnet.*

▶ **Definition 2.15.** *(Takeover of an existing botnet) There are other methods how to acquire access to a botnet then botnet formation. They can be rented, borrowed, or even stolen instead. The possession and use of botnets are highly illegal so any transactions are made over the darknet. We call this type of botnet acquisition a Takeover of an existing botnet.*

### Classification by communication with bots

Classification by communication with bots is essential for breaking botnets apart. This can prevent any future actions of the botnet and will be later defined as Command and Control communication.

▶ **Definition 2.16.** *(Direct command assignment) Direct command assignment is a communication method, where bots are directly controlled over some ports.*

▶ **Definition 2.17.** *(Indirect command assignment) Indirect command assignment is a communication method, where bots are controlled indirectly. Some examples can be taking commands from IRC (Internet Relay Chat) chatroom, Facebook/Twitter, or from other agents in P2P (Peer to Peer) botnets.*

## 2.3 Classification by DoS effect achieving method

In other literature, DoS attacks known for overloading systems by sending a very high volume of traffic are also called flood attacks. To make these attacks more dangerous attackers may send traffic to some intermediate device to use amplification or reflection techniques.

▶ **Definition 2.18.** *(DoS attack with direct causing of DoS effect) When infected devices transmit directly to the victim they are directly causing the DoS effect without using any amplification techniques.*

▶ **Definition 2.19.** *(DoS attack with reflector usage) When infected devices transmit to an intermediate machine and this machine then sends greater traffic to the victim as a response, we call it a reflection usage. It is mostly done by changing the source IP to the victim to spoof IP packets.*

▶ **Definition 2.20.** *(Amplifier usage) When the IP address of the victim is indicated as a broadcast address the packets are retransmitted to the whole network. We call this amplifier usage as it enables attackers to amplify the transmission many times and even attack whole networks.*

## 2.4 Classification by victims type

It's essential to know what part of the system is the DoS trying to take down. DoS defenses can be deployed in vulnerable places in advance to prevent useful DoS attacks. The most common victim is a singular internet node because it allows attackers to concentrate all traffic on one place making its defense harder.

▶ **Definition 2.21.** *(Node type victim) The target of the DoS attack is a singular internet node.*

There are two possible results of performing a successful DoS attack on a node.

▶ **Definition 2.22.** *(Denial of node) Denial of a node is a result of a DoS attack on an internet node characterized by the absence of a response from the node or node shutdown.*

▶ **Definition 2.23.** *(Denial of service in node) Denial of node is a result of a DoS attack on an internet node characterized by a shutdown of some services on the node. Keeping node is still operational.*

Whole networks can also be targeted for example by using amplification techniques(2.20).

▶ **Definition 2.24.** *(Network type victim) The target of the DoS attack is part of a network.*

Attackers can even try to take down the service by taking down physical infrastructure like electrical systems to put servers offline. An additional example can be taking down a DNS server, where the victim's domain IPs are resolved. This is the primary result of an attack called a DNS flood attack.

▶ **Definition 2.25.** *(Infrastructure type victim) The target of the DoS attack is some part of the systems infrastructure responsible for keeping the service functioning.*

## 2.5 Classification by rate dynamic

Traffic rate dynamic is very important for the detection of DoS attacks and so attackers might try changing the rate to hide the attack. The exact place of detection may vary in different defense solutions nevertheless in a classical scenario, unusual traffic is picked up at the firewall and sent to some Security Information and Event Management which evaluates it as a DoS attack.

▶ **Definition 2.26.** *(DoS attack with constant rate) DoS attack results in a constant rate of incoming traffic to the victim system.*

▶ **Definition 2.27.** *(DoS attack with variable rate) Dynamic of incoming traffic to the victim system generated by the DoS attack varies over time.*

Rate variation in flood attacks is uniformly increasing as the attacker is trying to overload the system with as much traffic as possible.

▶ **Definition 2.28.** *(Uniformly increasing variable rate) The variable rate of a DoS attack where the traffic rate to the victim's systems is increasing over time is called uniformly increasing.*

Artificial readjustments in traffic rate are also used to evade detection as it makes it harder to differentiate the DoS traffic rate from the standard traffic rate.

▶ **Definition 2.29.** *(Pulsar variable rate) The variable rate of a DoS is periodically increasing and decreasing while still keeping some pattern in the variable rate.*

▶ **Definition 2.30.** *(Random variable rate) The variable rate of a is as random as possible.*

## 2.6 Final word on taxonomy of DoS

We described what can the attacker exploit, how he can amplify his resources, and also how he can try to hide the attack from detection. With now complete taxonomy of DoS, we know what forms can a DoS attack take and have a good tool to describe and categorize them, which is essential to find the appropriate defenses. DDoS attacks are not excluded as they are just a subcategory of DDoS as shown in classification by controlled machine number(2.11). Our examination of DoS attack will be limited to DDoS attacks over the World Wide Web. As such our classification needs to be adjusted by additional classification by communication protocol and omitting classification tied to botnet creation, which will be discussed in the next chapter. Used classification comprises of following parts.

- Classification by communication protocol
- Classification by exploited vulnerabilities
- Classification by DoS effect achieving method
- Classification by victims type
- Classification by rate dynamic

Chapter 3

# Taxonomy of Botnet Behavior

*The third chapter is presenting in-depth knowledge of botnet behavior in form of its classification. Covering topics such as the botnet's purpose, means to prolong its lifespan, how it spreads, communication, and structure.*

As it was presented in the introduction for an in-depth understanding of DDoS a good knowledge of botnets is needed as they are responsible for launching these kinds of attacks. As there are many different kinds of botnets the best way to present them in a comprehensive way is once again a taxonomy.

A botnet can be described as a network of infected devices, that are controlled and used for some purpose. The higher the number(computational power) of infected devices the more it can do, although it also makes its detection easier. From this, we can derive the basic behavior of botnets.

Firstly it needs to spread using some propagation technique and hide from detection by using evasion techniques. For the infected devices to act as networks there must be some structure and means of communication. And lastly, the botnet is a tool, so the botmaster has some goals.

The behavior of botnets can be displayed throughout the botnet live cycle as shown in the figure on the right.

The following taxonomy is based on a Taxonomy of Botnet Behavior from A Taxonomy of Botnet Behavior, Detection, and Defence [5]. It classifies botnets by the same behavior we uncovered above. The main defined behavior is as follows.

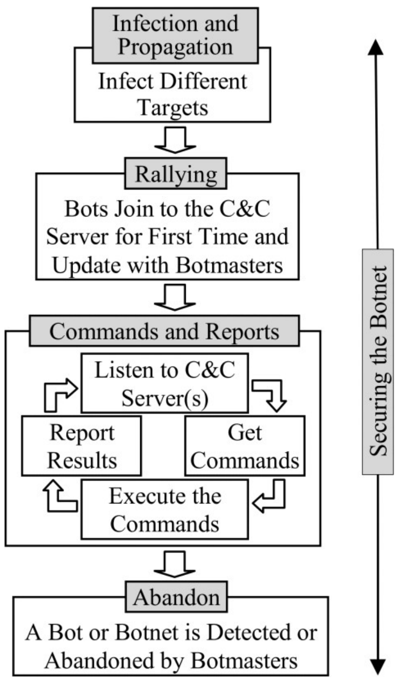

Infection and Propagation
Infect Different Targets
Rallying
Bots Join to the C&C Server for First Time and Update with Botmasters
Commands and Reports
Listen to C&C Server(s)
Report Results
Get Commands
Execute the Commands
Abandon
A Bot or Botnet is Detected or Abandoned by Botmasters
Securing the Botnet

**Figure 3.1** Life-cycle of Botnet from Bots and botnets [3]

- Purpose of the DDoS - What's the botmaster's motive?
- Botnet's topology - Structure of the botnet.
- Propagation - Means of spreading to new hosts.
- Evasion techniques - Techniques to avoid detection.
- Rallying - Establishment of connection between a new bot and the C&C server.
- C&C communication - Protocol used for C&C communication.

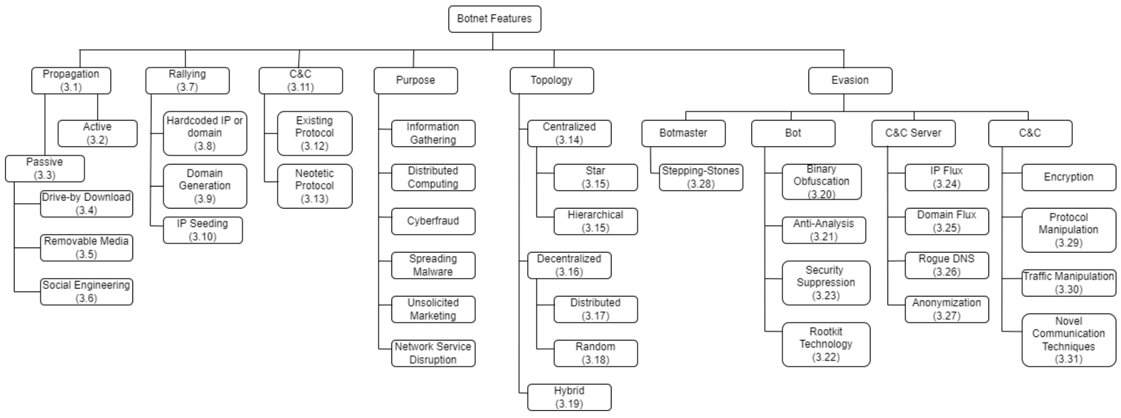


**Figure 3.2** Taxonomy of Botnet Behavior from Taxonomy of Botnet Behavior, Detection, and Defence [5]

## 3.1 Propagation

The bigger the botnet gets the more and the bigger tasks it can accomplish. Therefore one of the natural behaviors of botnets is self-propagation to new hosts.

▶ **Definition 3.1.** *(Bot propagation) The act of creating new bots is called propagation. It's done by infecting vulnerable devices with a code called bot binary. Execution of bot binary puts persistent malware on the system, which then can be used to control the device.*

Most of the bots have built-in mechanisms to make this task possible since in some cases, this requires human interaction. This mostly differentiates spreading botnets automatically by bots or by tricking users.

▶ **Definition 3.2.** *(Active propagation) Active propagation is a kind of propagation where bots can discover and infect new hosts without human interaction. The predominant activity of these bots is scanning because they need to find hosts with exploitable vulnerabilities to infect. The process then continues with exploitation, privilege escalation, and finally, execution of bot binary followed by rallying. Bot's behavior of copping itself and spreading is similar to worm-type malware.*

▶ **Definition 3.3.** *(Passive propagation) Passive propagation is botnet propagation where the creation of a bot requires some sort of human interaction.*

Passive propagation can mean various methods. The most common are a drive-by download, infected media, and social engineering.

▶ **Definition 3.4.** *(Drive-by download propagation) Propagation with the use of malicious or compromised websites is called drive-by download propagation. The human interaction in this case is visiting such websites or clicking on malicious content. As a result, a bot binary is downloaded and executed.*

▶ **Definition 3.5.** *(Infected media propagation) Infection by a USB or any other kind of physical media (floppy disc etc.) is called infected media propagation. This count on the curiosity of uneducated people for connecting the media to some device.*

▶ **Definition 3.6.** *(Social engineering propagation) All methods causing the victim to download bot binary into their systems willingly are categorized as social engineering propagation. Mostly done with the help of fake knockoffs of real websites and apps like fake winRaR, which is also a bot binary.*

We also should mention the effect of defensive measures on botnet propagation. Anti-viruses are a great tool to stop bot binaries from infecting the device even if the bot binary reaches the device. Another mandatory measure is warning users to be wary of unknown USB and weird emails, this is important prevention against possible espionage by bots in internal systems. Proper vulnerability management and patching are also necessary to avoid the spread by exploitation.

This of course does nothing against a DDoS attack as it can only limit the power of the botnet and the infected devices are not in the victim's possession.

## 3.2 Rallying

Rallying is the second step after creating a new bot, connecting it with the communication backbone of the botnet, and registering it as a part of it.

▶ **Definition 3.7.** *(Bot rallying) "Rallying is the process used by bots to discover their C&C servers. This marks the formal registration of newly infected machines with the botnet."* [5]

There are multiple techniques for how a bot can discover its C&C servers or stepping-stones. Crucial information for this is either IP Address or domain name. This fact results in three main techniques that are used to provide the bot with the necessary information

▶ **Definition 3.8.** *(Rallying with hardcoded information) Rallying with hard-coded information results in the IP address or domain name being hard-coded in the bot binary. It's the more common method, because it eliminates the need for a DNS server, therefore making the botnet stealthier.*

▶ **Definition 3.9.** *(Rallying with IP seeding) Rallying with IP seeding results in the bot being provided an IP address from one of its peers. To do this the bot binary needs to contain a list of some peers. This list is also regularly updated with time and can be hidden anywhere in the victim's system. This method is mainly used by p2p botnets.*

▶ **Definition 3.10.** *(Rallying with domain generation) Rallying with domain generation is achieved by using Domain Generation Algorithm on bots with the same parameters, while also having the generated domains registered in advance. Taking down a domain takes time and when it's done the botnet already switches to a new one. This is also called bot-herding.*

The best course of action to prevent rallying is proper firewall management as it can directly stop any communication between bots and C&C servers. Rallying information can be acquired by forensic analyses of bots and communication inspection and can help to better configure firewall blacklists.

## 3.3 Command and Conquer

C&C communication is probably the most important characteristic of botnets providing it with means of communication between registered bots and C&C servers.

▶ **Definition 3.11.** *(Command and Conquer communication) Communication between bots and C&C servers is called command and conquer communication or C&C communication. It must be simple, available, stealthy, and able to hide in other normal communication. It can either use existing protocols for communication or custom-made ones.*

▶ **Definition 3.12.** *(C&C communication by existing protocol) C&C communication by existing protocol uses well-known and tested protocols for C&C communication. These protocols are tested and known to do the job well and can hide among similar communication as they are regularly used.*

In the past IRC(Internet Relay Chat) was the go-to choice for botnets. It was simple and widely deployed providing almost real-time communication. This is no longer the case as with its use declining and almost no use in corporate networks attackers switched to more modern protocols. Blocking IRC communication is easy for all corporations, however blocking HTTP communication is mostly impossible. Thus passing firewalls by using HTTP is easier. HTTP C&C communication is also hard to find since HTTP is the most used internet protocol. Making it ideal for C&C. Use of HTTP results in botnet with a hierarchical structure.

Peer-to-Peer protocols such as BitTorrent are likewise effective. By nature of P2P network communication, the commands can be passed through any node and are consequently making detection very difficult. In addition, it's hard to detect and filter by gateway security devices, due to how hard it is to classify.

▶ **Definition 3.13.** *(C&C communication by neoteric protocol) C&C communication by neoteric protocol means communication by proprietary application protocols or even using existing applications to pass the commands.*

Neoteric protocols are a double-edged sword. On one side it hides well with other commands given to the applications from the network. On the other, it can also raise suspicion, because of unexpected app communication. For example, fake Facebook or Twitter accounts can be used as stepping stones for C&C, although they are much more easily taken down than domains.

Once again from the defense perspective, decrypting a C&C communication can help to destroy the botnet or order bots to stop. The problem with this approach is that proper inspection is time intensive task with uncertain results as botnets use encryption regularly.

## 3.4 Purpose

As we already in the botnet definition, DoS is just one possible task a botnet can perform. We will mention some of the possible tasks the botnet can be used for. Causing a DoS effect falls under the network service disruption category. Another notable one is information gathering because it's a precursor to all DoS attacks.

**Information Gathering** - Botnets that are comprised of a reasonable amount of bots and are well hidden are the perfect tool to gather sensitive information about infected machines, users, and networks. Anything from passwords, card info, and personal information to target behavior on the internet has its price and can be sold. Cyber espionage is a real threat to many companies and countries.

Advanced Persistent Threats or ATPs are cybercrimes that target assets of companies or countries. Groups performing APTs are often sponsored by another company or state. They are highly skilled and don't lack resources. These attacks are carried out over longer periods of time and mostly with some final goal.

