UNIVERSITAT DE BARCELONA

Final Degree Project

# DDoS Hive Comb

An IP Stresser made by the community.

Made by Laurentiu Nedelcu

Directed by Raül Roca Cánovas
Made in the Department of
Mathematics and Computer Science

Barcelona, 20 June 2021

# Abstract

Denial of service attacks (or DoS attacks) consist of attacking a service on a server in order to temporarily interrupt the services of the host connected to the Internet. There are a wide variety of techniques and strategies that can be used to achieve this goal.

This work aims to help these victims by creating an IP Stresser that allows the simulation of various DoS attacks. In this way, victims can check the stress capacity that their teams can handle and adapt different mitigation plans in various types of circumstances. The IP Stresser that will be developed in this project consists of a website accessible to everyone for free. The idea of this website is to try to create a community where with the help of other users a botnet can be created by means of a script that can be downloaded from this same website.

Although this work is more focused on the practical part, we will try to explain briefly the structure of the Internet and how these attacks act and influence society.

# Resumen

Los ataques de denegación de servicio (o ataque DoS) consisten en atacar un servicio de un servidor con el fin de interrumpir temporalmente los servicios del anfitrión conectado a Internet. Hay una gran variedad de técnicas y estrategias que se pueden usar para conseguir este objetivo.

Este trabajo pretende ayudar a estas víctimas creando un *IP Stresser* que permite la simulación de varios ataques DoS. De este modo, las víctimas pueden comprobar la capacidad de estrés que pueden manejar sus equipos y adaptar diferentes planes de mitigación en varios tipos de circunstancias. El *IP Stresser* que se desarrollará en este proyecto consiste en un sitio web accesible a todo el mundo de forma gratuita. La idea de este sitio web es intentar crear una comunidad donde con la ayuda de otros usuarios se pueda crear una *botnet* mediante un script que se deberá descargar desde este mismo sitio web.

Aunque este trabajo está más centrado en la parte práctica, intentaremos explicar, a lo largo del documento, por sobre la estructura de Internet y de qué manera estos ataques actúan e influyen en la sociedad.

# Abstracte

Els atacs de denegació de servei (o atac DoS) tracta d'atacar un servei d'un servidor amb la finalitat de interrompre temporalment els serveis de l'amfitrió connectat a Internet. Hi ha una gran varietat de tècniques i estratégies que es poden usar per aconseguir aquest objectiu.

Aquest treball pretén ajudar a aquestes víctimes creant un *IP stresser* que permet la simulació de diversos atacs DoS. D'aquesta manera, les víctimes poden comprovar la capacitat d'estrès que poden manejar els seus equips i adaptar diferents plans de mitigació a diversos tipus de circumstàncies. El *IP stresser* que es desenvoluparà en aquest projecte consisteix en un lloc web accessible a tothom de forma gratuïta. La idea d'aquest lloc web és intentar crear una comunitat on amb l'ajuda d'altres usuaris es pugui crear una *botnet* mitjançant un script que s'haurà de descarregar des d'aquest lloc web.

Tot i que aquest treball està més centrat en la part pràctica, intentarem explicar, al llarg del document, per sobre l'estructura d'Internet i de quina manera aquests atacs actuen i influeixen en la societat.

# Index

# Chapter 1

# Introduction

This first chapter will be an introduction to the Internet of things. It will explain how we are connected to the Internet in order to understand the purpose of this project and its content.

## 1.1 Internet of things

The **Internet of Things (IoT)** describes the network of physical objects that are connected to the Internet and that allows them to communicate with each other, that includes sending, receiving and exchanging data. Therefore, we consider that a physical object belongs to this network if it is capable of communicating with other physical objects through the Internet.

> ***Note:*** *during the project we will address to the physical objects connected to the Internet as machines, computers, servers or users. When we say users, we do not mean the human being but the computer with which it is connected to the network.*

## 1.2 Internet

The **Internet** is the biggest world-wide communication network of computers. The Internet is not centrated in a single network that is governed by a particular organization. Its structure is somewhat complex to explain since it consists of the grouping of some networks connected to others, which are made up of various cloud infrastructures and resources in general. We will try to explain its structure in a simple and necessary way just to understand the content of the project since it is not required to understand it in its entirety for this specific topic.

To begin with, although there is not a main organization that is in charge of the Internet, there are a number of organizations responsible for the adjustment of resources and the development of protocols necessary for the Internet to work properly. All of them work cooperatively from their respective roles in order to create policies, rules and resources that allow maintaining the global interoperability[1] of the Internet for the good of the public.

As we have already said, thanks to the internet, several machines can communicate with each other to send, receive or exchange data. In order for two or more machines to be able to communicate over the network, they must be able to recognize each other. For this purpose, there are **IP addresses** which allow each of these machines to be identified following the Internet protocol (IP).

[1] The ability of two or more systems or components to exchange information and use the information exchanged.

### 1.2.1 The flow of information on the Internet

The information transmitted on the Internet is divided into several blocks which are called **packets**. In order for computers to exchange information between them, it is necessary that they follow a protocol which indicates the regulation of the transmission of the packets. The most used protocol today is **TCP / IP**. In short, the TCP protocol is responsible for fragmenting the message sent into packets, then at the destination the packets will be reorganized to form the complete message again and thus deliver it to the corresponding application. On the other hand, the IP protocol is in charge of routing the packets, in this way the packets that make up a message can reach the destination by different paths.

> ***Note:*** *the OSI model is designed by the ISO organization in order to theoretically define the functions of a network system. While the TCP / IP protocol is the protocol used in the practical world and with which different computers can communicate with each other.*

### 1.2.2 Client-server architecture

Communication between two machines, which use the TCP / IP protocol, follow the client-server model, which consists of one machine acting as a client and the other as a server. This model is a network architecture in which one client (or more) requests and receives a resource from a centralized server. Client machines usually have a friendly interface to allow a user of that computer to request services from the server and at the same time show the results that the server sends in response to that request (normally the requested service if there was no error). On the part of the server, it must wait for a client to make a request and after processing it, respond with the requested information. Ideally, a server should offer a transparent interface to clients so that they do not know about the specific characteristics of the system that provides the service. In this way, it is possible to protect both the private information that the server may contain and possible modifications that it may suffer from an unwanted user.

The clients are usually users on personal or work computers, while the servers are in more specific places (such as in data centers) and on more powerful machines. This is adapted to the needs of each one and of the different tasks that they must perform on a routine basis. After all, one client typically makes one request at a time, while servers must respond to multiple clients simultaneously. However, a single client can use multiple servers, but still does not usually require the same hardware capabilities as a server.

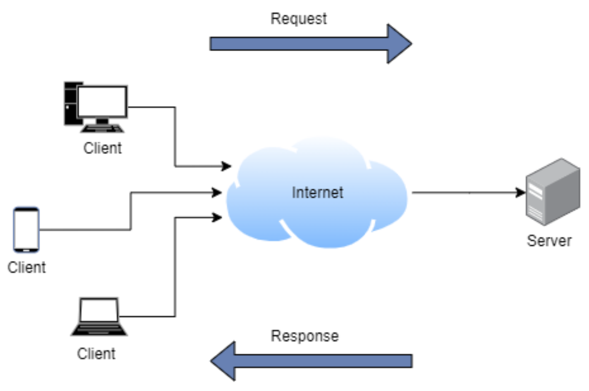


FIGURE 1.1 Client-server model. Three different devices and a server (physical objects) are connected to the Internet. Clients make requests to the server and the server responds to each request from each client.

Mention that this is not the only model that exists and that there is another one that is used less frequently called **peer to peer (P2P)**. The main difference between these two is that in the client-server architecture, there are clients designed to make requests to the server and the server provides these services, while in P2P systems, peers act both as service providers and as a service to the consumer. There are more characteristics that differ them, as is the case that the client-server model is much more expensive since it requires cloud infrastructures, more powerful computers and other types of resources; but in return, it offers a level of access to the client that allows protecting the resources provided by the server. In the case of P2P, security is often considered worse because the responsibility lies entirely on how users handle it. In addition, by not presenting this transparency interface, that we have discussed in the client-server model, it makes it a more vulnerable system in general.

For this project we will focus on the client-server model since it is the most used by companies due to the advantages it offers. Although servers can be very expensive, the security and stability they offer make them the preferred network model for most companies. Peer to peer systems are usually more used in situations where security is not the primary concern, as may be the case of home networks or small companies (by preference or because they cannot afford the expenses). In section 3.2 Attackers motivation, we will see in more detail why we are focusing so much on this model and why we take into account company networks and not home networks.