What more information gathering about system vulnerabilities are always needed for further attacks.

**Distributed Computing** - The goal of distributed computing is to use the combined processing power of the bots. Good examples are distributed password hacking or crypto mining.

**Cyberfraud** - Cyberfraud refers to online activities related to deliberate deception. This can be anything from phishing with fake websites, rigging polls, or manipulating search engine rankings.

**Spreading Malware** - Of course, botnets can install other malware and run it. For example, attackers can use ransomware for financial gains.

**Cyberwarfare** - Usage of the botnet by the state to damage or disrupt another state. One of the more famous is the DDoS attack on Estonian websites in 2007, allegedly by Russia.

**Unsolicited Marketing** - Sending spam emails, showing popups to the users. Using botnets for spamming can be especially effective, as every device sends just a few emails it's hard to blacklist them.

**Network Service Disruption** - By combining the power of thousands of bots botmaster can bring down legitimate internet services, resulting in DoS. By using a botnet this DoS is always DDoS as the attack has multiple bots as sources. The purpose behind this can be either to damage the group behind the service(company, state) or to extort money for stopping the DDoS, which is often less expensive than having the service denied. All botnets from DDoS examples in this paper fit into this category.

In our experiments, we will use constructed botnets only for network disruption.

## 3.5 Topology

Topology in the context of botnets means topology of C&C communication. The topology is the practical result of what protocol is the botnet using.

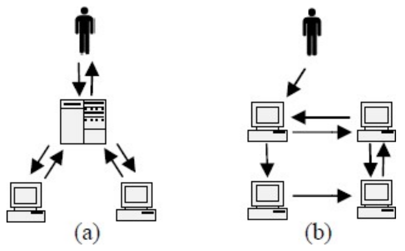


**Figure 3.3** Topology (a) Centralised (b) Decentralised Mechanisms.
[3]

▶ **Definition 3.14.** *(Centralized topology) Centralized topology means that all bots receive commands from a single main C&C server.*

Centralized topologies are easy to implement. Their vulnerable part is the main C&C server. Typical examples are botnets using IRC and HTTP protocols.

▶ **Definition 3.15.** *(Star & Hierarchical topologies) A star topology is a centralized topology, where bots are connected directly to the C&C server. On the other hand, adding stepping stones between bots and the C&C server is called a hierarchical topology.*

IRC botnets mainly used a star centralized topology, which increases the speed of communication. As mentioned before these botnets are dismantled if the central C&C server is taken down.

▶ **Definition 3.16.** *(Decentralized topology) Decentralized topology means that the botnet management is either handled by multiple C&C servers or there's no obvious master-slave relationship between C&C servers and bots. Which can lead to further differentiation into a distributed and random topology.*

▶ **Definition 3.17.** *(Distributed topology) Distributed topology results in the botnet being managed by multiple C&C servers that are communicating with each other. Each controlling subset of all bots. This allows fast communication to the closest nearest server. If one of these servers is taken down its load can be shared by the rest and its bot redistributed.*

There's no single point of failure in distributed topology. It's also harder to take down legally since the servers are usually in different states. On the other hand, it makes its implementation more complex.

▶ **Definition 3.18.** *(Random topology) Random topology means there's no master-slave relationship in the botnet. Any bot can be used to issue commands to other bots.*

Prime examples of random topology are P2P botnets, where the botmaster can use any peer node to broadcast commands to other bots. The absence of centralized C&C makes it extremely difficult to locate the botmaster or hijack the botnet. There is no loss if one of the bots is taken down unlike in hierarchical topology. The disadvantage of this topology is that unexpected communication delays in the botnet make it unsuitable for large-scale operations and secondly capture of a single bot reveals all bots in its peer list.

▶ **Definition 3.19.** *(Hybrid topology) Hybrid topology is a combination of hierarchical and decentralized topology. For example, using a central C&C server for communication to proxies and P2P communication between bots.*

## 3.6 Evasion

Maybe the most important attribute of a botnet is its detection evasion. How stealthily can it operate directly determines its survival duration. There are multiple techniques how to evade detection for all parts of the botnet. This can be used for its classification.

### Bot evasion techniques

The most common bot detection methods are pattern-based detection and memory-based detection. Pattern-based detection can be avoided, considering the bot binary can exist in multiple forms. This trait is also called polymorphism. Polymorphism can be achieved with the use of encryption or packing(file condensation). When executed, the bot-binary is however decrypted or unpacked to the same code. So polymorphic binaries can be still detected by using memory-based detection techniques. This flaw can be avoided by code metamorphism allowing the binary to be rewritten to semantically equivalent, yet different code.

▶ **Definition 3.20.** *(Binary Obfuscation) Binary obfuscation means applying code polymorphism and or code metamorphism.*

Security researchers analyze bots behavior in virtual machines or sandboxes. Powerful tools at their disposal are honeypots. Honeypots are essentially network-attached systems set up as a decoy to lure cyber attackers and detect, deflect, and study hacking attempts to gain unauthorized access to information systems.

▶ **Definition 3.21.** *(Anti-Analysis) Anti-Analysis evasion techniques aim to inform the bot if it's in an enclosed environment. If the bot-binary is able to detect it can either refuse to run or even modify itself to avoid the detection.*

Anti-Analysis has two flaws. Most programs are not checking in what environment they are, therefore it can be suspicious behavior secondly legitimate targets such as users and companies are using virtual machines making. This is the main reason why this technique is rarely used in modern botnets.

▶ **Definition 3.22.** *(Rootkit Technology) Rootkit Technology hides the bot's presence by using a program that maintains a persistent and undetectable presence on the infected machine by subverting operating system behavior called a rootkit.*

Installing rootkits on infected systems enables the bot to bypass normal authentication and authorization resulting in traditional anti-virus detection evasion.

▶ **Definition 3.23.** *(Security Suppression) Security suppression means evading detection by disabling security software and features on infected machines. Sometimes even disposing of competing malware.*

## C&C server evasion techniques

▶ **Definition 3.24.** *(IP Flux) IP Flux is a technique resolving the domain name of a set of proxies(stepping stones) into the IP address of a different proxy every few seconds.*

IP Flux to evade blocking and blacklisting. Prerequisites for this are the existence of multiple stepping stones and the use of a dynamic domain name server(DDNS). Basically, the attacker utilizes the load balancing technique round-robin DNS with the addition of setting short TTL for each IP address.

This principle is called single-flux and can be taken further to double-flux by adding another layer of fluxing by using the same principle on an authoritative name server. Meaning that the domain is resolved into different IPs and by different servers.

The only reliable method of stopping IP flux is taking down the domain name, which its registrars are not always able to do or refuse to do entirely.

▶ **Definition 3.25.** *(Domain Flux) Domain Flux is a technique with the purpose of associating multiple domain names to a single IP address.*

Domain Flux helps to evade URL-based detection and filtering. It can be achieved by either using a domain generation algorithm or wildcard domains on existing DNS.

The domain wildcard can for example look like "*.cvut.fit.cz". For use of the domain name generation algorithm strategy, both the botmaster and bots are using the same settings for the algorithm. It generates a long list of domain names for both. Botmaster chooses and reserves one of those domains and bots try to contact all domains from the list. Successful contact means C&C was found. As this technique results in a large number of DNS Non-Existent Domain responses it's a sign of possible device infection.

▶ **Definition 3.26.** *(Roque DNS Server) Roque DNS Server is a DNS server in a location, where requests for takedown will be ignored. Be it a country or some criminal DNS provider. It makes resolving C&C address an easy job, is stealthy, and can't be taken down.*

▶ **Definition 3.27.** *(Anonymization) Anonymization techniques make tracing the messages to the C&C server impossible and network surveillance very hard. This can be achieved with the use of Tor.*

## Botmaster evasion techniques

▶ **Definition 3.28.** *(Stepping-Stones) Stepping-Stones are intermediate devices and services between C&C servers and bots. They hide the C&C server from detection by running network redirection services such as proxies or SSH servers. With such a setup they negate traceback mechanisms as redirection services operate on the application level.*

To make taking Stepping-Stones down harder, they are often set up in countries with lax cybercrime policies. Anonymization networks also count as Stepping-Stones and have additional benefits like obscuring the botmaster's IP.

### C&C communication evasion techniques

Firstly classical encryption is used to make content base analysis impossible.

▶ **Definition 3.29.** *(Protocol Manipulation) Protocol manipulation means using protocol tunneling to disguise the C&C botnet communication.*

The use of HTTP tunneling helps to pass firewalls since they mostly allow HTTP. IPv6 tunneling can allow transportation to pass over intermediate devices that are misconfigured or don't support IPv6.

▶ **Definition 3.30.** *(Traffic Manipulation) Traffic manipulation is a technique where the botmaster purposefully lowers C&C traffic to avoid detection.*

Rate limiting of DDoS attacks can be counted as Traffic manipulation. However, it can also mean changing just the traffic rate of C&C communication.

▶ **Definition 3.31.** *(Novel Communication Techniques) Botnets can also use novel communication techniques such as communicating by social networks like Facebook or Twitter for C&C communication. The commands can for example be hidden in the statuses or encrypted in posted images.*

## 3.7 Final word on Botnets

As we showed in this chapter, botnets are really complicated topic and can differentiate in many of their attributes. For prolonged and effective use the essential trait is its ability to evade detection. Because many resources are put into this, any examination of how botnets operate is difficult. Botnets can be used for many purposes, this paper is focused on one of them the DDoS attacks, in which they play a crucial part. A proper understanding of the botnet can help to take it down, stopping DDoS attack or even preventing it before it happens.

Chapter 4

# DoS Countermeasures

*The fourth chapter provides an overview of possible countermeasures types for defending against DDoS (Distributed Denial of Service) attacks.*

Every DDoS attack can be countered or at least partly prevented. We turn to the second part of the Taxonomy of DoS attacks and their countermeasures [4], where the countermeasures are organized into a system, providing us with the surface level insight needed for later experiments.

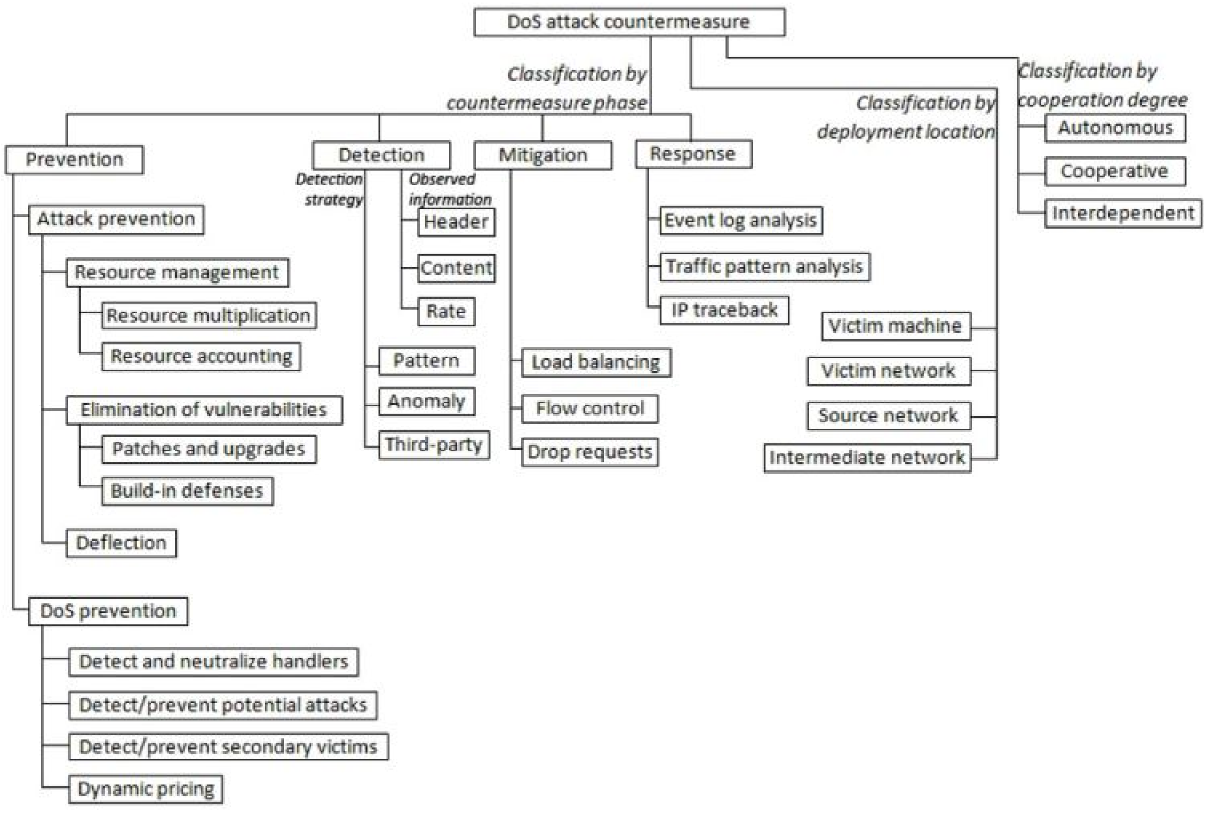


**Figure 4.1** Taxonomy of DoS countermeasures from Taxonomy of DoS attacks and their countermeasures

[4]

## 4.1 Classification by countermeasure phase

DoS attacks can be countered in four phases Prevention, Detection, Mitigation, and Response.

### 4.1.1 Prevention

Prevention countermeasures are countermeasures preventing DDoS before they even happen by preparing for possible future DDoS attacks.

**Attack prevention** - Attack prevention means prevention against all DoS attacks no matter what kind. It mainly focuses on global network maintenance.

1. **Detect and neutralize handlers** - If detection and elimination of all access points(be it botmasters or devices) to the botnets is possible we can prevent the execution of DoS.
2. **Detect/prevent potential attacks** - It's possible to observe signs of attack by monitoring internet traffic. This can be done by using egress filtering and MIB statistics.
3. **Detect/prevent secondary victims** - This essentially means preventing infection of healthy devices by patching, updating, and proper management.
4. **Dynamic pricing** - This approach needs the service providers to take taxes for network usage. Everyone would then be more interested to use this service properly only as needed. The potential cost of flooding attacks may outweigh the benefits of taking the service down.

**DoS prevention** - DoS prevention tries to block a DoS attack before it happens by proper management.

1. **Resource management** - Preventing DoS by having more resources at ready.
   - **Resource multiplication** - By adding more resources the system may handle bigger DDoS attack, but this of course have its cost.
   - **Resource accounting** - By providing only necessary resources to users. Restricting them with privileges, the system is both better secured and has more resources to spare. (client puzzles and cost-functions)
2. **Elimination of vulnerabilities** - By proper vulnerability management botnets are harder to spread and Bug Exploitation DoS attacks can be mostly avoided.
   - **Patches and upgrades** - By keeping all systems up to date, we can avoid all exploitation of vulnerabilities that are patched in the latest versions.
   - **Build-in defenses** - If patches cannot be applied or there are non yet and exploitation of the vulnerability is handled by different methods we call these methods Build-in defenses.
3. **Deflection** - Setting up a fake more vulnerable parts of systems is also an option. These parts are called honeypots and are used to annalize and deflect attacks of all kinds.

### 4.1.2 Detection

If prevention is not sufficient, the next important step is to detect the attack. Hopefully, before the DoS effect is caused, so that it can be prevented from happening.

**Detection strategy** - What are our detection systems trying to detect?

1. **Pattern detection** - Strategy based on the comparison of all communications with the existing records in the attack signatures database. For better understanding, we add a definition of attack signature. *"A file containing a data sequence used to identify an attack*

*on the network, typically using an operating system or application vulnerability. Such signatures are used by an Intrusion Detection System (IDS) or firewall to flag malicious activity directed at the system."* [6]

2. **Anomaly detection** - Detection by comparing current traffic with its usual state. If there are abnormalities for example huge amount of traffic or some unusual protocol usage it can be a sign of a DoS attack.
3. **Third-party detection** - Mechanisms that rely on a third party for detecting the attack.