# Chapter 2

# Project motivation

Following the client-server model explained, both components are essential for a fluid communication. The fall of one of these two components would cause the communication to be interrupted and the messages issued by the sender to go unanswered. Typically this phenomenon affects more the organization or owner(s) of the service provider than the customer itself. After all, the client loses connection to a single server, while the service providers lose many more clients which can lead to greater losses (usually monetary).

Failure of these servers can be due to hardware, software, or both at the same time. The reasons for these problems can occur "naturally" such as poor server maintenance, power failure or natural accidents (eg flood, physical damage). However, it could also be intentionally done by a third party. After all, just like the devices users use, servers are also computers and are vulnerable to viruses and malware.

Although depending on the main objective of this individual or external organization, whom we will call an attacker from now on, it is likely that their goal is not to infect the servers and they only seek to make them stop working for different reasons that we will see later. To achieve this goal, the attacker can subject these servers to what is known as a **Denial-of-Service attack (DoS attack)**. We will explain these attacks in much more detail in section 3 Denial-of-Service attack but briefly, as its name suggests, these attacks are based on denying the service of the servers by making them unable to communicate with their clients.

> ***Note:*** *In order to stop the servers from working, a third party can take a more violent action by physically breaking down the systems. We will not address this point of view during the project and will focus on the attack named in the previous paragraph.*

The **impact** that these attacks can cause on the attacker's target (usually companies) can negatively affect their reputation. In addition, it can directly interrupt a network resource with which the target generates income, as is the case with different electronic businesses. It not only affects the machines, but it can also affect the employees of a company since it can prevent them from carrying out their tasks and responsibilities. Furthermore, even if it is an attack that seems straightforward, it can be used to hide other malicious activities that could affect the personal safety of the target. A clear example of this case is the ransom DoS attack.

We have seen that being under this type of attack can harm the target in many ways, especially in the economic part. In addition, if the attacker's target generates income through the services it offers to its clients, as we have already said, it could suffer a great monetary loss. However, it is not the only facet that would affect the economic factor since stopping these type of attacks or repairing the damage caused, could lead to losses of millions of euros for some companies. A Kaspersky Lab study supports these facts as seen in the following article [1].

> *"[...] the financial impact of a Distributed Denial of Service (DDoS) attack is continuing to rise globally – totalling over $120K for SMBs and costing enterprises over $2M, per attack on average.*
>
> *Kaspersky Lab's IT Security Risks Survey 2017* shows that the average cost of a DDoS attack on organizations has risen dramatically over the past year. Whether as the result of a single incident or*

> *when DDoS has formed part of a multi-faceted cyberattack, the financial implications of reacting to a DDoS attack in 2017 is $123K for SMBs, compared to $106K in 2016.*
> *For enterprises, the cost has soared to more than half a million dollars – from $1.6M in 2016 to $2.3M in 2017, on average. The rising financial costs of DDoS attacks, coupled with unquantifiable impacts such as reputational damage, is crippling for many organizations.*
>
> *'DDoS attacks, both standalone or as part of an attack arsenal, can cost an organization thousands, if not millions – that's without counting reputational damage and lost clients and partners as a result,' said Kirill Ilganaev, head of Kaspersky DDoS protection, Kaspersky Lab."*

As specified in the article, the negative impact affecting companies continues to grow dramatically year after year. And this is because the presence of these type of attacks are increasing considerably. According to the statistics compiled by the Kaspersky study [2], the number of attacks has skyrocketed from 2018 to 2020. Furthermore, although there are times of "calm", there is no indication that the number of attacks will decline. As stated in the study, there was a decrease in attacks in Q3 of 2020 compared to Q2 of the same year, but that was because in Q2 the growth was too high.

Although a decline was not expected after so many years of growth, it is confident that the DDoS market would begin to stabilize. A more recent report also from Kaspersky [3], stated that in Q4 2020 some of its predictions were fulfilled, such as the stabilization of these attacks during the year. Although it should be noted, that they are still higher than in Q4 of 2019.

> ***Note:*** *when we talk about the number of attacks witnessed, we must take into account two factors: the total number of attacks received in a period of time and the amount of time that those received attacks have lasted.*

With this, it is expected to inform and understand the impact that is caused by this type of attacks against our society. Este proyecto consistirá en crear una comunidad donde podamos ayudar a enfrontar estos ataques DDoS. Para ello, crearemos un IP stresser que ayudará a distintos individuos o colectivos a testear sus planes de mitigación. No obstante, esta aplicación no tendrá ningún interés financiero y está hecho con la finalidad de buscar individuos que quiera aportar su granito de arena para crear las herramientas y mejorar las funcionalidades de este producto que venderemos totalmente gratuito. Al largo del proyecto, nos aseguraremos que el lector pueda entender que es un IP stresser y cómo funcionan estos ataques a la vez que se describirá mejor su impacto. Además, he escogido esta tematica porque quiero aprovechar este trabajo para poder profundizar mis conocimientos sobre las redes y como funcionan los protocolos web. También aprovecharé para explorar nuevas tecnologías que podrían ser de interés para la página web.

# Chapter 3
# Denial-of-service attack

As we have been explaining, a **denial-of-service attack (DoS attack)** is an attack that causes a computer system to drop by making its services not  to be accessible to other users on the network. In this way, we can say that the victim loses connectivity to the Internet due to the saturation of traffic that their ports may suffer or due to an overload of consumption of their resources. These attacks are originated from a single machine, this being the only source attacker that is saturating the victim's network as it can be seen in figure 3.1.

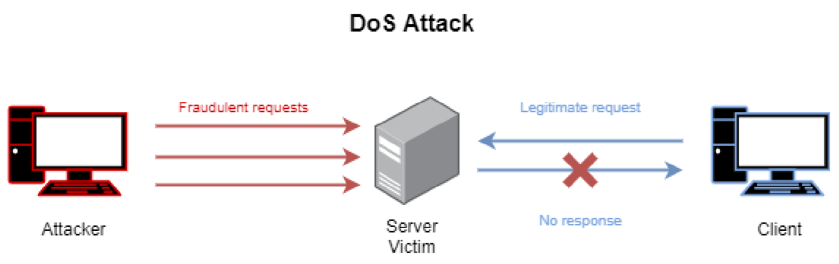


FIGURE 3.1 DoS attack. An attacker is sending fraudulent requests with the purpose of saturating the victims network from a single machine. As the server is out, the legitimate requests by clients will not get a response.

An enhancement of these attacks are the **distributed denial-of-service attacks (DDoS attacks)** which generate an even greater flow of traffic from various connection points (Figure 3.2). The fact of using multiple source nodes allows to amplify the capabilities of the attack, but at the same time it also makes more difficult to find the identity of the attacker. Therefore, it generally hampers mitigation strategy efforts.

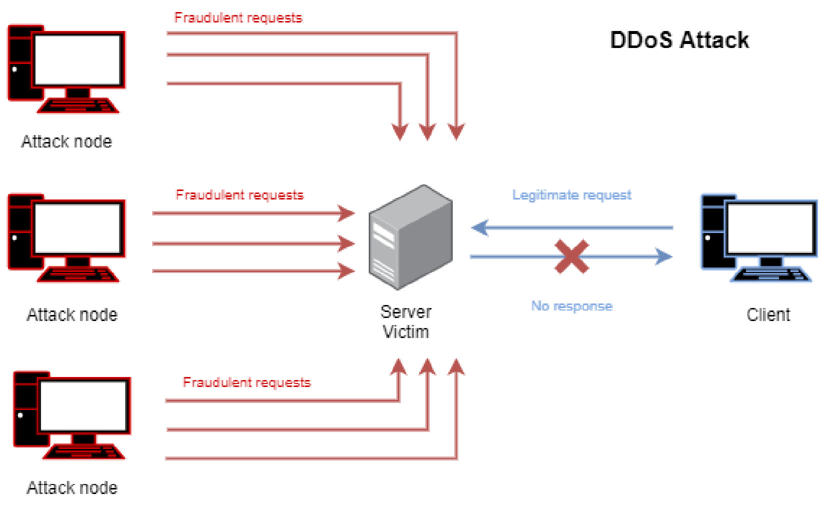


FIGURE 3.2 DDoS attack. An attacker is sending fraudulent requests with the purpose of saturating the victims network from different machines. As the server is out, the legitimate requests by clients will not get a response.

## 3.1 Botnet

The most common method of achieving DDoS attacks is for the attacker to own a **botnet**. A botnet is a set of devices connected to the Internet (Internet of things) whose security has been violated and control has been transferred to a third party. Each infected device is referred to as a bot. These bots are controlled by the attacker through the botnet, which can direct activities to these components through communication channels formed by network protocols such as IRC or HTTP.

The botnet architecture evolves in order to evade detection and strengthen the link between bots. As a general rule, botnets are known by the client-server structure (figure 3.3). In this way, the herding bot (the botnet controller) is able to perform all control from a remote location. The set of servers and elements that allows controlling the botnet is called the **command and control infrastructure (C&C or C2)**.

Using the client-server model guarantees command delivery and limits the complexity of communication. However, having a single command center server (or few) can be inconvenient for attackers. On the one hand, this causes a lot of traffic to the C&C since the bots must report the results of the request to the herding bot, which generally makes it easier to detect the botnet's controller. On the other hand, it is needed to keep track of the bots (evaluate availability), and in case a device is "cured" as in the case of routers[1] it would be necessary to re-infect them. All these maintenance and analysis operations can be very expensive for a single server and delegate these responsibilities to several intermediate nodes[2] could speed up the work and considerably decrease the traffic to the herder bot.

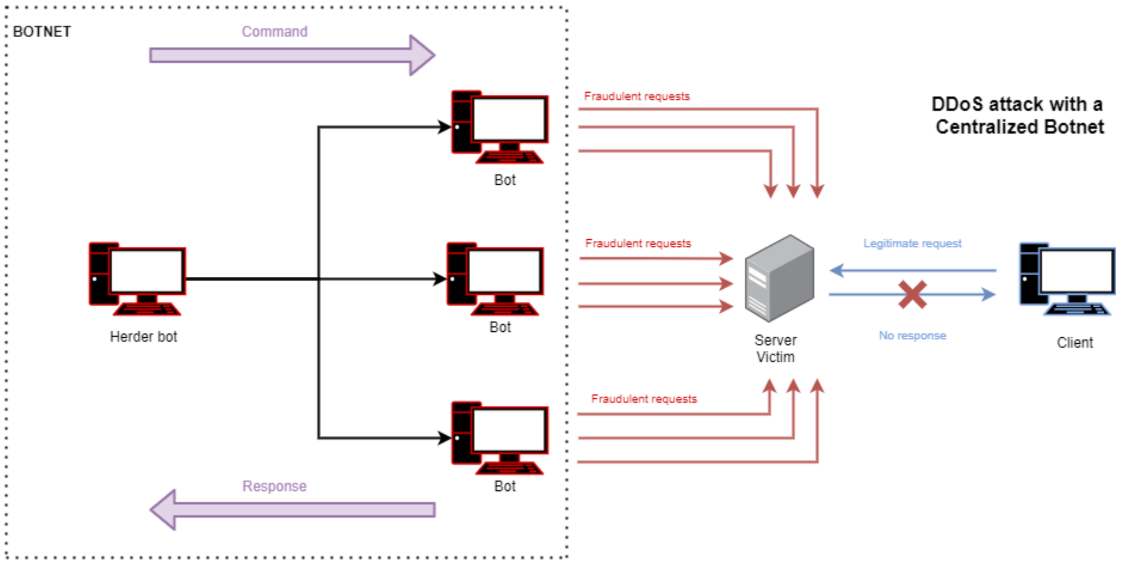


FIGURE 3.3 Client-server Botnet. A botnet that performs a DDoS attack. A herder bot sends a command to each bot to perform the attack. The external bots report the results with a response back to the herder bot.

The latest and most modern Botnets rely on P2P networks to communicate (figure 3.4). The main advantage of this method is that it protects the botnet from the vulnerabilities of a centralized C&C, where finding and eliminating the shepherd node eliminates the entire network. In P2P networks, we are facing a network where bots act as in the client-server model, but a central node is not required for communication. In other words, P2P bots act both as a command dispatcher and as a client receiving a command. This way, if a node fails, the attacker would not be in trouble as in the case of centralized botnets. However, this adds more complexity to the communication system, which ends up being a double-edged sword. Since on the one hand the complexity grows, making it more prone to the failure of messages received by a node. But on the other hand, as we have already said, it is more difficult to detect the origin of the attack and thus eliminate the botnet from the network.

[1] routers, like any other device connected to the network, belongs to the IoT, which are vulnerable to malware attacks. With simple malware, by restarting the router the infection disappears as it is removed from the RAM memory.

[2] the intermediate nodes are those bots that are between the herding bot and the external bots (which are in charge of performing the command). These can be in charge of monitoring a determinate number of external nodes and thus notify the herder node. In this way, the work and traffic of the herder node is facilitated.

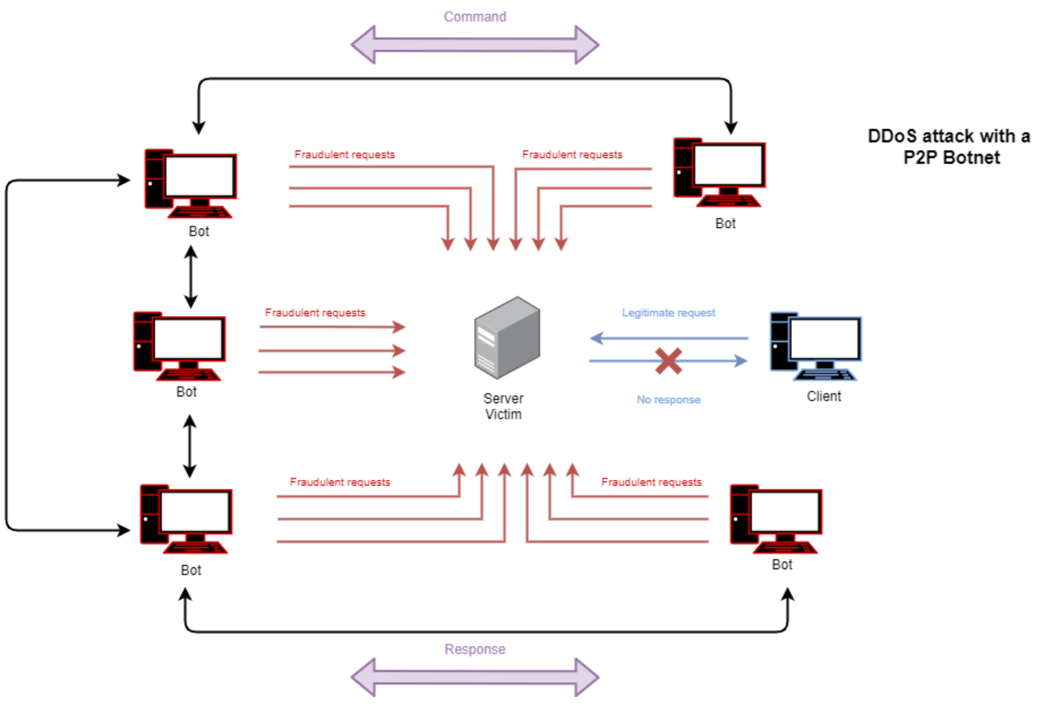


FIGURE 3.4 P2P Botnet. A botnet that performs a DDoS attack. Each bot can receive and send a command to other bots. The route between the bots could be different, this is just an example.

## 3.2 Attacker motivation

So far we have been able to see what DDoS attacks are and their impact on a target. However, we have not yet mentioned all the possible targets of these cybercriminals and why they want to attack them.

At first glance, and perhaps the most obvious reason from the outset, it is because of the extortion of money from the victim. Generally, the attacker will contact the target anonymously in order to ask for a certain amount of money to stop the attack or prevent an attack in the future. The attacker usually attacks them beforehand to demonstrate to the victim their capabilities to perform a DDoS attack and they have it under control by acting at a specific time. The money transfer is usually done on a virtual currency such as Bitcoin.

However, there are more reasons such as promoting a political idea. The attacker at this time, usually makes these attacks public through a social network or other types of public forums where they express their ideologies or their complaints. The victims in these cases are usually large brands or companies, which often suffer reputational damage from news coverage.

Another reason may be due to rivalry between companies, performing these attacks with the aim of financially impacting or embarrassing a commercial competitor. These attacks are often long-lasting and focus on the resources that generate income for them. The fact that there are DDoS services under contract, where someone can buy a DDoS attack for low cost, has made these cases easier and more common [4].

There are also more personal reasons or could be said for personal vendetta. This dates back to the origin of DDoS attacks, where there were online disputes between small groups (especially through chat systems). The main purpose of these attacks is to "punish" for a perceived wrong [6].