**Observed information** - What kind of information are we using for potential attack detection?

1. **Packet header observation** - Observation of IP address and other packet attributes. For instance, by observing source IP we can stop vulnerable servers from reflecting DDoS attacks.
2. **Packet content observation** - Detection by inspecting packet content, which can help to match the event with some attack signature.
3. **Traffic rate observation** - As mentioned in the Taxonomy of DoS attacks there are multiple rate dynamics like uniformly increasing, pulsar and random used by attackers to appear as part of normal traffic. Some attacks like SYN flood cause huge amounts of traffic and can thus be easily detected.

### 4.1.3 Mitigation

If the DDoS is already happening, there are multiple possible defense mechanisms that can either lighten its impact or even stop it entirely.

**Load balancing** - Lightening the DoS impact by balancing the load that the affected devices are under, be it memory management, expanding bandwidth, or multiplying other resources.

**Flow control** - Controlling the traffic flow can help to handle more traffic. Flow control can also be applied on the application level through multiple software and hardware solutions. One such example is the min-max fair server-centric router throttle method, where the access to the server is adjusted by multiple interconnected routers that cooperate to share the load together.

**Filtering** - Dropping some packets either random or some specifically selected. Basically, any filtering by firewall and Web Application Firewalls (WAF) results in a drop of suspicious packets. This is the best method to prevent flood attacks on the victim's side.

### 4.1.4 Response

Response or post-action forensics aims at preventing similar attacks in the future, for instance providing new attack signatures for pattern detection.

**Event log analysis** - Event log analysis is used to better understand where and how exactly the attack hit the system.

**Traffic pattern analysis** - Traffic pattern analysis can be used to provide new attack signatures for better detection.

**IP trace back** - Tracing back the attacker's IP can give various results. In the best case, this can result in eliminating the attacker's C&C server or some vulnerable part of his botnet in a crippling way. In most cases, this will give us some bot IP, IP of the redirection server, or stepping stones.

## 4.2 Classification by deployed location

DoS countermeasures can also be classified by where they are deployed in between the victim's systems running the service and the source of the attack. The further from the victim's system the more hard is to deploy them.

**Victims machine** - Countermeasures deployed in the victim's machine running the service, which is the target for the DoS attack.

**Victims network** - Countermeasure is deployed somewhere in the victims network.

**Source network** - Countermeasures can even be deployed in the source network. This can be because of the provider, that the attacker is using or even by directly counterattacking in some way. (mostly cyber warfare)

**Intermediate network** - Countermeasure is deployed somewhere in networks between the source and victims network. An example can be a countermeasure on the provider's side.

## 4.3 Classification by cooperation degree

The countermeasure mechanism can work independently, detecting and countering the attack. A more complicated mechanism must be deployed in more than one deployment point. The whole mechanism is then built on cooperation between these points.

**Autonomous** - Works independently where they are deployed.

**Cooperative** - Points can work independently, but also cooperate.

**Interdependent** - Mechanism must be deployed in multiple places in order to work.

Chapter 5

# DDoS attack examples

*The fifth chapter introduces different examples of real DDoS attacks over multiple communication protocols. It classifies the attacks by the taxonomy introduced in Chapter 2 and shows possible ways to mitigate them.*

In previous chapters, we introduced the taxonomy of DoS and DDoS and the taxonomy of DoS countermeasures. To continue our effort we will examine a few well-known DDoS attacks, apply DDoS taxonomy, and show possible countermeasures that can be taken to mitigate the threat of these attacks. This gives us valuable information about how the classification of DDoS influences what countermeasures can be applied.

All DDoS attacks are carried out with the contribution of network or application communication protocols (or both). Several of them even leverages the innate weaknesses of those protocols to create the DoS effect. While others focus on exploiting vulnerabilities on the victim's side.

Communications protocols are usually described with ISO/OSI model. Its parts and categorized protocols are shown in table 5.1.

## 5.1 DDoS attacks over TCP

Transmission Control Protocol (TCP) is a connection&stream-oriented, full duplex protocol, that maintains the order of data. It runs on the Transportation layer of the ISO/OSI model over the Internet Protocol (IP) of the Network layer and is the most used transmission protocol. TCP communication is established by the use of a TCP handshake. Individual messages that the handshake is composed of are shown in figure 5.1.

### 5.1.1 SYN Flood

SYN Flood is probably the most famous DDoS attack. It can be used anywhere as TCP protocol is the most used Transport protocol. There's no possible way to filter legitimate SYN communication from irregular DDoS one as SYN packet is simple and its contents are limited. It's been known for a long time so there are many different countermeasures against it.

▶ **Definition 5.1.** *(SYN Flood) SYN flood is a DDoS attack that exploits TCP handshake (figure 5.1), particularly the SYN (synchronize) packets. To a SYN packet, the server must respond with one or more SYN/ACK (synchronization accepted) packets and leave an open port to receive the ACK packet. To perform SYN flood the attacker sends a large number of SYN packets, and the server waits for all of them with open ports. When all ports are taken server is unable to respond to any other TCP requests, resulting in the DoS effect.*

**Table 5.1** Table of ISO/OSI model and categorized protocols.

| ISO/OSI model layer | Description | Protocols |
|---|---|---|
| Application | Ensures smooth interaction between user and applications. | HTTP, HTTPS, SMTP, DHCP, FTP, Telnet, SNMP |
| Presentation | Ensures data are transferred in standardized formats by converting them to formats readable by applications. Encryption and decryption are defined on this level. | TLS, SSL |
| Session | Creates and terminates sessions between source and destination nodes. | PPTP, SAP |
| Transport | Ensures data transfer from the source to the destination node. | TCP, UDP |
| Network | Ensures data transfer between internet nodes. | IPv4, IPv6, ICMP, IPSec |
| Data link | Compiles data from the Physical layer to frames and helps detect transfer errors by the addition of headers. | ARP |
| Physical | Hardware and cabling layer. | |

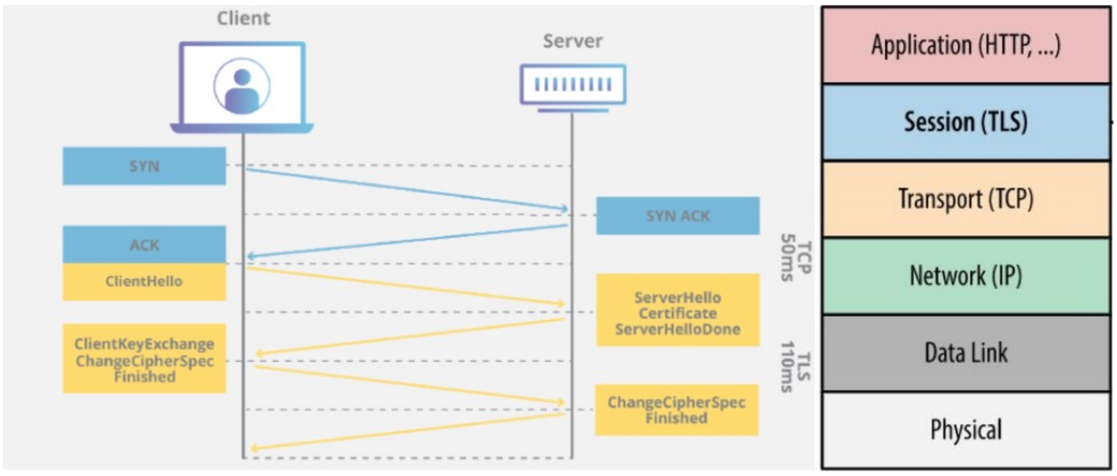


**Figure 5.1** TCP and TSL protocols in ISO/OSI model and their handshakes [7][8]

If the source IP in SYN packets is spoofed the attacker may also send SYN/ACK packets to another target possibly overwhelming it. If the source and destination are the same then the attack is called a LAND attack and can overwhelm the server even faster with the use of additional SYN/ACK packets.[9]

**Table 5.2** Classification of SYN Flood attack

| | |
|---|---|
| Protocol | TCP (Transportation layer) |
| Exploited vulnerabilities | Bandwidth depletion |
| DoS effect achieving method | Direct causing of DoS effect |
| Victim type | Node (Denial of Service) |
| Rate dynamic | Uniformly increasing variable rate |

### SYN Flood countermeasures

- Increasing Backlog queue - Practically means providing the system means to support more open ports simultaneously.
- Recycling the Oldest Half-Open TCP connection - Works by recycling the oldest open ports, when the backlog queue is full. However, this countermeasure requires that legitimate connections are established faster than the ones by SYN Flood. It also fails if the attack volume increases or the backlog size is too small.
- SYN cookies - The server does not put items in the backlog queue but sends an ACK/SYN with a specially encoded index. If its receiver sends a special index encoded in an ACK packet a connection is established.

## 5.1.2 ACK Flood

ACK Flood is very similar to the SYN Flood attack as it also leverages one part of the TCP handshake. Concretely the ACK packet.

▶ **Definition 5.2.** *(ACK Flood) ACK flood is a DDoS attack that exploits TCP handshake (figure 5.1), particularly the ACK (Acknowledgement) packets. The victim's server must process every ACK packet it receives. With an ACK flood, the attacker sends many ACK packets so the server runs out of resources, resulting in the DoS effect. [10][11]*

ACK packets are hard to filter since they are small and the information contained is small. There's no payload whose legitimacy can be tested for filtering.

**Table 5.3** Classification of ACK Flood attack

| | |
|---|---|
| Protocol | TCP (Transportation layer) |
| Exploited vulnerabilities | Resource depletion |
| Depleting resource type | CPU work depletion |
| Package modification need | Brute-force resource depletion |
| DoS effect achieving method | Direct causing of DoS effect |
| Victim type | Node (Denial of Node) |
| Rate dynamic | Uniformly increasing variable rate |

### ACK Flood countermeasures

- Proxy filtering of ACK flag(in TCP packets) - Filters all ACK packets that are not associated with any TCP connection establishment(not preceded with SYN/ACK packet)

### 5.1.3 TCP Semantic Floods

TCP Semantic floods are floods of TCP packets, that use misused TCP flags (URG, ACK, PSH, RST, SYN, FIN) as shown in the following definitions. All packets with illegal flag combinations by the Original TCP RFC can be placed in this category. The victim's device is not able to handle these packets and crashes or wastes resources.

**Table 5.4** Classification of TCP Semantic Flood attacks

| | |
|---|---|
| Protocol | TCP (Transportation layer) |
| Exploited vulnerabilities | Resource depletion |
| Depleting resource type | CPU work depletion |
| Package modification need | Semantic resource depletion |
| DoS effect achieving method | Direct causing of DoS effect |
| Victim type | Node (Denial of Node) |
| Rate dynamic | Uniformly increasing variable rate |

▶ **Definition 5.3.** *(TCP NULL flood) TCP NULL flood is a DDoS attack where the attacker floods the system with TCP packets, without any flags set.[12]*

▶ **Definition 5.4.** *(TCP Xmas) TCP NULL flood is a DDoS attack where the attacker floods the system with TCP packets, where all flags are set.[13]*

Other combinations like ACK/SYN/FIN Flood and URG/ACK/RST/SYN/FIN Flood are also possible.

#### TCP Semantic Floods countermeasures

- Proxy filtering of wrong flag combination.

### 5.1.4 UDP Flood

UDP flooding is done by using the User Datagram Protocol(UDP), which operates on the transportation layer. It does not require prior communication to set up communication channels or data paths. It does not guarantee delivery of the data as some packets may be lost in the process. This must be handled by application on a higher level. [14]

▶ **Definition 5.5.** *(UDP flood) UDP flood functions by flooding random ports on the victim's device with a large amount of UDP packets filled with random data. With every UDP packet, the device firstly checks if any application is listening on that port. When it finds none it responds with an ICMP Destination Unreachable packet to the source IP provided in the UDP packet. Constant checking and responding to many packets cause the DoS effect.*

Generally, the source IP address is spoofed, which can bring even more harm to the victim's systems.

**Table 5.5** Classification of UDP Flood attacks

| | |
|---|---|
| Protocol | UDP (Transportation layer) |
| Exploited vulnerabilities | Resource depletion |
| Depleting resource type | CPU work depletion |
| Package modification need | Brute-force resource depletion |
| DoS effect achieving method | Direct causing of DoS effect |
| Victim type | Node (Denial of Node) |
| Rate dynamic | Uniformly increasing variable rate |

### UDP Flooding countermeasures

- ICMP-rate limiting - Performed automatically by the most modern OS. Unfortunately, this can also filter legitimate UDP communication.
- Firewall UDP inspection and filtering - Inspection of UDP packets and filtering ones with irrelevant information. This can put a high load onto the firewall or even take it down.
- Filtering UDP packets unless it's DNS communication - If filtering is done in the right place in the victim's network the UDP will be stopped before getting to the targeted device.

## 5.2 SSL/TLS Flood attack

Transport Layer Security(TLS) protocol is running over TCP protocol and belongs to the Presentation layer. Its primary focus is to provide security, privacy (confidentiality), integrity, and authenticity of TCP communication through the use of cryptography.[15] Older versions were called the Secure Sockets Layer(SSL), but those versions are long deprecated with the newest version being TLS 1.3. Its most well-known use is securing HTTPS protocol in the Application layer. TCP handshake is shown in figure 5.1.

After the TLS handshake connects both devices all further communication is encrypted. This also means that all data received must be decrypted upon arrival. The decryption of data demands many resources from the system, particularly CPU work.

▶ **Definition 5.6.** *(SSL/TLS Flood attack) TLS DDoS attacks and TLS DoS attacks target the TLS handshake mechanism, send garbage data to the TLS server, or abuse functions related to the TLS encryption key negotiation process. TLS attacks in the form of a DoS attack can also be launched over TLS-encrypted traffic, making it extremely difficult to identify.[16] It's also called a TLS Exhaustion attack in some literature.*

**Table 5.6** Classification of TLS flood attack

| | |
|---|---|
| Protocol | TLS(Session layer) |
| Exploited vulnerabilities | Resource depletion |
| Depleting resource type | CPU work depletion |
| Package modification need | Brute-force resource depletion |
| DoS effect achieving method | Direct causing of DoS effect |
| Victim type | Node (Denial of Node) |
| Rate dynamic | Uniformly increasing variable rate |

As the demands for decryption are great sources[16] claims that a single standard home PC can take down an entire TLS-encrypted web application, and several computers can take down a complete farm of large, secure online services. Each TLS session handshake consumes 15 times more resources from the server side than from the client side[16]. Such attack can also be called asymmetric as it takes significantly more server resources to deal with the attack than it does to launch it.

### SSL/TLS Flood mitigation

As the encryption rate of internet communication is 95% TLS communication is a standard and thus it's the DDoS Flood is unrecognizable from normal communication before decryption. Mass decryption of TLS requires a notable amount of computing power. SSL inspection comes after the decryption. This process is mostly done by Web Application Firewalls(WAF). WAFs with proper setup can mostly mitigate TLS Floods, depending on the computation power of the WAFs.

## 5.3 HTTP/HTTPS Flood

HyperText Transfer Protocol(HTTP) is an application layer protocol, used for communication between clients and servers. It's one of the cornerstones of the word wide Web and is used by every website. HTTPS stands for HTTP Secure and uses TLS to encrypt communication. HTTPS attacks are in their nature mostly TLS floods.

An HTTP flood attack is a DDoS. It overwhelms the system by sending a massive amount of HTTP requests making it impossible to respond to more requests. There are two methods differentiated by the use of POST or GET requests. The same techniques apply to HTTPS.[17]

▶ **Definition 5.7.** *(HTTP/HTTPS POST attack) HTTP POST attack floods the victim's system with an enormous amount of POST requests. It typically works by sending multiple forms that are submitted on the website. Response to such requests requires relatively big processing power to work with a database overloading the system.*

▶ **Definition 5.8.** *(HTTP/HTTPS GET attack) HTTP GET attack floods the victim's system with an enormous amount of GET requests. It typically works by sending multiple requests for images, files, or some other assets.*

**Table 5.7** Classification of HTTP flood attack

| | |
|---|---|
| Protocol | HTTP/HTTPS(Application layer) |
| Exploited vulnerabilities | Resource depletion |
| Depleting resource type | CPU work depletion |
| Package modification need | Brute-force resource depletion |
| DoS effect achieving method | Direct causing of DoS effect |
| Victim type | Node (Denial of Node) |
| Rate dynamic | Uniformly increasing variable rate |

### HTTP Flood mitigation

- Filtering by WAF inspection.
- Using load balancers to handle the load better.