There are also attacks sponsored by the state or initiated by a terrorist organization. The main goal is usually to silence the victim, in this way they focus on attacking communication infrastructures and commerce. These attacks are usually much longer and better organized since the attackers usually have much more resources than in a general attack [7].

## 3.3 DDoS categories

Next we will see different types of DDoS attacks, and we will see that there are different techniques that allow denying the service of a server without the need to send a lot of traffic. We will comment a bit on how we can classify these attacks, so that it is easier to identify the type of attack when we see them during the practice. Moreover, we will mention as an example only those attacks that have been carried out during the project.

> ***Note:*** *it is recommended to take a look at Appendix A, Internet before continuing in case it is needed to refresh some network concepts like how it is structured and how computers communicate with each other over the Internet. Moreover, examples of all the attacks explained will be seen in the section 5, Results of the project.*

### 3.3.1 Volumetric DDoS attacks

The most common form of attack that stands out for its simplicity. Volumetric attacks are based on trying to saturate the target's network. As already explained, the Internet is created from individual networks interconnected with each other. In this way, to access small networks through the Internet it is necessary that the ISP give us access to it. All these connections between networks transmit a volume of data which we call traffic. The traffic capacity that an individual or organization can send and recieve from other networks is determined by the technical capacity or by the limitation of the contract that has been signed with an ISP provider. Without taking this limitation into account, it is difficult enough to increase the link capacities, since it is required to increase the pipe size to improve the bandwidth, which would be equivalent to an investment of money and time. Therefore, thanks to these bandwidth limitations, there are volumetric DDoS attacks that take advantage of the small network capacity of the target compared to the overall capacity of the set of devices that are connected to the general network.

Figure 3.3 seen above illustrates a widespread volumetric DDoS attack where the target has a link capacity limited by the bandwidth contracted (or available). The attacker through a botnet sends a command from the herding bot to all bots to send traffic to the target at the same time. Since the victim's ISP is not capable of handling so much data volume caused by the DDoS, it reaches a point that it loses (or does not let through) much of that data, including those that come from legitimate clients.

For example, the target may have a 2 Gbps link (the highest speed offered by any major provider [8]) and each node in the botnet is capable of sending 20Mbps traffic. So, with a botnet of only 100 bots, the attacker could saturate the victim's network (100 * 20Mbps = 2Gbps). Finding botnets of that size or larger is not a big challenge since in the early 2000s, the average size of a botnet was said to be 20,000 bots [9] and, currently, botnets of 100,000 bots have already been reported (they are already common nowadays).

#### Attack impact

The main effect of volumetric attacks is the saturation of the link that connects the victim's network to the Internet. This causes the loss of packets sent from legitimate clients or the outage of the network. The percentage of packets lost will depend on the traffic excess. Furthermore, these attacks cause more resources to be used from the target devices, such as routers or switches. If these devices are not well configured to handle such amount of traffic, they will end up degrading their performance until they even hang or reboot. Moreover, if the victim uses a payment plan for the amount of traffic used by their network, they can get a very expensive bill since the traffic coming from volumetric attacks, even if they are not wanted, are still packets transmitted over the network.

#### Effectiveness

The fact that any Internet user can be a victim of these category of attacks shows the simplicity of them. The attacker does not require any knowledge of the target's network, just knowing their public IP address and having the ability to launch the attack (sending packets) is more than enough. In addition, these attacks must be stopped before passing through the ISP, so the participation of service providers is necessary to mitigate them.

Also comment that any type of packets sent is useful, it is not necessary to take into account a specific protocol. This also applies if the ports or services of the target are closed or not available, since as we already said, this affects the entrance of the target's network and by then they are not yet processed. Therefore, since volumetric attacks do not rely on established connections, the attacker can spoof the source IP addresses to hinder the trace.

#### Ping flood

This volumetric attack generates a large number of ICMP packets. The attacker overwhelms the victim with ping packets (echo request), without waiting for responses (echo replies). The packet size is usually constant where the content of the message is usually a repeating character or random text. The source IP address is usually forged since, as we have said, the attacker does not need the target's responses.

#### UDP flood

The attack consists of generating a large volume of UDP packets that will be sent to the victim's IP address. The size of the packets, as in the previous case, are of constant length with the same type of content in the messages. The targets of these attacks are usually services that use these type of packets, such as DNS, which makes its mitigation difficult since it is the same type of packets that is used for legitimate communication.

### 3.3.2 Protocol DDoS attacks

Protocol attacks are based on the collection of attacks that try to break the environment by exploiting the various vulnerabilities or inefficiencies of the victim's network. Many of these attacks exploit the weaknesses of the protocols used in layers 3 and 4 following the OSI model. Unlike volumetric attacks, these attacks do not attempt to saturate the Internet connection but rather to break it with a small amount of traffic. In these cases, each attack has its peculiarities and its behaviour varies depending on the characteristics of the protocol used to carry out the attack.

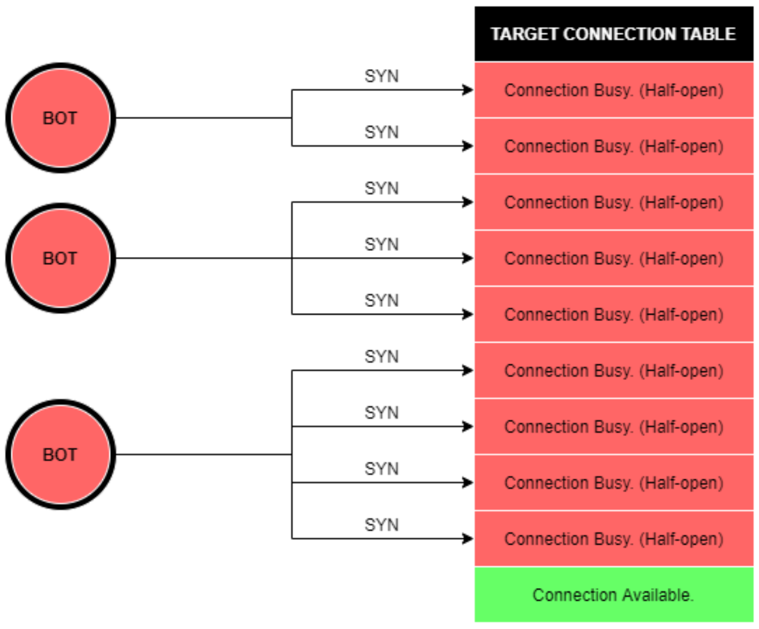

TARGET CONNECTION TABLE
BOT
BOT
BOT
SYN
SYN
SYN
SYN
SYN
SYN
SYN
SYN
SYN
Connection Busy. (Half-open)
Connection Busy. (Half-open)
Connection Busy. (Half-open)
Connection Busy. (Half-open)
Connection Busy. (Half-open)
Connection Busy. (Half-open)
Connection Busy. (Half-open)
Connection Busy. (Half-open)
Connection Busy. (Half-open)
Connection Available.

FIGURE 3.5 TCP DDoS attack. Each bot from the botnet can make 1 or more connections with the target just like when we use (as legitimate clients) multiple sessions across diferents tabs in the same browser window. If all entries in the table are busy, other clients will not be able to connect to the server.

### Attack impact

Although each attack may impact slightly differently, in general they all affect the victim in a common way. To start with, some devices connected to the victim's network will be affected in terms of memory usage. Those devices used to track the network such as firewalls, IDS / IPS or elastic load balancing use much more memory when they suffer a DDoS attack since each request sent by the attacker will get assigned a place in memory to establish the connection. In addition, these same devices will consume more CPU to be able to carry out their operations. For example, this behavior is common in TLS / SSL protocols as they decrypt traffic but at considerable CPU cost.

Other attacks focus on exceeding the limits of buffers and tables that store physical memory. These have a storage limit, which can be totally overwhelmed by the attacker's requests. Therefore, all requests that come from legitimate clients will be rejected as it will not be possible to store more.

### Effectiveness

These attacks are more complex to detect, as they are low bandwidth attacks and may not be noticed. Protocol attacks that seek to interact with the target's network devices can go unnoticed since their behavior can be understood as inactivity.

Even if there are large companies that have many servers with thousands of CPU cores and hundreds of tera bytes of memory, they can be taken down by medium-sized botnets (100,000 bots). So, regardless of the size of the victim's infrastructure, the Internet will always be bigger and favor the attacker with many resources.

Furthermore, like volumetric attacks, some attacks do not require a valid source IP address, so it can be potentially spoofed.

### SYN flood

These attacks attempt to exhaust the resources of the target device by not completing the normal TCP 3-way handshake. The attack (figure 3.5) starts as in a normal process where the attacker will send a SYN packet as a client that wants to establish a connection with the server. The target's server will add a semi-open connection to its internal connection table and respond with a SYN-ACK. Since the attacker is not interested in completing the connection, he will discard that SYN-ACK and will not send that final ACK to establish the connection. The server will keep the state of this semi-open connection for a period of time (timeout). After opening a large number of semi-open connections, the server will no longer be able to establish new connections since the table would be full and legitimate users will not be able to access the victim's services.

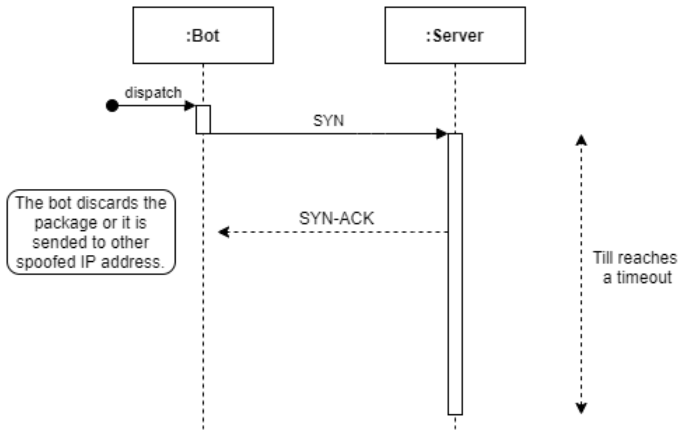

:Bot
:Server
dispatch
SYN
The bot discards the package or it is sended to other spoofed IP address.
SYN-ACK
Till reaches a timeout

FIGURE 3.6 Classic SYN flood. Sequence diagram of how a bot communicates with the server.

### Slow loris

These low bandwidth attacks try to drain the victim's resources by sending requests that never completes. After completing the connection using the TCP 3-way handshake, each bot sends small packets slowly towards the server to keep the connection open. Some servers, depending on how they are configured, close open connections if they do not receive any activity after a period of time from the client that requested the connection. Therefore, since the attacker is interested in keeping the server busy through those ports and keeping alive the connection, the bots will send requests slowly that will never complete.

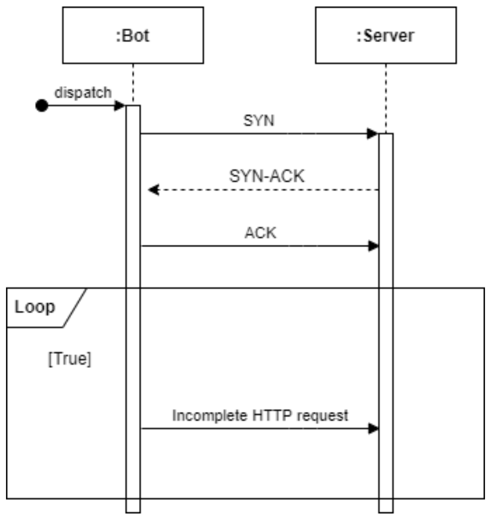


FIGURE 3.7 Slowloris. Sequence diagram of how a bot communicates with the server.

## 3.3.3 Application DDoS attacks

Application-level DDoS attacks are those attacks that attempt to exploit specific or inefficient weaknesses in the victim's application. These attacks follow the protocols used in the application so the traffic caused is legitimate from the server's perspective. These attacks are capable of imitating real actions made by users, which makes them more difficult to detect. Application functions such as logging in or searching are often vulnerable to these types of attacks due to the volume of data that is transmitted between the server and the client. For example, doing a simple search with a small packet message can return a long response from the server.

### Attack impact

These attacks can directly influence CPU performance and memory usage. This is achieved by targeting the resources that normally require them. Likewise, they can increase the use of databases servers since depending on the requests (such as starting a session or searches) they can access the stored data. The goal is not to get the real data of users, but to cause traffic and make the database server work until it reaches a point of stress. In addition, in an application there may be other storage types like log files or session data storage. So, an DDoS attack of this category could make work a lot these resources. For example, usually when a system is unstable or has failures, log files are required to troubleshoot these issues. Moreover, these files could be used specifically to detect possible DDoS incidents. Then these sheets could use more memory than expected, which can lead to a problem for the victim.

Application attacks are also effective in triggering performance degradation by exceeding some application limits. For example, many applications use multithreading to speed up some operations. If it is not well programmed, creating many thread workers can affect the performance and consumption of system resources.

### Effectiveness

As already mentioned, these attacks can simulate actions made by legitimate users, making them difficult to detect. Moreover, the deeper they affect the application the more unnoticed they are. For example, high CPU usage can be an obvious indication that something out of the ordinary is happening and can be an index of a DDoS attack. However, the same high usage in a database can be misdiagnosed as a performance problem or misconfiguration. For these type of detections it is possible that the victim needs extensive analysis and detection of anamolias so it could take "a long time"[3] to him to realize this.

These attacks stand out in that they require very few resources and needs little traffic to be sent. Although depending on the skills of the attacker, it is possible that only one computer is needed, that is, the attacker may be capable enough to launch an effective attack without the requirement of having a botnet. After all, as in the other types of attack, the attacker has the entire Internet to be able to search and use resources that are most convenient for him/her.

Also comment that normally companies work with dynamic methodologies, improving and creating new features constantly. However, each of these features can be a DDoS risk, since especially when they are launched into production for the first time they usually have vulnerabilities that attackers can explore and take advantage of.

### HTTP/HTTPS request flood

These attacks are based on the communication between the client and the server, where the first makes a request and the second responds with the data that has been requested or sends an error message. Like any application-level attack, there are many combinations and possibilities. The attack in general is based on sending protocol requests at a high enough rate so that the server is not able to respond to each of them. From here, it is up to the attacker's imagination and knowledge of the operation of these protocols to complicate or improve this attack. In addition, knowing weak points of the victim's application is a factor in favor of the attacker since he will know what he can focus on when developing his attack. The most common attacks of this type will be discussed below.

One of the most common functionalities offered by web pages is to give customers the opportunity to search. Depending on the implementation and need, it is likely that some of these searches require querying the databases or accessing APIs. Making HTTP requests that focus on searches can be very harmful for the victim because querying databases is not usually very desirable due to the workload suffered by the computers and the time it takes, especially when there is a large volume of data.

Another of the most common functionalities is the registration and login of users, especially the last one. This is a very important process for almost every type of application as it allows to identify or verify a valid user. These types of checks require explicit access to the database, which is where the user data is. We are talking about the same case as before, these are expensive operations performed by the server and the fact that it must manage many login requests at the same time could end up overwhelming it.

The attacker can also take advantage of the fact that although HTTP requests are very small in byte size, responses can be significantly larger. Therefore, the attacker can focus on specific resources of the application, in order to exhaust the output bandwidth of the victim (we have already seen how this can harm the target). Simple examples of this can be sample requests from catalogs or large pdf documents.

Less common but very vulnerable functionalities are those that allow users to upload files to their platform. Allowing content to upload to those pages makes it easier for the attacker to send large traffic legitimately or at least that look legitimate. In this way the victim's network is saturated and the storage space is exhausted with junk data. This also applies to those websites that allow forms, since in addition to the form itself that already has its size, many have the option of adding files. A very clear example of this are the typical work forms where it also offers you the possibility of attaching your curriculum vitae or cover letter.

[3] the time is relative for each company. Some companies might get more harmed than others in the same amount of time.

### NTP amplification

These are based on reflected volumetric attacks in which the attacker exploits the functionalities of the NTP protocols with the intention of overwhelming the victim's network with a large volume of UDP packets. This, like any other amplification attack, takes advantage of Internet resources to increase the traffic to be sent to the target. The way in which it achieves this is by sending UDP packets with a specific message to different NTP servers that will read the request and will produce a result that will be sent to the IP address of the one that have made the request. That is why it is required to spoof the source IP address with the target's IP address, so that all those results reach the victim's network even if it was not the one who made the request. Figure 3.8 shows in a simplified way how this attack works where a single bot sends requests to NTP servers, and these produce a result (this is much larger than the request made by the attacker) which is sent to the victim's network.