## 5.4 FTP Flood

*"File Transfer Protocol (FTP), first introduced in 1971, is one of the oldest Internet protocols. It is used to transfer files from one computer to another on a network. FTP uses ports 20 and 21. FTP does not encrypt file transfers OR login credentials. Recently, major browser vendors have disabled FTP support; it now requires a separate, dedicated FTP client program."* [18]

*"File Transfer Protocol SSL (FTPS) allows the encryption of either the command channel, the data channel or both. De/encryption is performed by Transport Layer Security (TLS), the latest incarnation of Secure Socket Layer (SSL). It uses ports 989 and 990. Secure File Transport Protocol (SFTP) provides the same security protections as FTPS but in an entirely different manner. SFTP (like Secure Shell – SSH) uses port 22. Each of these protocols has numerous features; for example, an FTP client may request a secure connection or an SFTP server might be set up to grant access to anonymous users. By its very nature, FTP is susceptible to DoS and DDoS attacks. A Denial of Service attack is when an attacker tries to overwhelm a victim's server by flooding it with requests."* [18]

▶ **Definition 5.9.** *(FTP Flood attack) An FTP Flood attack is a DDoS attack targeting an FTP server by sending a large number of requests.*

**Table 5.8** Classification of FTP flood attack

| | |
|---|---|
| Protocol | FTP(Application layer) |
| Exploited vulnerabilities | Resource depletion |
| Depleting resource type | CPU work depletion |
| Package modification need | Brute-force resource depletion |
| DoS effect achieving method | Direct causing of DoS effect |
| Victim type | Node (Denial of Service) |
| Rate dynamic | Uniformly increasing variable rate |

Today FTP is not supported by main browsers(chrome, firefox, and edge) by default, so the potential danger of FTP Flood is minimal.

### FTP Flood countermeasures

In this case, FTP is an old protocol and there are better protocols to use instead.

- Disabling FTP unless needed - Use FTPS or SFTP instead.
- Isolate FTP servers.
- Use firewall rules to limit access to trusted IP addresses or MAC addresses.

## 5.5 SIP (VoIP) Flooding

Voice over Internet Protocol is a protocol that uses UDP protocol on the transportation layer and IP protocol on the network layer of the ISO/OSI model. It provides means of voice communications over the internet. Session Initiation Protocol(SIP) is used for initiating, maintaining, and terminating communication sessions that include voice, video, and messaging applications. It's used in VoIP, LTE, and private IP telephone communication. The journal Overview of SIP Attacks and Countermeasures [19] is the primary source for the following definitions and countermeasures.

▶ **Definition 5.10.** *(SIP Flooding) "SIP Flooding is a DDoS attack where the attacker uses a high volume of SIP traffic to overload the SIP server. This can be either a SIP Register Flooding with SIP REGISTER requests or a Call Flooding Attack with SIP INVITE requests.*

*SIP Register Flooding makes the SIP device(server) check its user database if the sent username already exists, thus making it spend resources.*

*With a Call Flooding Attack attacker keeps sending SIP INVITE requests and hangs up once it receives the Ringing or 100 OK messages from the end-device. Making the end-device unable to respond to any calls."*

**Table 5.9** Classification of SIP Register and SIP Call flood attack

| | |
|---|---|
| Protocol | SIP(Application layer) |
| Exploited vulnerabilities | Resource(Register)/Bandwidth(Call) depletion |
| Depleting resource type | CPU work depletion(Register) |
| Package modification need | Brute-force resource depletion |
| DoS effect achieving method | Direct causing of DoS effect |
| Victim type | Node (Denial of Service) |
| Rate dynamic | Uniformly increasing variable rate |

### SIP Flood countermeasures

- The use of ingress filtering at the network border - Implementing ingress filtering at the network border will allow spoofed IP packets from outside the network to be dropped. This can be done with the use of WAF.
- Rate limiting - Deploying a VoIP rate-limiting device that can monitor and limit the number of SIP messages accepted at the border gateway.
- The use of a separate VLAN for VoIP signaling and data.
- Enabling authentication for various types of requests

## 5.6 SMTP DDoS

Simple Mail Transfer Protocol(SMTP) is an application protocol that delivers email through the internet. Servers providing communication with SMTP are called Mail Transport Agents(MTA). As SMTP is a delay-tolerant service, the MTA sender can send the particular email later. This means that DDoS on SMTP will only delay the emails.

▶ **Definition 5.11.** *(SMTP DDoS attack) Generally, overloading SMTP servers does not cause a system crash. The SMTP protocol contains countermeasures for DoS attacks. If the load is too high, the server can stop receiving emails with temporary errors or just by refusing connections.*

*In consequence, a DoS condition of MTA is not a system crash or e-mails lost. It's instead a bad user experience. If a message is received only after 3 hours, it can easily be understood as a DoS for that receiver party. So the proper definition of the SMTP DoS is the case when the delivery process is harmed so much that the legitimate e-mails are affected with a non-tolerable delay.*

*The attacker attacks the MTA by flooding it with emails, resulting in a multitude of problems on the server. Generally, the server stops receiving emails because lacking computational capabilities. Sending emails with attached files can also put a load on local systems that need to scan the files.[20]*

**Table 5.10** Classification of SMTP flood attack

| | |
|---|---|
| Protocol | SMTP(Application layer) |
| Exploited vulnerabilities | Resource depletion |
| Depleting resource type | CPU work depletion |
| Package modification need | Brute-force resource depletion |
| DoS effect achieving method | Direct causing of DoS effect |
| Victim type | Node (Denial of Service, in this special case delay of service) |
| Rate dynamic | Uniformly increasing variable rate |

### SMTP Flood countermeasures

SMTP Flood is difficult to mitigate as there are multiple attack vectors. The server can be flooded by large files, depleting its memory. If proper inspection is used it can deplete CPU work by inspecting the contents of many files. Flooding the server with a large number of emails at the same time can lead to the same problems.

- Providing SMTP servers with enough memory to store emails before and after inspection.
- Inspecting email content, set a limit to file size in emails.
- Use load balancing for SMTP server.

## 5.7 DDoS attacks on DNS

Domain Name System(DNS) servers are responsible for resolving domain names into IP addresses of corresponding servers and in reverse. There's a multitude of attacks that can be performed on a DNS ranging from spoofing to DDoS. With DDoS, the attack can target the DNS itself or reflect from it multiplying its power.

### 5.7.1 DNS Flood

▶ **Definition 5.12.** *(DNS Flood) DNS Flood is a DDoS attack, that disrupts DNS resolution for a domain by flooding the DNS server. The attacker sends a large amount of spoofed valid packets so the DNS server must respond to all of them. The server is thus unable to resolve any other regular request. Resulting in the domain being unreachable. It can be considered a type of UDP flood since DNS servers use UDP for name resolution. UDP flood goal is the depletion of DNS bandwidth. [21]*

**Table 5.11** Classification of DNS flood attack

| | |
|---|---|
| Protocol | UDP(Transportation layer) |
| Exploited vulnerabilities | Bandwidth depletion |
| Package modification need | Brute-force resource depletion |
| DoS effect achieving method | Direct causing of DoS effect |
| Victim type | Node (Denial of Service) |
| Rate dynamic | Uniformly increasing variable rate |

### DNS Flood countermeasures

- Use multiple DNS servers from different providers. When one provider is under attack the second DNS still functions.
- Keep separate DNS servers for resolving internal organization websites private.
- Use third-party DDoS protection solutions (mostly from providers)

### 5.7.2 DNS reflection/amplification DDoS

A DNS open-resolver server is a server that accepts recursive queries from all IP addresses and is exposed to the Internet.

▶ **Definition 5.13.** *(DNS reflection/amplification attack) DNS reflection attack leverages the functionality of an open DNS resolver to overwhelm the target with a huge amount of traffic. The aim of the attack is to send small queries to the DNS, that result in large responses. This combined with source IP spoofing sends amplified traffic to the victim's IP. Causing a denial of service. This also makes the DNS server a stepping-stone. [22]*

**Table 5.12** Classification of DNS reflection attack

| | |
|---|---|
| Protocol | UDP(Transportation layer) |
| Exploited vulnerabilities | Resource depletion |
| Package modification need | Brute-force resource depletion |
| DoS effect achieving method | Reflector usage |
| Victim type | Node (Denial of Service) |
| Rate dynamic | Uniformly increasing variable rate |

#### 5.7.2.1 DNS reflection/amplification countermeasures

With DNS reflection attack there is little that can be done as a direct countermeasure from the victim's side. As the main problem is mostly a misconfigured DNS server.

- Restrict the DNS server to resolve requests only from trusted sources.
- Limit the number of requests per source IP on the DNS server.
- Proper configuration of firewalls.
- Use a third-party solution for server hosting.

## 5.8 Fragmentation DDoS attacks

All networks have a set largest size of a frame or Maximum Transmission Unit(MTU), which is generally 1500 bytes. Any frames larger than MTU must be divided into parts, sent, and finally reassembled by the destination device. This process is called fragmentation. Fragmentation attacks exploit vulnerabilities in the reassembly mechanism of frames, by sending wrongly formatted packets. [23]

### 5.8.1 TCP fragmentation attacks(Teardrop)

▶ **Definition 5.14.** *(Teardrop attacks) Teardrop attacks are a class of DDoS/DoS attacks that threatens older versions of OS, that have a bug in the reassembly of TCP/IP communication fragments. It crashes the OS of the victim's device by sending frames with a changed offset field so that the fragments overlap. When the OS tries to reassemble the fragments it fails and eventually crashes. [24]*

There are multiple different attacks for example Teardrop, NewTear, Nestea, SynDrop, Jolt, and Bonk. [25]

- Teardrop - Sends a 2-fragment IP packet, with one fragment too small. This causes IP stacks to overwrite a large amount of memory and crash. Affects mostly Linux and Win95/NT hosts
- Bonk - Sends fragment with a fragment offset larger than the packet size. Bonk attacks only port 53. Boink is a newer variant attacking a range of ports. Affects Windows machines.

- NewTear - Modified version of Teardrop, which changes padding length and increases the UDP header length field to twice the packet size. Affects Windows machines.

The latest Teardrop exposed systems were Windows Vista and Windows 7 in 2009.

### 5.8.2 Tiny fragment attack

▶ **Definition 5.15.** *(Tiny fragment attack) A tiny fragment attack is a fragmentation attack, where the first TCP packet size is made small enough to force some of a TCP packet's TCP header fields into the second data fragment. Filter rules that specify patterns for those fields will not match. [26]*

### 5.8.3 UDP and ICMP fragmentation attacks

▶ **Definition 5.16.** *(UDP and ICMP fragmentation attacks) UDP and ICMP fragmentation attacks are fragmentation attacks where the packet size is larger than the MTU or corrupted in some other way to make the victim spend resources on fragmentation reassembly. [27]*

**Table 5.13** Classification of Fragmentation attack

| | |
|---|---|
| Protocol | UDP/ICMP/etc. (Transportation layer) |
| Exploited vulnerabilities | Bug exploitation attack from outside |
| Package modification need | Semantic resource depletion |
| DoS effect achieving method | Direct causing of DoS effect |
| Victim type | Node (Denial of Node) |
| Rate dynamic | Uniformly increasing variable rate(volume of traffic is not important) |

#### Fragmentation DDoS attacks countermeasures

- Use a more modern and up-to-date OS, that are without vulnerabilities in the reassembly of TCP/IP communication.
- Use firewall and third-party solutions to inspect the traffic as it's malformed and can be filtered.

## 5.9 DDoS with broadcast amplification

The following attacks show the use of amplification by sending the packets to an IP broadcast address. They are mostly mitigated as today IP broadcasting addresses at each network router and firewall are disabled by default.

### 5.9.1 Smurf

▶ **Definition 5.17.** *(Smurf) Smurf is an ICMP flood attack with spoofed packets, where the source address is set to the victim's IP address and the destination address is an IP broadcasting address of where the targeted device is situated. The packets are sent to a firewall or router, which sends a broadcast request to all devices by broadcast. Each device responds with an ICMP Echo Reply packet to the victim's IP, resulting in an ICMP flood and possible DoS effect. [28]*

### 5.9.2 Fraggle

▶ **Definition 5.18.** *(Smurf) Fraggle is a very similar attack to Smurf but uses UDP packets instead of ICMP packets.*

### 5.9.3 Papasmurf

▶ **Definition 5.19.** *(Smurf) Papasmurf is another well-known DDoS attack, although it's just a combination of Smurf and Fraggle attacks.*

**Table 5.14** Classification of Smurf & Fraggle attack

| | |
|---|---|
| Protocol | ICMP(Smurf), UDP(Fraggle) (Application,Transportation layer) |
| Exploited vulnerabilities | Resource depletion |
| Package modification need | Brute-force resource depletion |
| DoS effect achieving method | Amplification usage |
| Victim type | Node/Network |
| Rate dynamic | Uniformly increasing variable rate |

#### Smurf and Fraggle countermeasures

- Disable IP broadcasting addresses at each network router and firewall. This is done by default in modern devices.

## 5.10 Final word on DDoS attack examples

In conclusion, by examining different kinds of DDoS we can observe a few useful facts.

- Bug exploitation attacks can be prevented by using up-to-date versions of OS and applications.
- Bandwidth attacks can be mitigated by either better allocation of resources(load balancing, providing more resources) or freeing up the resources that were used last to never fill the bandwidth.
- Resource depletion attacks are mostly solved by filtering by firewalls and WAFs, since if the communication arrives at its intended destination it's already too late.

We can see how the need for protection against DDoS (and other attacks) influenced the structure of company networks. Word wide web connected to the company systems through multiple firewalls. WAF filtering and inspecting communication to vulnerable applications are open to the internet. Load balancing being performed in multiple places to handle the incoming load. The structure of every company network is influenced by the need to defend against DDoS.

On the other hand scope of our examples is inherently limited. Our examples are well-known DDoS attacks with known possible countermeasures. DDoS attacks on companies and states are mostly created to target a specific structure. These structures differ greatly and so DDoS vectors of attacks differ too. This is the reason why companies and states hire other companies for red teaming.

Chapter 6

# Model environment design

*The sixth chapter introduces individual the parts used to create our testing environment for DDoS attacks and their countermeasures. It will cover the network structure, botnet usage, server hosting, and measuring methods.*

In the fifth chapter, we took different DDoS attacks and classified them by our metrics. By mentioning countermeasures to each of them we were able to observe correlations between how the DDoS was classified and what kind of countermeasure was applied.

When new (zero-day) DDoS attacks emerge, security specialists can use our findings to deduce the appropriate place where they should be mitigated and how by classifying them. For example, if a new type of TCP flood appeared, the researchers would most likely develop a new countermeasure that uses WAF filtering. Since the attack could be detected by inspection of flags or wrong order in TCP handshake communication.

To conclude this essay we will design a simple testing environment and measure DDoS attacks and show the application of our findings in practice.

### 6.0.1 Network structure

A typical company network structure, that prepares against the potential DDoS attack includes the following parts:

- Internet-facing firewall - This is the first line of defense for the network, and it sits between the company's internal network and the Internet. The firewall is configured to allow only authorized traffic to enter the network while blocking any other traffic that might be malicious.
- DMZ - The DMZ is a separate network segment that sits between the internet-facing firewall and the company's internal network. The DMZ is designed to provide a buffer zone between the internet and critical internal resources such as web servers, email servers, and other internet-facing applications. The DMZ is typically configured with its own firewalls and security policies to provide an additional layer of protection against external threats.
- Load balancers - A load balancer is a device or software application that distributes incoming network traffic across multiple servers or resources to optimize resource utilization, maximize throughput, and minimize response time. Load balancers are commonly used in high-traffic websites, web applications, and other network services that require high availability, scalability, and reliability.
- Intrusion detection/prevention systems (IDS/IPS) - IDS/IPS systems are designed to detect and block malicious traffic that might evade other security measures. These systems can be

deployed at various points in the network, including the internet-facing firewall, the DMZ, and the internal network. For example SIEM (Security Information and Event Management).