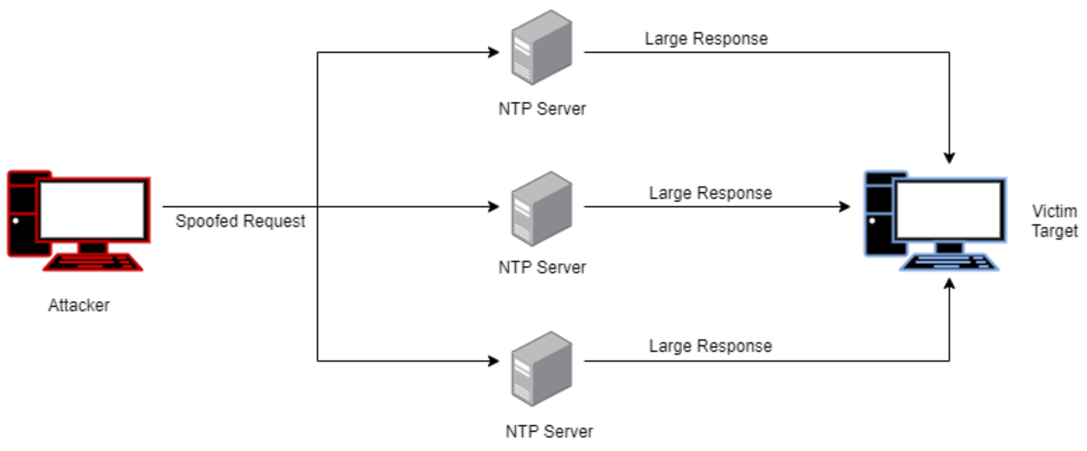


FIGURE 3.8 NTP amplification (D)DoS attack. Represents how a single bot perform the attack.

# Chapter 4

# DDoS Hive Comb

So far we have been able to see what the (D)DoS attacks are and the damage that their impact produces on a target. To contrast this, the victim need to be prepared for possible future attacks or must act at the time the attack occurs. Much will depend on the capabilities of the attacker and the mitigation techniques used by the victim. In this paper we will not focus on mitigation techniques but rather we will offer a stress service to help companies or any individual who wants to check the mitigation capabilities of their network infrastructure. Next we will see how these **IP stressers** work and a detailled explanation on how it was implemented.

## 4.1 IP stresser

IP stressers are tools designed to test the robustness of a server or network infrastructure. The person in charge of doing the test, whom we will refer to as an administrator, will run a test plan to see if different resources are capable of handling an additional load. These services, unlike a DDoS attack, are legal since the target gives their consent to be tested in a situation similar to a real attack. With this, the contractor can know to what extent his services are capable of performing without suffering damage, and if they would be able to defend theirselves against attacks of a similar magnitude[1]. The providers of these services usually have their own ''botnet'', which consists of having their own or rented computers explicitly configured to be able to listen to and carry out the commands sent by the administrator. Having these services is not usually cheap and that is why stress services need a source of income to be able to expand or maintain their own services. To do this, many offer a payment plan to use their services over a period of time.

These should not be confused with **booters**, which are on-demand DDoS attacks services offered mostly by cybercriminals that have the purpose of taking down websites and networks in exchange for a monetary cost. These unlike the previous ones are illegal. Although their services may seem very similar and offer very similar cost plans, booters providers are not interested in being discovered. For this reason, they usually mask themselves using proxy services so as not to leave a trace of their activities on the network. Likewise, their accepted forms of payment are usually with cryptocurrencies, paypal or any other way that does not leave a mark.

In our case we offer an IP stresser which will be totally free but limited for anyone who wants to test their website or network. The intention of DDoS Hive Comb is to create a community where together we can confront cybercriminals and thus be able to help each other to fight against the maximum possible cases of DDoS attacks. As there will not be a source of income and the computer equipment is not already available to create a botnet to launch the commands, a small script will be implemented on the computer of each user who wants to participate in the project. This script will be a downloadable file as long as the user gives his consent, accepts the rules of conduct and understands the role he/she forms. With this script downloaded and executed on the user's computer, it will be able to communicate with the administrator's computer, which will be responsible for the execution of all the tests. Next, it will be explained in much more detail how the project was created and how it is structured.

[1] Some providers claim that they offer 5GB or 10GB services, but in reality that is their maximum capacity that could occur if they only focus on a single target at a time.

## 4.2 Target audience

The target audience of our website can be divided into three person prototypes. For the project, user personas have not been created for a matter of time, but the potential market has been investigated and analyzed to identify different public contexts that we think would be the ideal clients for our website.

First, we find companies of any size and that offer services or products through the use of the Internet. As we have seen, they are very susceptible to these DDoS attacks and must be prepared to deal with them. Typically, the larger the company is, the more options they have to suffer these attacks. In addition, we have IT costumers, who are people interested in the IT sector and want to further deepen their knowledge by doing different tests on their personal network or understanding the operation of different DDoS attacks. These users could buy or contract booters, but remember that these are illegal, therefore a service like ours could help them personally to improve their capabilities. And finally, educational centers or other organizations that do not precisely use the Internet to sell products, are potential customers seeking to protect their network infrastructure from possible DDoS attacks.

## 4.3 User roles

Coming up next, we will explain the type of users that can be found in the project and thus the role they play. Each one of them have their own responsabilities and permissions to interact with the website. We only have to distinguish between 3 of them, which are: administrator, bee user and visitor user.

> ***Note:*** *during the project if an user has joined the community we will say that he/she is part of the* ***hive*** *.*

The **administrator**, also called as the **queen bee** or **hive queen**, has access to all the functionalities of the website and is responsible for executing the tests. Like any other user who is part of the hive, the queen bee can use its own computer to launch the commands and not just send the orders that the other users of the hive must execute. At the moment, there will only be a single administrator in the entire project.

The **bee user**, or also called the **hive user** or simply **user**, is that one who has already agreed to join the hive. None of these users have more privileges than others since all must respect a standard of conduct that will allow no malicious actions that could harm the hive. User bees will not be able to access all the functionalities of the website such as the queen bee, which is why, at least for the moment, the user will have to accept that the queen bee is the one in charge of carrying out the tests. Likewise, they will have permission to download the script that will be used to communicate their computer with the administrator's machine.

The **visiting user**, or also called as **anonymous user**, will be the one who visits the web page but is not yet part of the hive. The anonymous user will not have permission for anything other than reading how the project works or the rules of conduct, in addition to accessing the hive if interested.

## 4.4 Technology stack

Next, the use of the set of technologies, software and tools used for the development of the project will be discussed and justified. In general, I want to say that for the choice of this stack I have relied a little bit on my own experience since I wanted to use already known tech stack so as not to invest too much time in learning new technologies I do not know about, among other reasons that will be discussed later with more detail. Even so, I have used some new technologies that I wanted to try before but I never had the opportunity to do it. Therefore, I have used this same project as a motivation to learn the use of those and thus expand my personal stack report.

To begin with, the operating system with which I have decided to work is **Windows**. Although I have Linux and I like to work with it, I preferred to choose Windows because of its simplicity and because I already had some software installed that I was going to use during the project. Also, I have some issues with package dependency in Linux OS that I have been dragging around for a while. I could also have used a virtual machine, but the problem is that they slow down the computer a lot and it is difficult to work with them for this reason. So I finally decided not to think about it any more and to work with Windows in a virtual environment so as not to have the same problems that I have with Linux.

Next, the decision of the frameworks used in the backend and in the frontend follows. For the first one, I have chosen to use **Django**, one of the most popular backend frameworks nowadays. Until that day, I had experienced working only with Flask and Django, and I knew the names of others due to their popularity and reputation such as Laravel or Express.js. However, for personal tastes I preferred to choose a framework that works in Python language because I am better at it. As for why Django and not another framework that uses Python language is because I have had a good experience with it in previous projects, especially with its documentation. The fact that Django offers such clear and robust documentation has made me opt for it and make me interested in knowing its full potential. In addition, it is highly scalable and customizable, which in my opinion satisfies the idea of the application that we want to accomplish in this project.

On the other side, for the frontend framework I had initially decided to use React.js. However, I finally opted not to use it as although I started working with it, I did not have a good experience mainly because I found its documentation to be somewhat poor. To be honest, I must admit that I do not like working with a frontend framework in general, as for me, it increases complexity and limits creativity when developing functionalities compared to doing it from scratch. However, I understand that using these helps the developer create a better user experience (UX) for clients, so I am aware that I can not stick with the idea of never using a frontend framework. I just think that I did not start off on the right foot, and decided to refrain from using it as I was wasting a lot of time with things that at first glance are simple. I am sure I will give it another chance at another personal job, but this time I was more aware of learning web protocols and having the application made for the deadline.

For the database, I have decided to use a SQL database to store information related to users such as store each bee user who joins in the hive. My tutor advised me to use MongoDB (noSQL database), but I rejected the proposal and decided to use **PostgreSQL**. The reason I did not accept was because I had zero knowledge about noSQL databases and at the same time my idea was to save little data. Therefore, I thought that using a relational database was more than enough to fulfill my ideas. At first I had not understood very well why I was recommended to use MongoDB, but after a research I understood that these document-oriented databases are highly scalable and offer a performance improvement for reading data since they have a less complicated structured. By less complicated structured, I mean that a document database has all the information needed in one place, so the database just has to retrieve all the information from one location. Still, I don't think I made a bad choice since in my case I wanted to save structured data, which requires a SQL database, while MongoDB works with unstructured data. Then within all the relational database options, I have chosen PostgreSQL for its simplicity of work and for its good performance even under stress compared for example with MySQL. However, throughout this semester I have been exploring more technologies that I could integrate into the project, and I decided that I wanted to use a time series database called InfluxDB to monitor my website traffic. The time series are designed to save data over time in a very optimal way in pairs of time and value. So I considered it interesting to add it to my project, but as I was short of time, I did not finish implementing it.

To be able to check the functionalities of my web page that consist of sending and capturing packets, I have used the protocol analyzer called **Wireshark**, which helped me a lot throughout the entire project. Probably one of the tools that has been most useful to me since it allowed me to understand the format of the packets and the communication flow between the tested ports. I found many errors and bugs in my project thanks to this tool, especially when capturing the packets that I was sending because I was not filtering quite well the packets and I had some problems to extracting some information from these packets.

**Github** has been used to locate the project and **git** commands to work with different branches during the project implementation period. In the repository where the application is located, it can be found various releases that were made during that time. The branch system that I used was mainly one branch per task.

For testing the application I have used **pytest** tool, which has allowed me to create different unit tests. Likewise, this tool has been used to calculate the total coverage of the application. For the continuous integration of the application, a repository has been cloned from Github in a virtual environment with **Travis CI**.

## 4.5 Planification

Next, we will see how the project has been planned during the development period. For this, the Agile methodology has been followed with some personal adjustments since this work has been done individually.

To begin with, various **user stories** have been created to be able to specify the behavior that users will have and also to be oriented when designing the web page. Moreover, for each of the stories a scoring system has been used to assess the difficulty and priority of these when implementing them in the application. The scoring system is called **Story Points** and it consists of expressing an estimate of the total effort that should be made to implement an element of the product backlog integrally. Likewise, Story Points will be assigned according to the complexity and volume of the work, as well as the risk or uncertainty. Thus, it will be possible to be more aware of what can be achieved in a specific period of time and generate a solution plan as the stories are completed or, on the contrary, unforeseen or difficulties arise. However, comment that the estimate could be inaccurate in some cases since only my point of view would be seen. That is, since there is no team with which to estimate work on a specific scale, there are no different opinions about a specific story and it becomes more difficult to estimate accurately.

To carry out this estimation of the stories complexity, I have used the Fibonacci sequence as a scale, starting from 1 to 13. On this scale, 1 represents that there is little complexity, and 13 will mark the maximum complexity. The intermediate numbers, to the extent that the number is greater, will represent a greater complexity. On the other hand, to estimate the priority of each story, the same scale has been used but with the inverse logic. So in this case, 1 represents the highest priority and 13 the lowest priority. For intermediate numbers, the higher the number, the less priority will be given to that story.

Furthermore, I have used a tagging system to classify user stories because it made it easier for me to understand the relationship that a particular story has with others. Above all, it has been useful to me when a group of stories are related to each other and there may be a dependency between them making one not possible to implement without the other. This should not be given if the estimates are made correctly, since in itself the stories that are essential to do the others should have higher priority. However, as I have already mentioned, not having many points of view made this part very complicated for me and I had to find a solution to counteract this part. The tagging system that I have used to divide the user stories is divided into 15 categories.

### 4.5.1 User Stories

Next, all the user stories for each of these categories will be exposed along with the scoring system explained above. Moreover, a "Completed" column is added to see which one of these stories has been completed during the project. Mention that we refer to the web developer as the sole responsible for doing the project, as there is no QA team or DevOps team to divide the work. So even if we refer to the web developer to perform a specific action, we mean that it acts as QA and also as DevOps at the same time.

| Category | User Story | Story Points (complexity) | Priority | Completed |
|---|---|---|---|---|
| DB | As a web developer, I want to specify the relationships of my entities so that the database knows how my models are related. | 2 | 1 | ✔ |

| Category | User Story | Story Points (complexity) | Priority | Completed |
|---|---|---|---|---|
| DB | As a web developer, I want to install a backend framework in order to build the backend structure of my website. | 1 | 1 | ✓ |
| DB | As a web developer, I want a database to store my project's data. | 2 | 1 | ✓ |
| DB | As a web developer, I want to install influxDB in order to monitor the traffic of my website. | 1 | 8 | ✓ |
| DB | As a web developer I want to collect data performance of the website and show it in the website in order to visualize and analyze if the website is under stress. | 8 | 8 | ✓ |
| WEB | As a user, I want a navigation bar to help me move more comfortably through the website. | 5 | 3 | ✓ |
| WEB | As a user, I want to see a footer at the bottom of the page to know that I have reached the end of the page. | 2 | 5 | ✓ |
| WEB | As a web developer, I want to install React in order to use this framework to build my frontend app. | 2 | 2 | ✓ |
| HIVE | As a user, I want to join in this project to be part of the hive. | 5 | 2 | ✓ |
| HIVE | As a user, I want to see how many bees this page has under its domain to do a DDoS Attack. | 3 | 5 | ✓ |

| Category | User Story | Story Points (complexity) | Priority | Completed |
|---|---|---|---|---|
| HIVE | As a user, I don't want to continue being part of this project because I have changed my mind. | 5 | 3 | ✓ |
| HIVE | As a user, I want to understand how things work in order to be aware of how I contribute to the project. | 2 | 8 | ✓ |
| T&C | As a user, I want to know what is being done with my data so that I can be sure that there is no problem with it. | 2 | 8 | ✓ |
| NOTIF | As a new user, I want to be registered in order to get notified when an attack starts. | 2 | 8 | |
| NOTIF | As a user, I want to be notified when an attack is performed to see how the page performs. | 2 | 8 | |
| DEPLOY | As a web developer, I want to deploy my website to be reached by other users. | 5 | 2 | ✓ |
| DEPLOY | As a web developer, I want to install apache and configure with wsgi DJANGO in order to deploy my app. | 5 | 3 | |
| ABOUT | As a user, I want to know more about this project to know what I am involved in. | 5 | 13 | |
| TEST | As a web developer, I want to research the best technologies that fit on my page so that I can test my application. | 3 | 2 | ✓ |

| Category | User Story | Story Points (complexity) | Priority | Completed |
|---|---|---|---|---|
| TEST | As a web developer, I want to test the compatibility of the page in different browsers to see if it works properly. | 5 | 5 | |
| TEST | As a web developer, I want to comprove if the app works properly in a mobile phone to see if there are any bugs or errors to be fixed. | 5 | 8 | |
| TEST | As a web developer, I want to comprove if the app is smoothly enough and the interactions of the useres are performed as expected in order to check if it is needed to do somo changes. | 3 | 5 | |
| TEST | As a web developer, I want to custom CI workflow with Django to automatize testing. | 3 | 5 | ✓ |
| CRED | As a web developer, I want to improve the credentials page in order to make it more secure for the user and give a better experience for the user | 5 | 3 | ✓ |
| CRED | As an admin, I want to insert my credentials to have access to the attack page. | 3 | 2 | ✓ |
| ATTACK | As an admin, I want to be able to select the type of attack to be executed in order to have the control of the actions. | 3 | 3 | ✓ |
| ATTACK | As the admin, I want to do a ping flood attack on my page to check that I am capable to send ICMP packets. | 8 | 3 | ✓ |

| Category | User Story | Story Points (complexity) | Priority | Completed |
|---|---|---|---|---|
| ATTACK | As the admin, I want to do a SYN flood attack on my page to check that I am capable to send TCP packets. | 8 | 3 | ✓ |
| ATTACK | As the admin, I want to do a UDP flood attack on my page to check that I am capable to send UDP packets. | 8 | 3 | ✓ |
| ATTACK | As the admin, I want to do a HTTP flood attack on my page to check that I am capable to send HTTP packets. | 13 | 3 | ✓ |
| ATTACK | As the admin, I want to do a NTP amplification attack on my page to check that I am capable of perform this attack. | 13 | 5 | ✓ |
| ATTACK | As the admin, I want to do a slowloris attack on my page to check that I am capable of perform this attack. | 13 | 5 | ✓ |
| ATTACK | As the admin, I want to see how each attack work and make a test of it before performing the actual attack. | 5 | 8 | ✓ |
| ATTACK | As the admin, I want to perform a DDoS attack using all the bots available of the botnet in order to put a server under stress. | 13 | 3 | ✓ |
| SNIFFER | As the web developer, I want to track the traffic of my website to see which IP's are communicated with me. | 5 | 3 | ✓ |
| DEFENSE | As the admin, I want this page to be secure so it can handle Ping Flood attack. | 8 | 8 | [Cancelled] |