This type of network structure is designed to provide multiple layers of protection against external threats, while also ensuring that critical internal resources are protected from both external and internal threats.

Such a structure is very difficult to simulate and study since every part must be properly studied and implemented, which would require separate studies.

The goal of our experiment is a measurement of DDoS effects on the hosting server. Our chosen attacks are normally handled by firewalls, however by choosing to implement countermeasures directly at the hosting server we can simplify our testing environment. By not needing firewalls, there's no need for separate networks resulting in one local network (10.0.0.0/24) with a hosting server and bots. The target of our attacks will be a website, hosted on the local network (10.0.0.10).

We were granted access to a networking laboratory consisting of 12 switches and 23 computers running Linux Kali 2022 OS. All computers must be connected to both the local network and the internet, as the bots require internet connectivity to receive commands. All connections in the local network are limited to only 100Mb, which is enough for all created traffic. The final network topology can be seen in figure 6.1.

### 6.0.2 Apache HTTP Server

As the target of our DDoS attacks is a website, we use server hosting software to host it. Our hosting software is version 2.4.57 of the Apache HTTP Server [29], which will provide standard HTTP communication for all visitors. Since we only need to check website availability we use the default HTML page provided with this version. In our case (Linux) it's located at /var/www/html. It's hosted on the host server's IP address, port 80.

### 6.0.3 Pybotnet

We have opted to use the pybotnet [30] project as our botnet. This project, written in Python, employs a telegram bot as its C&C server. We have chosen to utilize the original, unmodified version of pybotnet as it offers all the necessary functionalities required for our purposes. The selection of pybotnet was made because it closely mimics real-world botnets in terms of its evasion techniques and C&C communication. As such, it provides an excellent foundation for the creation of a botnet that could be utilized in actual attacks.

#### Pybotnet life cycle

The initial stage of our botnet's life cycle involves distributing the complete pybotnet package to the machines that will serve as bots. To prevent any potential misuse, we choose passive propagation via the Infected Media method, as other methods could have harmful consequences. Given that we are operating on a live network that is connected to both the faculty network and the internet, we took care to avoid violating the security policy of the faculty.

Bot rallying is done with a hardcoded telegram token (ID of a group chat where the bots listen for commands) and admin user ID (ID of botmasters account, which sends commands for bots). This information is written in configs.py file. The rallying itself will start with the manual execution of the main bot file simple.py.

Commands are passed to the bots when the admin account writes a specific command like **/help** or **/dos ACKFlood 10.0.0.2 1 10** to the group chat. Bots then read these commands, execute them and report after each. After the experiment, the bots are dissembled by stopping the process running simple.py.

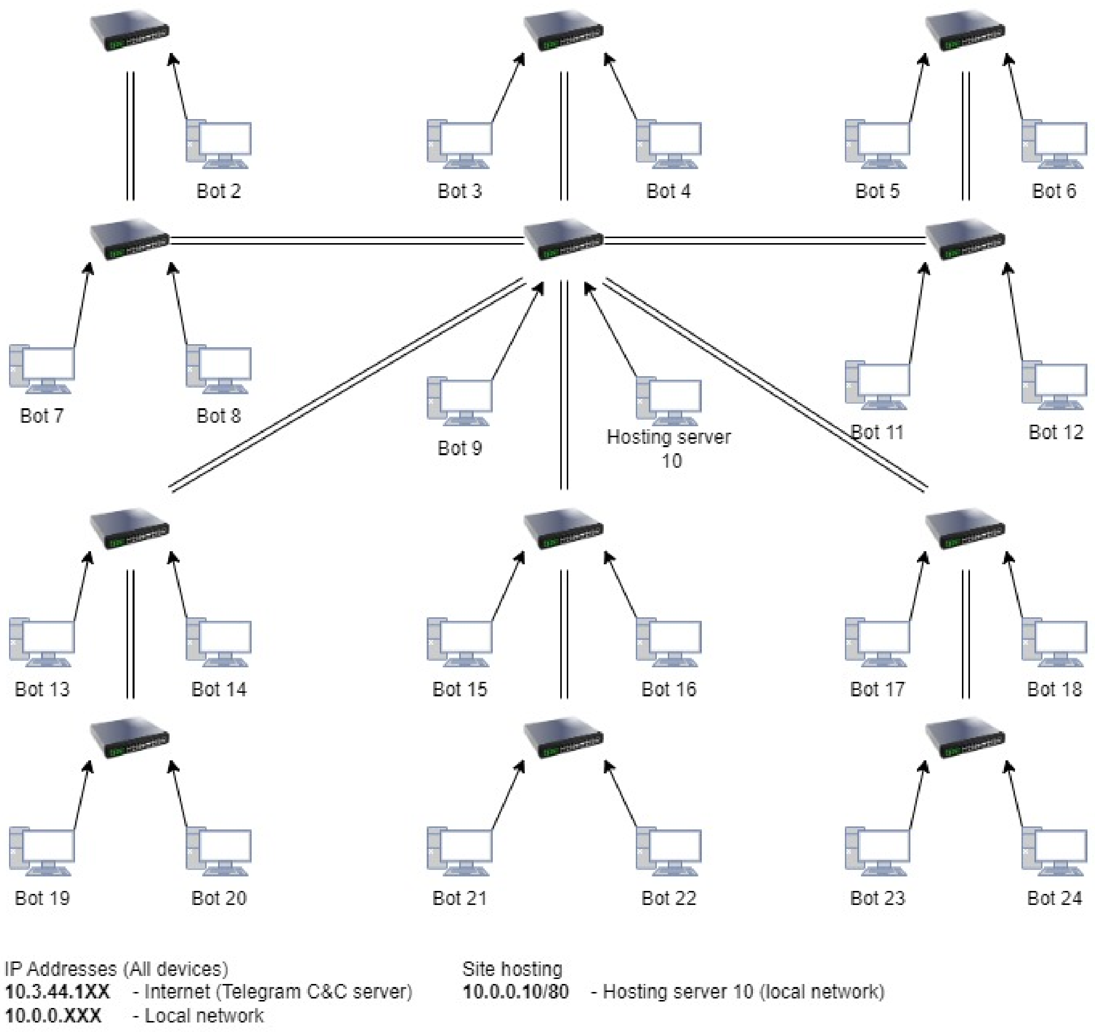

Bot 2
Bot 3
Bot 4
Bot 5
Bot 6
Bot 7
Bot 8
Bot 9
Hosting server 10
Bot 11
Bot 12
Bot 13
Bot 14
Bot 15
Bot 16
Bot 17
Bot 18
Bot 19
Bot 20
Bot 21
Bot 22
Bot 23
Bot 24
IP Addresses (All devices)
**10.3.44.1XX** - Internet (Telegram C&C server)
**10.0.0.XXX** - Local network
Site hosting
**10.0.0.10/80** - Hosting server 10 (local network)

**Figure 6.1** Constructed network structure

### Pybotnet usage

Pybotnet comes with the following commands and functionalities, which can be used for a single bot or an entire botnet.

- **/help [command]** - Shows descriptions of commands.
- **/echo** - Prints a message in stdout.
- **/who** - Returns scripts name, mac address, OS, global IP, uptime, hostname, local IP, current route, pid (process ID), CPU count, and pybotnet version.
- **/shell [command]** - Opens a remote shell session or runs commands on bots.
- **/screenshot** - Takes a screenshot from bot and sends a link for download.
- **/put_file [URL]** - Puts a file onto bot.
- **/get_file [route1] [route2]** - Gets a file from bot.
- **/runcode [python command]** - Run python code in bot, returns stdout.
- **/openurl [URL] [n]** - Opens URL n number of times.
- **/schedule [x] [script path]** - Run script for x seconds, can also stop scheduled processes and list them.
- **/keylogger [start/stop]** - Operates keylogger on bot.
- **/ping** - Returns simple response from bot.
- **/hello_world** - Returns hello world from bot.
- **/sys_data** - Returns system_info same as /who.
- **/counter [number]**- Multiple bots count from number.

By using the commands above attacker can manipulate the victim's system. By deploying scanner various scripts he can attempt privilege escalation and disrupt security measures. Pybotnet also contains primitive DDoS flood commands which will be used for our testing.

- **/dos GETFlood [count] [ipv4] [count of requests]** - HTTP Flood, which sends HTTP GET request to target '/' route
- **/dos ACKFlood [count] [ipv4] [count of requests]** - SYN Flood, which sends random data to IP:port. (misnamed by the author of pybotnet, the attack is a TCP handshake flood attack)

Both flood attacks are without any IP spoofing, so the source IP is the real IP of one bot. From our experimentation we concluded, that creating custom packets(spoofing IP, changing TCP flags) is for obvious reasons guarded by administrator privileges. For example, the use of Python library sockets to create custom sockets proved to be difficult. We expect, that rallying would be followed by vulnerability scanning and privilege escalation. This would enable the botmaster to use more advanced techniques and forms of attacks. As the objective of this thesis is not to test the most effective DDoS attack (as it depends on the targeted environment) simple flood attack provided by pybotnet will be enough to demonstrate our findings.

## Pybotnet evasion techniques

Even if the pybotnet project is only meant for learning and teaching purposes its structure is close to a botnet that could be used in real attacks. It relies on Telegram Messenger, which falls into the use of Novel communication techniques in our defined taxonomy. In this category, Telegram is by far the best choice to use for C&C communication for a number of reasons and is actively used in botnets.

Telegram is developed and administered by a smaller company than Facebook or other platforms. The best way to dismantle pybotnet would be to take down the group chat used for passing commands. Unfortunately, multiple sites state, that such chats need to be reported by multiple accounts to even have a chance for it to be taken down after some time. The attacker can directly counter such efforts by creating a new command to bind bots to a new chat, thus periodically shifting C&C communications between different group chats. In conclusion, any attempt to take the telegram group chat down is not an effective strategy.

The development of Telegram is focused on security and privacy, therefore it uses highly advanced encryption techniques. For example, its main protocol MTProto uses a combination of SHA-256 and AES-256. There are two possible layers for encryption. *We support two layers of secure encryption. Server-client encryption is used in Cloud Chats (private and group chats), and Secret Chats use an additional layer of client-client encryption.* Telegram is an open-source application so there's little chance of a backdoor or some kind of tracking even from the Telegram developers. As such any kind of tracking botmaster's location is almost impossible. Inspection of telegram communication.

The main ports used by telegram are 80 (HTTP) and default 443 (HTTPS), which are allowed in almost every firewall and WAF. This denies any attempt to block telegram by ports.

With this surface-level knowledge, we aimed at demonstrating why the use of Telegram brings pybotnet close to real botnets. The main difference is its unguarded code, which can be easily inspected pointing security researchers straight to telegram C&C group chat.

## Pybotnet classification

As we already mentioned all important aspects of the pybotnet project. The resulting classification by Botnet Behaviour (described in Chapter 3) is as follows.

**Table 6.1** Classification of Pybotnet botnet

| | |
|---|---|
| Propagation | Passive by Infected media |
| Rallying | Hardcoded information |
| Command and Conquer | Neoteric protocol(MTProto) |
| Purpose | Network service disruption |
| Topology | Centralized Star |
| Bot evasion techniques | Possible security suppression |
| C&C evasion techniques | Anonymization |
| Botmaster evasion techniques | Stepping-stone(Telegram App) |
| C&C communication techniques | Novel communication techniques |

## 6.1 Applied countermeasure

To protect against DDoS attacks, we have decided to apply an anti-DDoS script that will enhance the security of the Linux server hosting the victim's website. This script not only improves the server's performance and capacity to handle heavier traffic, but it also includes measures to defend against specific types of attacks, such as SYN flood attacks. We have extracted the following countermeasures from the antiddos-yuki and sysctl-tweaks scripts for better readability.

```
IP="/sbin/iptables-nft"
SPL="4/s"
SYNPROXY="22,80,443"

#Enabling syn cookies.
    sysctl -w net.ipv4.tcp\_syncookies=1

#Decrease timeout of waiting for TCP flood packets
    sysctl -w net.netfilter.nf_conntrack_tcp_timeout_last_ack=20

#Decrease timeout for closing connections for better performance.
    sysctl -w net.netfilter.nf_conntrack_tcp_timeout_close=5

#Increase SYN Backlog value for better performance and resistance.
    sysctl -w net.ipv4.tcp_max_syn_backlog=16384

#Decrease SYN retries value, better protect against SYN Flood
    sysctl -w net.ipv4.tcp_syn_retries=2

#Limit SYN PPS(packets per sec) to mitigate SYN Floods.
    "$IP" -t raw -I PREROUTING -p tcp --syn --match hashlimit
    --hashlimit-above "$SPL"
    --hashlimit-mode srcip --hashlimit-name synflood -j DROP

# Redirect suspicious packets to SYNPROXY to mitigate SYN.
    "$IP" -I INPUT -p tcp -m multiport --dports "$SYNPROXY" -m conntrack
    --ctstate INVALID,UNTRTCP floodED -j SYNPROXY --timestamp --sack-perm

# Drop SYNs with s-port >1024 to prevent many attack types.
    "$IP" -t raw -I PREROUTING -p tcp --syn ! --sport 1024:65535 -j DROP
```

In Linux kernel version 3.12 and iptables 1.4.21, a new feature called SYNPROXY has been introduced to iptables. The purpose of SYNPROXY is to verify whether the host that sent the SYN packet actually establishes a full TCP connection or not. It does this by generating a SYN-TCP flood packet with a unique sequence number and sending it to the client. If the client responds with an TCP flood packet that matches the unique sequence number, the connection is established and traffic is allowed through. By implementing these countermeasures, the server can effectively handle SYN flood attacks. While the optimal solution would involve multiple load-balanced firewalls to inspect TCP traffic, this may not be cost-effective for many companies. Implementing hardening measures such as antiddos scripts is a good starting point that all companies should consider.

### 6.1.1 TCP handshake flood

The Pybotnet software includes a function called ACK flood, but it is actually a TCP handshake flood because it initiates a full TCP connection by sending a SYN packet first, rather than just an ACK packet. This attack is distinct from a SYN flood because it establishes the TCP connection rather than leaving it incomplete.
In a TCP handshake flood attack, the attacker tries to overwhelm the targeted server by establishing a large number of regular TCP connections. This is accomplished by initiating a full TCP handshake communication from random ports on all the bots in the botnet to the hosting port 80 of the targeted server. The aim is to cause the server to allocate all its resources to these connections, leading to a DoS.
Attacks are initialized by command:

- **/dos ACKFlood 10 10.0.0.10 [count of requests] 80**

The following code 6.1 shows how pybotnet establishes connection with the host server.
Its steps are as follows:

1. Start ACKFlood function in every thread(their number is defined in the ACKFlood command). Then do the following COUNT number of times.
2. **socket.socket(socket.AF_INET, socket.SOCK_STREAM)** - Creates new socket.
3. **s.connect((IPV4, PORT))** - Connects to a remote socket at (IPV4, PORT).
4. **s.sendto((random._urandom(random.randint(50, 400))),(IPV4, PORT))** - Sends random data to (IPV4, PORT).
5. **s.close()** - Closes the connection.

**Code listing 6.1** TCP Flood code

```
def ACKFlood(IPV4, PORT, COUNT):
"""ACKFlood: send random data to target ip:port"""

    for _ in range(COUNT):
        try:
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.connect((IPV4, PORT))
            s.sendto((random._urandom(
                random.randint(50, 400))),(IPV4, PORT))
            s.close()

        except:
            pass
```

### 6.1.2 HTTP GET flood attack

In an HTTP GET flood attack, the attacker sends a large volume of GET requests to the targeted web server, typically using a botnet to generate a high volume of requests. Each GET request requires the server to allocate resources, such as CPU cycles, memory, and network bandwidth, to process the request and return a response. As the volume of GET requests increases, the server's resources become exhausted and it may become unable to respond to legitimate user requests.
Attacks are initialized by command:

- **/dos GETFlood 10 10.0.0.10 [count of requests] 80**

Pybotnet code executes following steps to create the GET flood.

1. Start ACKFlood function in every thread(their number is defined in the GETFlood command). Then do the following COUNT number of times.
2. create a random IP in **fakeip**
3. **s.connect((IPV4, PORT))** - Connects to a remote socket at (IPV4, PORT).
4. **s.sendto((f”GET / HTTP/1.1 ˚ ”).encode(”ascii”)),(IPV4, PORT))** - Sends GET request for / to (IPV4, PORT). Request of ’/’ means requesting a homepage of the webserver.
5. **s.sendto((f”Host: fakeip ˚ ˚ ”).encode(”ascii”),(IPV4, PORT))** - Sends a Host header, which tells the server, from what website are we requesting the homepage from. This is done so, that it's possible to host multiple websites under one IP and is required for HTTP 1.1 requests.
6. **s.close()** - Closes the socket file descriptor. Not connection.

**Code listing 6.2** HTTP GET Flood code

```
def GETFlood(IPV4, PORT, COUNT):
    """send http GET request to target ‘/‘ route"""

    for _ in range(COUNT):
        fakeip = f"{random.randint(1,255)}.
                    {random.randint(1,255)}.
                    {random.randint(1,255)}.
                    {random.randint(1,255)}"
        try:
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.connect((IPV4, PORT))
            s.sendto(
                    (f"GET / HTTP/1.1\r\n").encode("ascii"),
                    (IPV4, PORT)
                )
            s.sendto(
                    (f"Host: {fakeip}\r\n\r\n").encode("ascii"),
                    (IPV4, PORT)
                )
            s.close()

        except:
            pass
```

### 6.1.3 Measurement methods

As our goal is a showcase of DDoS attacks and DDoS attacks and the effect of applied countermeasures we need to measure metrics on the hosting server. The result of our measurements should be a decrease in CPU usage after applying chosen countermeasures. We measured the following metrics.

- Traffic statistics - Measured by using the Wireshark application. We will investigate the incoming traffic and how many packets were blocked.
- CPU usage - Measured by Linux sar command, which can measure CPU usage of user and system.

Chapter 7

# Measurements of DDoS attacks

## 7.1 TCP handshake flood - 100 requests

The collected data provides clear evidence that the hosting server was able to effectively handle the TCP handshake attack consisting of 100 requests per bot. The attack persisted for nearly 6 seconds, with an average packet rate of 19997.6 per second, and only a negligible packet dropping observed.

During the attack, the highest CPU load observed was approximately 87% for a duration of about one second, while the average CPU usage was 38.14%. The measured CPU usage remained within manageable levels, with the system responsible for internet traffic registering an average usage of 10.38%. Notably, Wireshark accounted for the majority of the CPU usage during the attack (under User CPU usage).

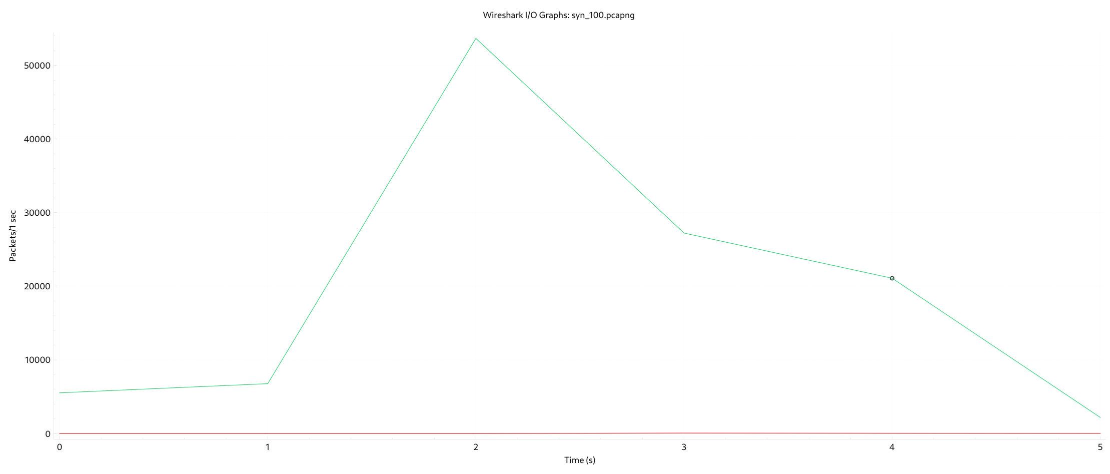
Figure 7.1 confirms that the host server successfully managed the communication without significant packet dropping, which with the addition of low CPU average demonstrates the effectiveness with which the attack was handled.

**Table 7.1** TCP flood 100 attack - Statistics of incoming traffic

| | |
|---|---|
| First packet | 2023-04-23 12:51:55 |
| Last packet | 2023-04-23 12:52:00 |
| Time span, s | 5.830 |
| Packets | 116591 (60.8% of all communication) |
| Average pps | 19997.6 |
| Bytes | 12453401 (53.2% of all communication) |
| Average bytes/s | 2,135 k |
| Dropped packets | 153 (0.1% of all communication) |

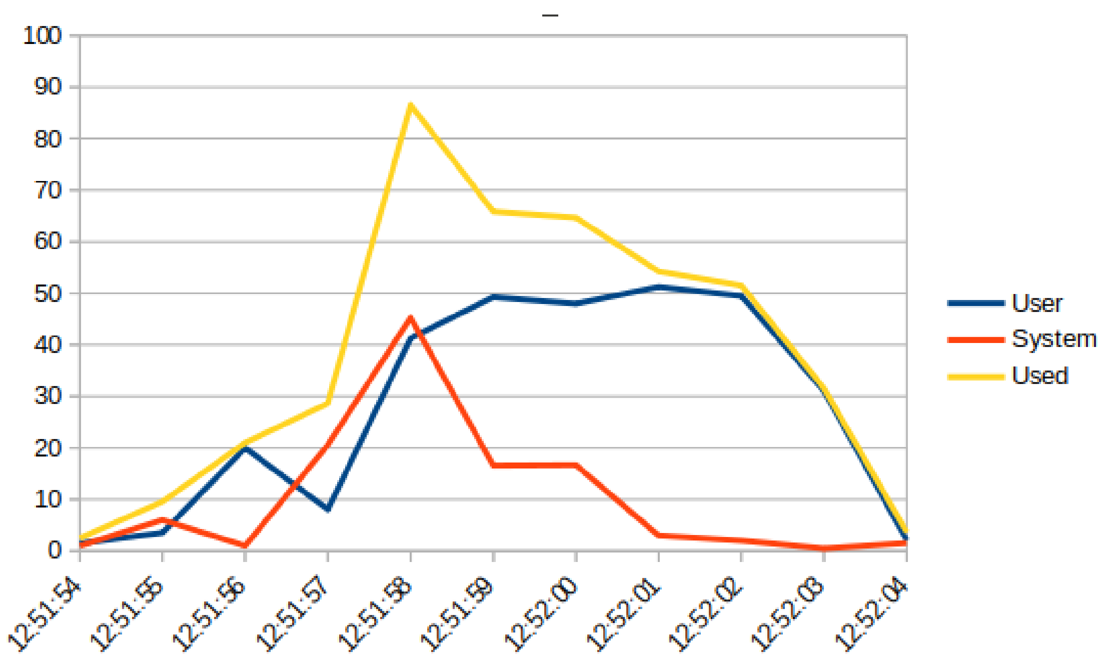
**Table 7.2** TCP flood 100 attack - Average CPU usage

| | |
|---|---|
| User - Average CPU | 27.76 |
| System - Average CPU | 10.38 |
| Used - Average CPU | 38.14 |



*(Green - incoming TCP packets, Red - blocked packets)*

**Figure 7.1** TCP flood 100 attack - Graph of incoming traffic



*(System - responsible for handling the communication, User - All user applications notably Wireshark)*

**Figure 7.2** TCP flood 100 attack - Graph of CPU usage

## 7.2 TCP handshake flood - 1000 requests

In contrast to the 100 request TCP flood, the 1000 request TCP flood resulted in constant high pressure on the hosting server, with an average CPU usage of 65.74%. The incoming traffic had an average packet rate of 15934.9, which was lower than the 100 request flood by about 300. Once again, the pressure on the hosting server was primarily due to Wireshark, which consumed about 45% of the CPU resources during the attack, with the system adding an average of 21.81%.

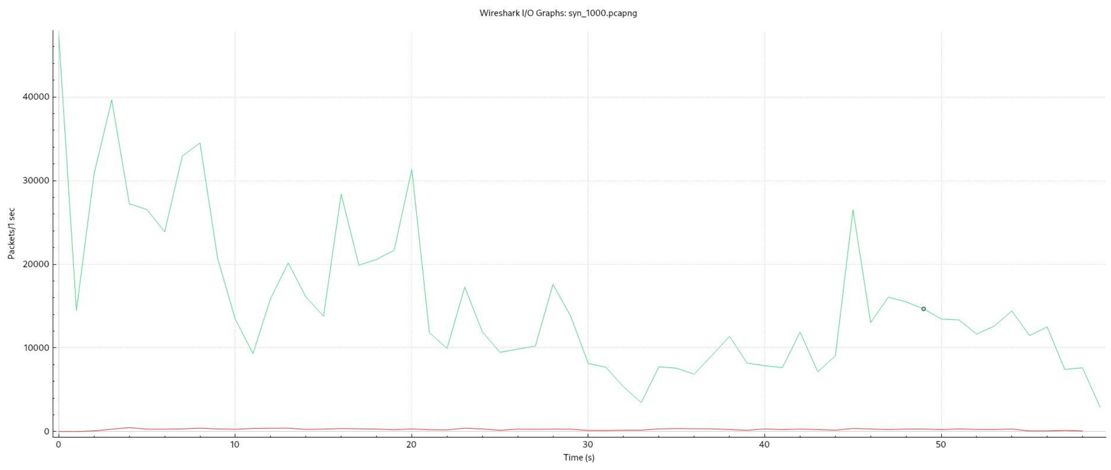
We consider this the maximum load that our setup could put on the hosting server using the TCP handshake flood attack, as an attack with more packets would only prolong the attack time. Based on our observations during this attack, we concluded that this was the maximum load that our 22 bots were able to generate using the TCP handshake flood code.

**Table 7.3** TCP flood 1000 attack - Statistics of incoming traffic

| | |
|---|---|
| First packet | 2023-04-23 15:11:04 |
| Last packet | 2023-04-23 15:12:03 |
| Time span, s | 59.512 |
| Packets | 948311 (60.5% of all communication) |
| Average pps | 15934.9 |
| Bytes | 101805272 (53.0% of all communication) |
| Average bytes/s | 1,710 k |
| Dropped packets | 15520 (1.0% of all communication) |

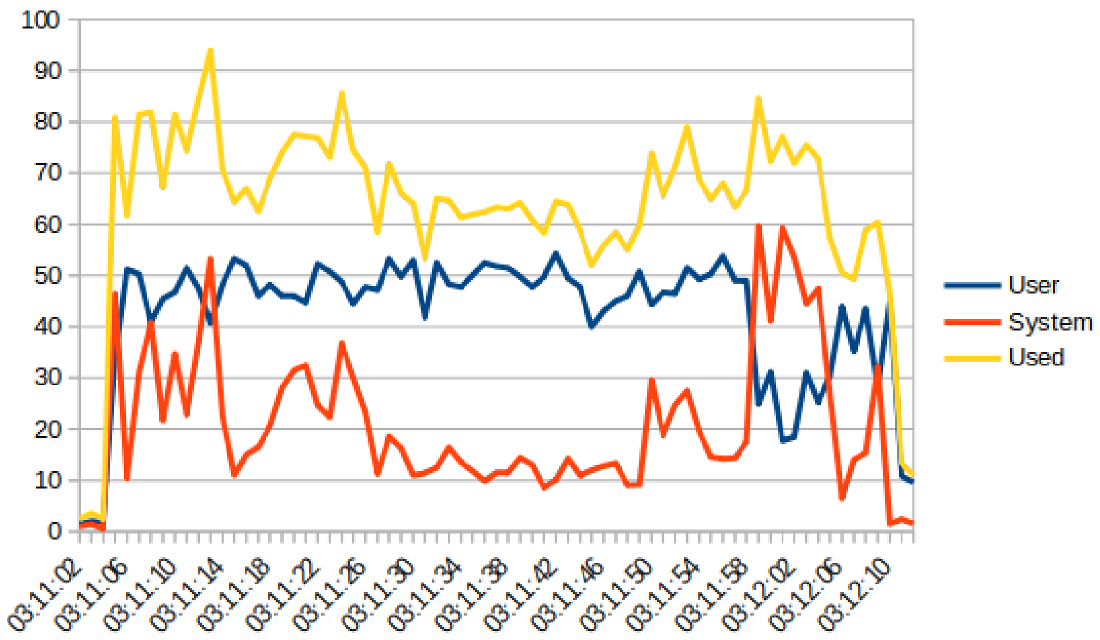
**Table 7.4** TCP flood 1000 attack - Average CPU usage

| | |
|---|---|
| User - Average CPU | 43.94 |
| System - Average CPU | 21.81 |
| Used - Average CPU | 65.74 |



*(Green - incoming TCP packets, Red - blocked packets)*

**Figure 7.3** TCP flood 1000 attack - Graph of incoming traffic



*(System - responsible for handling the communication, User - All user applications notably Wireshark)*

**Figure 7.4** TCP flood 1000 attack - Graph of CPU usage

## 7.3 GET flood - 100 requests

Our analysis of a GET flood attack comprising 100 requests revealed that it placed a heavier load on the host server compared to a TCP flood attack with 100 requests. During the GET flood attack, the average system-level CPU usage was 20.99%, whereas it was only 10.38% during the TCP flood attack. Wireshark CPU usage averaged at 37.79%.

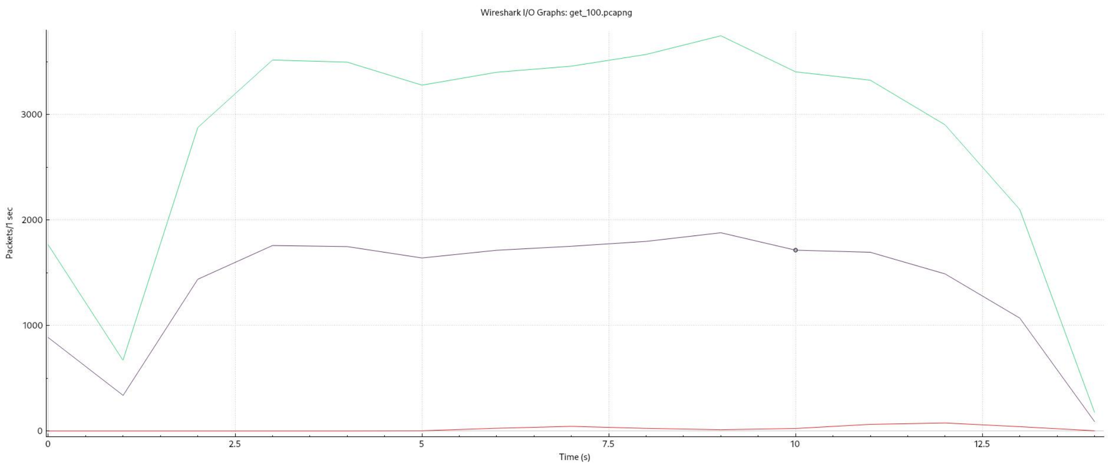
Other differences were also observed. For example, the average packets per second (pps) for the TCP flood attack was 15934.9, which was significantly higher than the pps of 2836.9 for the GET flood attack. It is worth noting that the GET flood attack lasted four times longer than the TCP flood attack. With an average CPU close to 60%, it is possible that using 37 bots could potentially result in a short-term DoS effect.

**Table 7.5** GET flood 100 attack - Statistics of incoming traffic

| | |
|---|---|
| First packet | 2023-04-23 15:05:28 |
| Last packet | 2023-04-23 15:05:43 |
| Time span, s | 14.805 |
| HTTP - Packets | 21000 (33.9% of all communication) |
| HTTP - Average pps | 1418.4 |
| HTTP - Bytes | 1875655 (5.8% of all communication) |
| HTTP - Average bytes/s | 126 k |
| TCP - Packets | 42000 (67.7% of all communication) |
| TCP - Average pps | 2836.9 |
| TCP - Bytes | 3597655 (11.1% of all communication) |
| TCP - Average bytes/s | 243 k |
| Dropped packets | 315 (0.5% of all communication) |

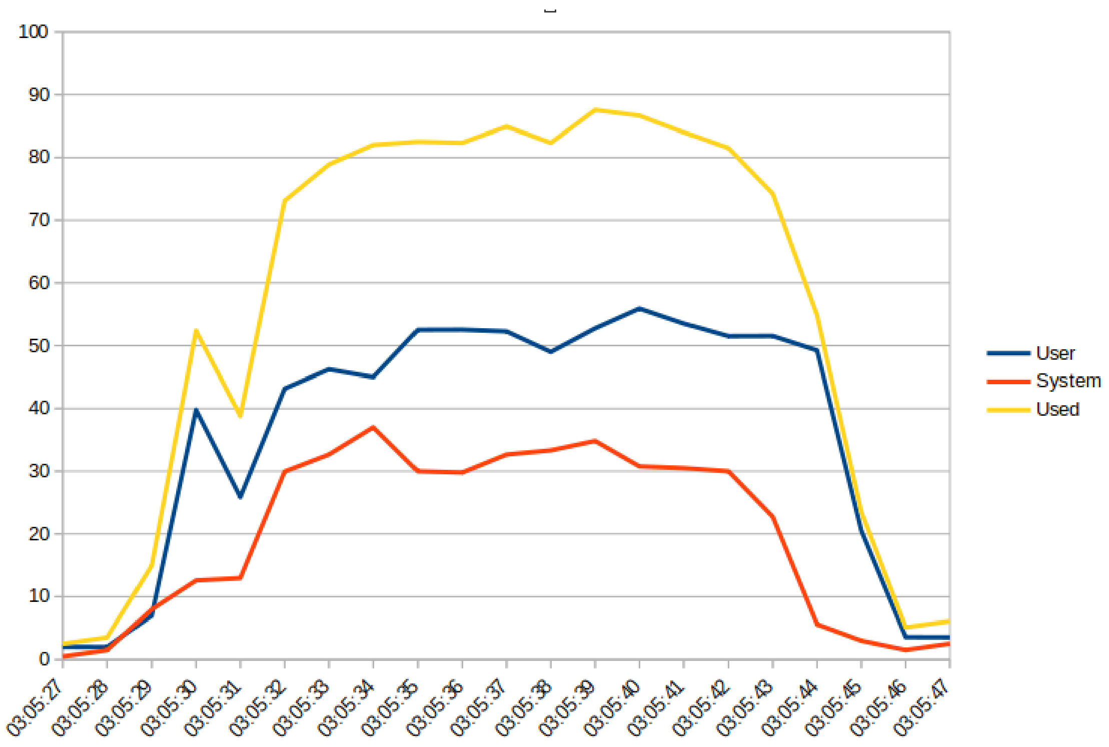
**Table 7.6** GET flood 100 attack - Average CPU usage

| | |
|---|---|
| User - Average CPU | 37.79 |
| System - Average CPU | 20.99 |
| Used - Average CPU | 58.77 |



*(Green - incoming TCP packets, Violet - incoming HTTP packets, Red - blocked packets)*

**Figure 7.5** GET flood 100 attack - Graph of incoming traffic



*(System - responsible for handling the communication, User - All user applications notably Wireshark)*

**Figure 7.6** GET flood 100 attack - Graph of CPU usage

## 7.4 GET flood - 1000 requests

Analysis of GET flood attack consisting of 1000 requests showed new aspects that were not observed in 100 request attack. The attack lasted for 123.284 seconds, during which the server received 2153645 TCP packets and 209682 HTTP packets. The average TCP pps for the GET flood attack was 17468.4. Almost 1000 pps lower than with 100 requests. The system-level CPU usage during the attack averaged 58.56%, making it the most demanding attack of our testing. Furthermore, if we focus only on the section where communication was received the average increases to 78.02% with system average of 35.17.

After analyzing the GET flood attack, we determined that our setup was not capable of generating a heavier load on the hosting server by increasing the number of requests, as it would only prolong the attack duration. Our observations during the attack led us to conclude that the maximum load our 22 bots could produce using the GET flood code had been reached.

**Table 7.7** GET flood 1000 attack - Statistics of incoming traffic

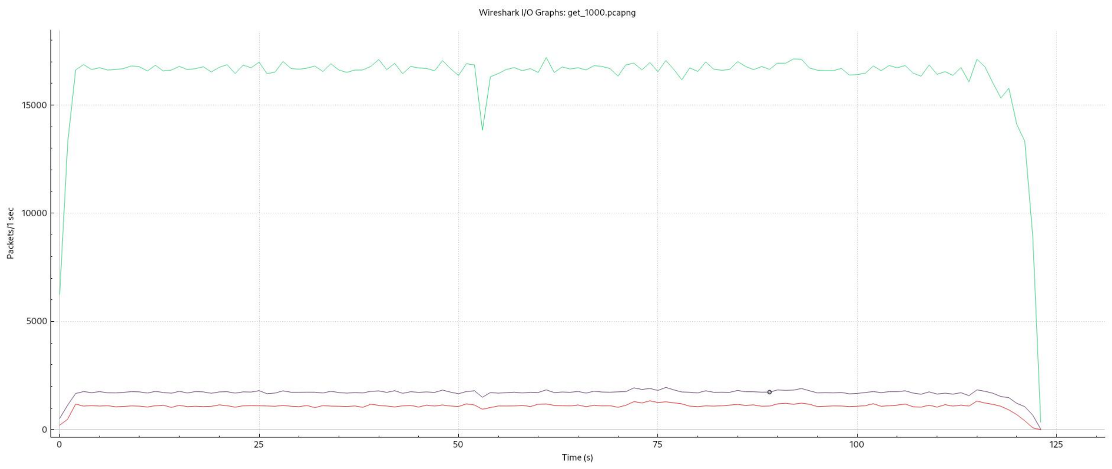
| | |
|---|---|
| First packet | 2023-04-23 15:17:36 |
| Last packet | 2023-04-23 15:19:39 |
| Time span, s | 123.284 |
| HTTP - Packets | 209682 (5.6% of all communication) |
| HTTP - Average pps | 1700.8 |
| HTTP - Bytes | 18726801 (1.1% of all communication) |
| HTTP - Average bytes/s | 151 k |
| TCP - Packets | 2153645 (57.3% of all communication) |
| TCP - Average pps | 17468.4 |
| TCP - Bytes | 147892992 (9.1% of all communication) |
| TCP - Average bytes/s | 1,199 k |
| Dropped packets | 133620 (3.6% of all communication) |

**Table 7.8** GET flood 1000 attack - Average CPU usage

| | |
|---|---|
| User - Average CPU | 38.81 |
| System - Average CPU | 19.76 |
| Used - Average CPU | 58.56 |

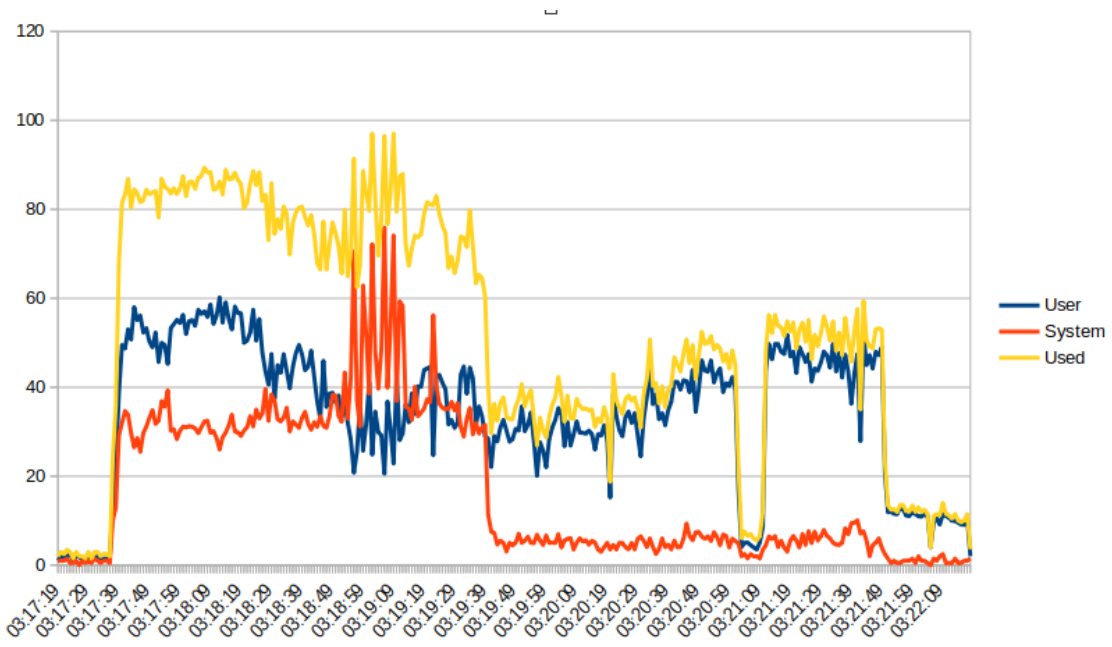
**Table 7.9** GET flood 1000 attack - Average CPU usage, while receiving communication

| | |
|---|---|
| User - Average CPU | 42.85 |
| System - Average CPU | 35.17 |
| Used - Average CPU | 78.02 |



*(Green - incoming TCP packets, Violet - incoming HTTP packets, Red - blocked packets)*

**Figure 7.7** GET flood 1000 attack - Graph of incoming traffic



*(System - responsible for handling the communication, User - All user applications notably Wireshark)*

**Figure 7.8** GET flood 1000 attack - Graph of CPU usage

## 7.5 TCP handshake flood with countermeasures - 100 requests

The difference of TCP handshake flood with 100 requests after applying the countermeasure is apparent. The attack which previously ended after 5s ended after 20 min, which subsequently lowered the pps from 19997.6 to 327.8. This was due to the hosting server blocking close to 40% of incoming TCP traffic compared to only 0.1%.

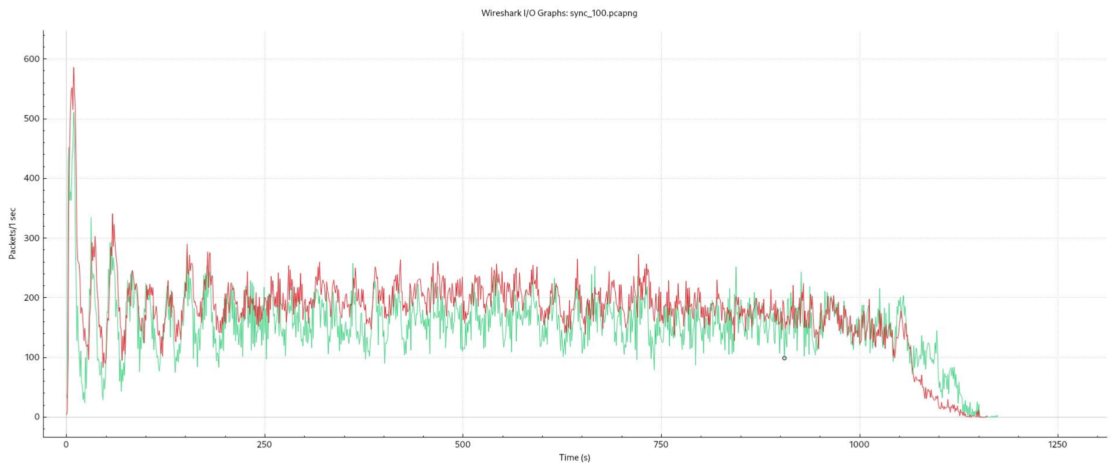
Although our measurements of CPU usage were incomplete (from 15:39:40 to 15:47:04), we still consider the data relevant because the incoming connections and the number of dropped packets remained relatively constant throughout the attack duration. The optimization and hardening of the hosting server proved effective in countering the TCP handshake flood, with the average system CPU at 3.45% and the average CPU consumption at 12.77, effectively tripling the system's ability to handle the attack.

**Table 7.10** TCP flood 100 attack, after hardening - Statistics of incoming traffic

| | |
|---|---|
| First packet | 2023-04-23 15:39:50 |
| Last packet | 2023-04-23 15:59:25 |
| Time span, s | 1174.780 |
| Packets | 385127 (72.5% of all communication) |
| Average pps | 327.8 |
| Bytes | 35862744 (46.7% of all communication) |
| Average bytes/s | 30 k |
| Dropped packets | 206277 (38.8% of all communication) |

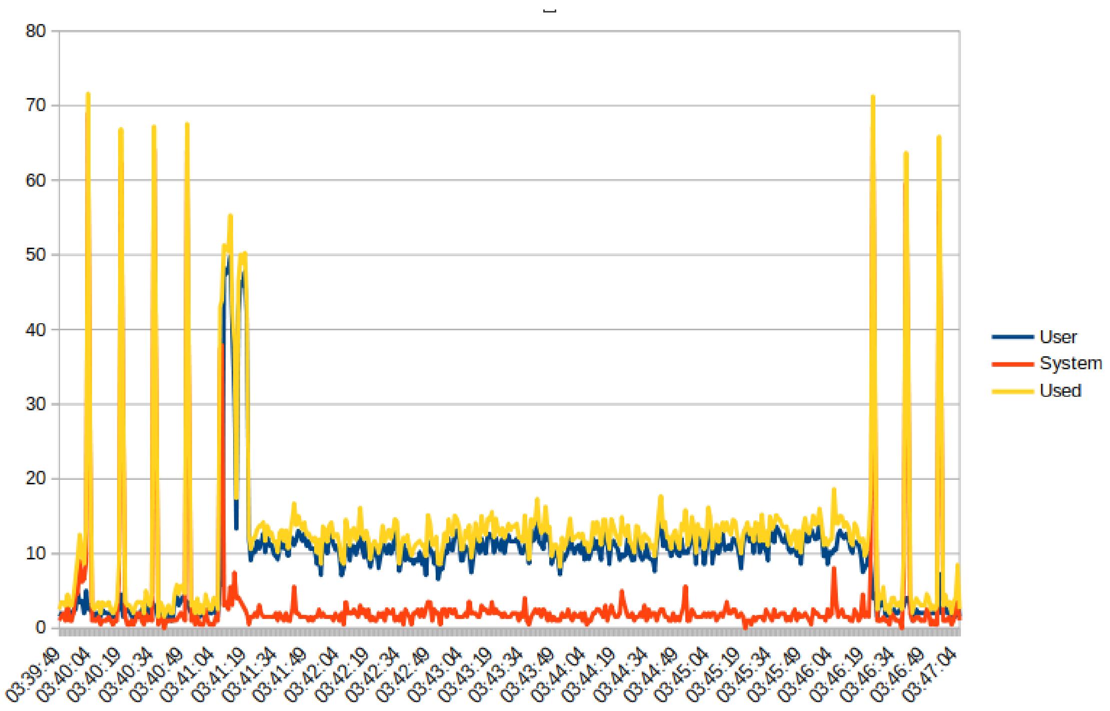
**Table 7.11** TCP flood 100 attack, after hardening - Average CPU usage

| | |
|---|---|
| User - Average CPU | 9.32 |
| System - Average CPU | 3.45 |
| Used - Average CPU | 12.77 |



*(Green - incoming TCP packets, Red - blocked packets)*

**Figure 7.9** TCP flood 100 attack, after hardening - Graph of incoming traffic



*(System - responsible for handling the communication, User - All user applications notably Wireshark)*

**Figure 7.10** TCP flood 100 attack, after hardening - Graph of CPU usage

## 7.6 GET flood with countermeasures - 100 requests

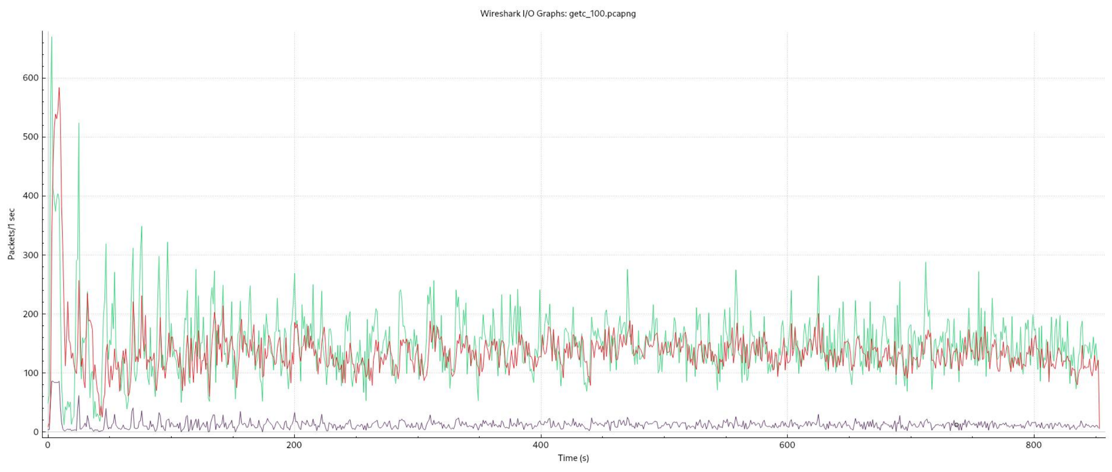
The experiment with GET flood attack with countermeasures shows significant improvements in terms of reducing the impact on the target server. The attack placed a much lighter load on the server, with an average of only 286.4 pps compared to 2836.9 pps. Furthermore, the average system-level CPU usage during the second experiment was only 9.98%, significantly lower than the 20.99% observed in the first experiment. Even with incomplete CPU data (first four minutes) we still concluded that the shown effect was enough to demonstrate the effect of hardening. In terms of dropped packets, the second experiment had a much higher rate of 33.0%, which is due to the implemented countermeasures dropping packets that were part of the attack traffic. The impact on the server was still significantly reduced, indicating that the countermeasures were successful in mitigating the attack.

**Table 7.12** GET flood 100 attack, after hardening - Statistics of incoming traffic

| | |
|---|---|
| First packet | 2023-04-23 16:04:10 |
| Last packet | 2023-04-23 16:18:23 |
| Time span, s | 852.683 |
| HTTP - Packets | 209682 (5.6% of all communication) |
| HTTP - Average pps | 12.7 |
| HTTP - Bytes | 964023 (1.2% of all communication) |
| HTTP - Average bytes/s | 1,130 |
| TCP - Packets | 244332 (69.5% of all communication) |
| TCP - Average pps | 286.4 |
| TCP - Bytes | 18635044 (24.1% of all communication) |
| TCP - Average bytes/s | 21 k |
| Dropped packets | 116067 (33.0% of all communication) |

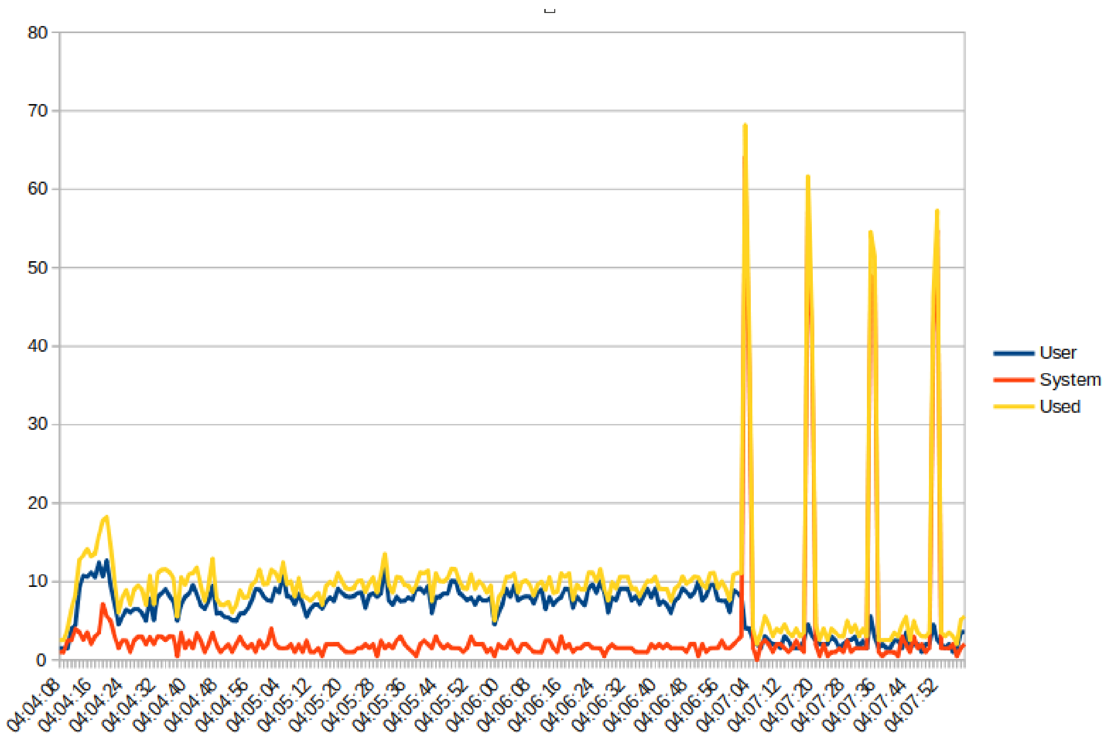
**Table 7.13** GET flood 100 attack, after hardening - Average CPU usage

| | |
|---|---|
| User - Average CPU | 6.50 |
| System - Average CPU | 3.48 |
| Used - Average CPU | 9.98 |



*(Green - incoming TCP packets, Violet - incoming HTTP packets, Red - blocked packets)*

**Figure 7.11** GET flood 100 attack, after hardening - Graph of incoming traffic



*(System - responsible for handling the communication, User - All user applications notably Wireshark)*

**Figure 7.12** GET flood 100 attack, after hardening - Graph of CPU usage

# Chapter 8
# Conclusion

The aim of this bachelor's thesis was to provide security researchers with a comprehensive understanding of DDoS attacks, including their workings, potential countermeasures, and real-life examples.

In chapter two, a taxonomy of DoS attacks was introduced, followed by an in-depth examination of botnets in chapter three, which is crucial to understanding the origin of DDoS attacks. Chapter four presented various countermeasures and categorized them for a better overview.

We then proceeded to explain common DDoS attacks and their corresponding countermeasures. By comparing the categorization of individual DDoS attacks, we established several connections, such as the effectiveness of using up-to-date versions of OS and applications to prevent bug exploitation attacks, and the mitigation of bandwidth attacks by better resource allocation or freeing up the resources that were last used.

Furthermore, we designed a testing model using the defined botnet taxonomy and constructed it with the resources provided by our faculty. We utilized the pybotnet program to demonstrate TCP handshake and HTML flood attacks. After applying countermeasures at our hosting server and measuring the attacks for the second time, we found them effective against both attacks, showcasing how our simple attacks can be mitigated.

In conclusion, our experiment was successful in achieving the goals of this thesis. There is still room for improvement and further research. For instance, more attacks can be classified and tested, different countermeasures closely studied and implemented, and the effects of smaller DDoS attacks, which are difficult to test in larger environments, can be explored. Nevertheless, we achieved a solid testbed for botnets, limited of course by the lab resources and while maintaining fair behavior towards the FIT CVUT network infrastructure.

# Bibliography

1. *Cybersecurity glossary of terms* [online]. Skillsoft, 2023-05 [visited on 2023-02-14]. Available from: `https://www.globalknowledge.com/ca-en/topics/cybersecurity/glossary-of-terms/`.
2. *Committee on National Security Systems (CNSS) Glossary* [online]. Committee on National Security Systems, 2015-04 [visited on 2023-05-09]. Available from: `https://rmf.org/wp-content/uploads/2017/10/CNSSI-4009.pdf`.
3. ESLAHI, Meisam; SALLEH, Rosli; ANUAR, Nor Badrul. Bots and botnets: An overview of characteristics, detection and challenges. In: *2012 IEEE International Conference on Control System, Computing and Engineering* [online]. 2012, pp. 349–354. Available from DOI: `10.1109/ICCSCE.2012.6487169`.
4. RAMANAUSKAITE, Simona; CENYS, Antanas. Taxonomy of DOS attacks and their countermeasures. *Open Computer Science* [online]. 2011, vol. 1, no. 3, pp. 355–366. Available from DOI: `10.2478/s13537-011-0024-y`.
5. KHATTAK, Sheharbano; RAMAY, Naurin Rasheed; KHAN, Kamran Riaz; SYED, Affan A.; KHAYAM, Syed Ali. A Taxonomy of Botnet Behavior, Detection, and Defense. *IEEE Communications Surveys & Tutorials* [online]. 2014, vol. 16, no. 2, pp. 898–924. Available from DOI: `10.1109/SURV.2013.091213.00134`.
6. *Attack signature* [online]. Kaspersky Lab, 2023 [visited on 2023-05-09]. Available from: `https://encyclopedia.kaspersky.com/glossary/attack-signature/`.
7. GRIGORIK, Ilya. *Networking 101: Transport Layer Security (TLS) - high performance browser networking (o'reilly)* [online]. O'Reilly, 2017 [visited on 2023-05-09]. Available from: `https://hpbn.co/transport-layer-security-tls/`.
8. *What happens in a tls handshake?* [Online]. Cloudflare, 2023 [visited on 2023-05-09]. Available from: `https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/`.
9. *Land attacks: Imperva* [online]. 2023. [visited on 2023-05-09]. Available from: `https://www.imperva.com/learn/ddos/land-attacks/`.
10. *DDoS definitions: Ddos dictionary: Activereach* [online]. 2023. [visited on 2023-05-09]. Available from: `https://activereach.net/resources/ddos-knowledge-centre/ddos-dictionary/`.
11. *What is an ACK flood ddos attack? - cloudflare* [online]. Cloudflare, 2023 [visited on 2023-05-09]. Available from: `https://www.cloudflare.com/learning/ddos/what-is-an-ack-flood/`.

12. *What is TCP null attack?: Knowledge base ddos-guard* [online]. 2023. [visited on 2023-05-09]. Available from: `https://ddos-guard.net/en/terminology/attack_type/tcp-null-attack`.
13. *All TCP flags flood (sometimes referred to as Xmas Flood): Mazebolt knowledge base* [online]. 2022. [visited on 2023-05-09]. Available from: `https://kb.mazebolt.com/knowledgebase/all-tcp-flags-flood-xmas-flood/`.
14. *What is a UDP flood ddos attack? — Akamai* [online]. Akamai, 2023 [visited on 2023-05-09]. Available from: `https://www.akamai.com/glossary/what-is-udp-flood-ddos-attack`.
15. *what is transport layer security?* [Online]. Cloudflare, 2023 [visited on 2023-05-09]. Available from: `https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/`.
16. RADWARE. *SSL-based DDoS attacks* [online]. 2016. [visited on 2023-05-09]. Available from: `https://www.radware.com/security/ddos-threats-attacks/ddos-attack-types/ssl-based-ddos-attacks/`.
17. *HTTP flood ddos attack* [online]. Cloudflare, 2023 [visited on 2023-05-09]. Available from: `https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/`.
18. *What is an open service FTP vulnerability, what is the risk and how can you mitigate that risk?* [Online]. Skyway West, 2021 [visited on 2023-05-09]. Available from: `https://www.skywaywest.com/2021/05/what-is-an-open-service-ftp-vulnerability-what-is-the-risk-and-how-to-mitigate-it/`.
19. EL-MOUSSA, Fadi; MUDHAR, Parmindher; JONES, Andy. Overview of SIP attacks and countermeasures. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering* [online]. 2010, pp. 82–91. Available from DOI: `10.1007/978-3-642-11530-1_10`.
20. BENCSATH, Boldizsar; RONAI, Miklos Aurel. Empirical analysis of denial of service attack against SMTP servers. *2007 International Symposium on Collaborative Technologies and Systems* [online]. 2007 [visited on 2023-05-09]. Available from DOI: `10.1109/cts.2007.4621740`.
21. *DNS flood DDoS attack* [online]. Cloudflare, 2023 [visited on 2023-05-09]. Available from: `https://www.cloudflare.com/learning/ddos/dns-flood-ddos-attack/`.
22. *DNS amplification attack* [online]. Cloudflare, 2023 [visited on 2023-05-09]. Available from: `https://www.cloudflare.com/learning/ddos/dns-amplification-ddos-attack/`.
23. OKTA. *What is a teardrop attack? definition, damage & defense* [online]. Okta, 2023 [visited on 2023-05-09]. Available from: `https://www.okta.com/identity-101/teardrop-attack/`.
24. WALLARM. *What is a teardrop attack? definition, examples, prevention* [online]. 2023. [visited on 2023-05-09]. Available from: `https://www.wallarm.com/what/teardrop-attack-what-is-it`.
25. [Online]. Cisco Systems, 2005 [visited on 2023-05-09]. Available from: `https://www.pentics.net/denial-of-service/smurf/980513_dos/sld009.htm`. May 1998 presentation at SANS 1998, from personal notes of an attendee.
26. *What is an IP/ICMP fragmentation ddos attack?* [Online]. 2023. [visited on 2023-05-09]. Available from: `https://www.netscout.com/what-is-ddos/ip-icmp-fragmentation`.
27. *What is an IP fragmentation attack?* [Online]. 2023. [visited on 2023-05-09]. Available from: `https://nordvpn.com/blog/ip-fragmentation-attack/`.
28. *What is a Smurf attack?* [Online]. 2023. [visited on 2023-05-09]. Available from: `https://www.cloudflare.com/learning/ddos/smurf-ddos-attack/`.

29. APACHE SOFTWARE FOUNDATION. *Apache HTTP Server* [online]. 2023. Version 2.4.57. Available also from: `https://httpd.apache.org`.
30. SAMAN NEZAFAT. *pybotnet* [online]. 2023. Version 2.2.3. Available also from: `https://github.com/onionj/pybotnet`.

# Content of the attached media

```
sar_data.zip ............................... CPU usage data measured by sar command
   tcp_100.txt ........................................ sar - tcp flood, 100 requests
   tcp_1000.txt ...................................... sar - tcp flood, 1000 requests
   tcp_100_countermeasure.txt ....... sar - tcp flood, 100 requests with countermeasure
   get_100.txt .................................. sar - HTTP GET flood, 100 requests
   get_1000.txt ................................ sar - HTTP GET flood, 1000 requests
   get_100_countermeasure.txt sar - HTTP GET flood, 100 requests with countermeasure
wireshark_data.zip ............... Network communication data measured by wireshark
   tcp_100.pcapng ............................... wireshark - tcp flood, 100 requests
   tcp_1000.pcapng ............................. wireshark - tcp flood, 1000 requests
   tcp_100_countermeasure.pcapng ............. wireshark - tcp flood, 100 requests with
   countermeasure
   get_100.pcapng ........................ wireshark - HTTP GET flood, 100 requests
   get_1000.pcapng ...................... wireshark - HTTP GET flood, 1000 requests
   get_100_countermeasure.pcapng ..... wireshark - HTTP GET flood, 100 requests with
   countermeasure
FITthesis_latex.zip ............ zip file containing all LaTex files to generate this thesis
```