| Category | User Story | Story Points (complexity) | Priority | Completed |
|---|---|---|---|---|
| DEFENSE | As the admin, I want this page to be secure so it can handle SYN Flood attack. | 8 | 8 | [Cancelled] |
| DEFENSE | As the admin, I want this page to be secure so it can handle UDP Flood attack. | 8 | 8 | [Cancelled] |
| BOTNET | As a web developer, I want to know how I can create a botnet in order to make my own botnet. | 3 | 3 | ✓ |
| BOTNET | As a web developer, I want to create a communication system between a client and a server in order that the server can send commands to the client ant this must perform the command in his terminal. | 13 | 3 | ✓ |
| BOTNET | As a developer, I want to configure the command and control in order to apply commands for only dos attacks. | 13 | 3 | ✓ |
| BOTNET | As the user, I want to download the script to make part of the botnet and participate in the attacks. | 3 | 3 | ✓ |
| RESEARCH | As the web developer, I want to understand better how Volumetric DDoS attacks work in order to apply changes to the project. | 5 | 2 | ✓ |
| RESEARCH | As the web developer, I want to understand better how Protocol DDoS attacks work in order to apply changes to the project. | 5 | 2 | ✓ |

| Category | User Story | Story Points (complexity) | Priority | Completed |
|---|---|---|---|---|
| RESEARCH | As the web developer, I want to understand better how App-level DDoS attacks work in order to apply changes to the project. | 5 | 2 | ✓ |
| RESEARCH | As the web developer, I want to understand better how Reflection/Amplification DDoS attacks work in order to apply changes to the project. | 5 | 2 | ✓ |
| RESEARCH | As a web developer, I want to improve my skills and knowledge about the async methods of python in order to know how to use it with Django | 3 | 5 | ✓ |
| RESEARCH | As a web developer, I want to investigate the difference between asgi and wsgi in order to know what fits better for the project deployment. | 3 | 5 | ✓ |
| RESEARCH | As a web developer, I want to investigate different technologies and tools which can be beneficial for the project. | 3 | 8 | ✓ |
| SECURITY | As a web developer, I want to get Django Doctor to improve my webiste security. | 2 | 8 | |

**Table 4.1** Table with user stories grouped into categories. There is no special order, being the first does not mean that the first is more important than the others.

### 4.5.2 Work distribution

Once the user stories have been created and the scoring system has been assigned, the question arises as to what criteria we follow to choose the user stories. The doubt falls in which more importance is given, if to the priority or to the complexity, or perhaps to make an average between the two. In this case, as I was not very sure that my estimate was the most accurate, I wanted to find a way to be sure that I could go at a good pace to be able to make all those stories or at least most of them and not have to lose much time in each sprint which I will choose to implement. Regarding sprints, an agreement has been reached with my tutor to carry out one-week sprints where an implementation of a subset will have to be made. It has been set as a goal to do a total of 12 sprints, therefore, it would be necessary to distribute all that work during these 12 sprints. With a margin of 1 or 2 more weeks to complete some other tasks if it was necessary.

To do so, all the story points (SP) have been added to calculate the average that should be done per week or sprint. The total number of SP has been 233, and when dividing it by the total number of sprints, we have an average of 19.42 recommended story points to be done per week to be able to complete them all. Then, the highest priority stories were chosen until a number around the average of SP per sprint was reached. In the case that a story had a higher priority than others, but its SP is too high to catch it in that week, a story of lower priority and also with less SP was chosen to be able to advance on the fly. For example, if we already had a total of 15 SP accumulated for that sprint with the highest priority stories available, then if the next one already had a SP of 13, we would miss the recommended estimation per sprint (28 SP accumulated > 19.42 SP recommended). So we will choose a story that had for example 5 SP and lower priority. This way we would have a total of 20 SP to do in that sprint. Another criteria that was taken into account is this label system that we have discussed to classify the US into different categories. That is why, sometimes we tried to choose stories from the same category even if it had less priority, since being related to the others that were being carried out in that sprint we could have something functional to present in the realese, and not just a piece that did not contribute much by itself.

In short, we can define the following steps to distribute the user stories during the sprints.

1. Get the total number of story points and sprints that will be done during the project.
2. Make an average estimation of how many SP will be done per sprint.
3. Choose the user stories that will be done in a sprint taking into account that the sum of their SP is close to the value obtained in the average estimation.
    - Criteria 1: Take those that have the highest priority.
    - Criteria 2: If the total number of SP exceeds the average estimation, choose a lower priority US so that the total sum of SP for that sprint matches the estimate.
    - Criteria 3: Try to choose a US that is from the same category as the other ones to have a functional subset to present on the day of the sprint realese.

The advantages of using this strategy is that we distribute the workload equally for each sprint. Moreover, it is possible to do a deployment where users can already make interactions with the web page even if it is not yet complete, in this way they can be more interested since they would see updates in each sprint where they can perform actions than in the previous one they could not. This would generate a greater engagement to users with the intention to not lose them and at the same time, to show that we are a group that works dynamically. In this way, we also make sure that we will do the majority of the US that we have planned, since a lot of perseverance will be required to achieve the goal set. Also, we managed to counteract the bad estimation of some US for the reasons that have been explained above.

Obviously, all these benefits would occur in the ideal case where I would be only focused on this project and there were no problems when implementing the US, which would cause a delay that would be somewhat difficult to recover from. Therefore, this strategy also has drawbacks in which one of them, as we have already mentioned, is that a lot of perseverance is required to reach the goal set. This is why this becomes a double-edged sword, since I am not only focused on this project so I will not have the same availability for each sprint.

To see a more detailed view of the user stories and how I distributed them in the different sprints, check the following link. In this excel you can find in the "Product Backlog" tab, not only the aforementioned US, but also the acceptance criteria of each one and also the tasks that needed to be carried out in order to complete them. Then in the "Planification" there is a graph showing how many SP were finished and what was expected to finish until a respective sprint. Finally, we have the "Sprint X" where we can find additional information for each sprint over the US and the tasks that have been performed, the number of hours that were spent on each task, a graphical view of how I worked during each day of each sprint and much more information.

## 4.6 Database storage

Now we will focus on the data that we have stored for the use of this application. As already mentioned, for the application we have used a PostgreSQL relational database and I wanted to use the Time series InfluxDB database. Even if I did not implemented at the end, I want to explain later my idea in this section.

Firstly, in figure 4.1 we can see the sets that we will store in our relational database and how are they related. Starting with the User, these will be all those who join the Hive in order to register their IP address and their role, which at the moment the options are only User bee or Administrator. Furthermore, we optionally store their email to notify the user before an attack or for possible future implementations. If the user has the role of user bee, for the moment no more information will be necessary about him since with his IP address we would have enough. His IP address will be useful to us to find and communicate with his device that will be linked to our Botnet. On the other hand, if the user's role is administrator, then we will need to store extra information since to access the operations page where the attacks will take place, it will be required to enter some credentials in addition to other security measures that will be explained later. For credentials, an administrator must enter their username and password. We add a salt to the password that contains a series of random characters before the password is hashed. This is intended to better protect administrators' privacy and mitigate hash table attacks by forcing attackers to recalculate using the salts for each admin. Moreover, we will also have a field that stores the number of times the admin has tried to log in with the account marked by its IP address. That is, for a specific IP address we will see if the user has entered the correct credentials, otherwise one unit will be added to the attempts field. When this field reaches the value of 3, the admin account will be locked for a certain time. The missing fields last_time_logged and cookie_session_device are for other security measures, which will be seen later. Therefore, those admin accounts that have been blocked for exceeding the total number of attempts to log in or for other reasons also linked to the behavior of the admin, will be stored in AdminBlocked. In this table we store the time that marks until when the account is invalid, so that until that time passes the admin will not be able to access again by entering their credentials. However, we will also have a table for those users who are no longer part of the Hive. That is, when a user leaves the Hive, we delete everything we have stored about him in the User table and we store his IP address with a timeout so that he can rejoin. This is a security measure, so that the user does not exploit the functionality of creating and deleting a user from the database.

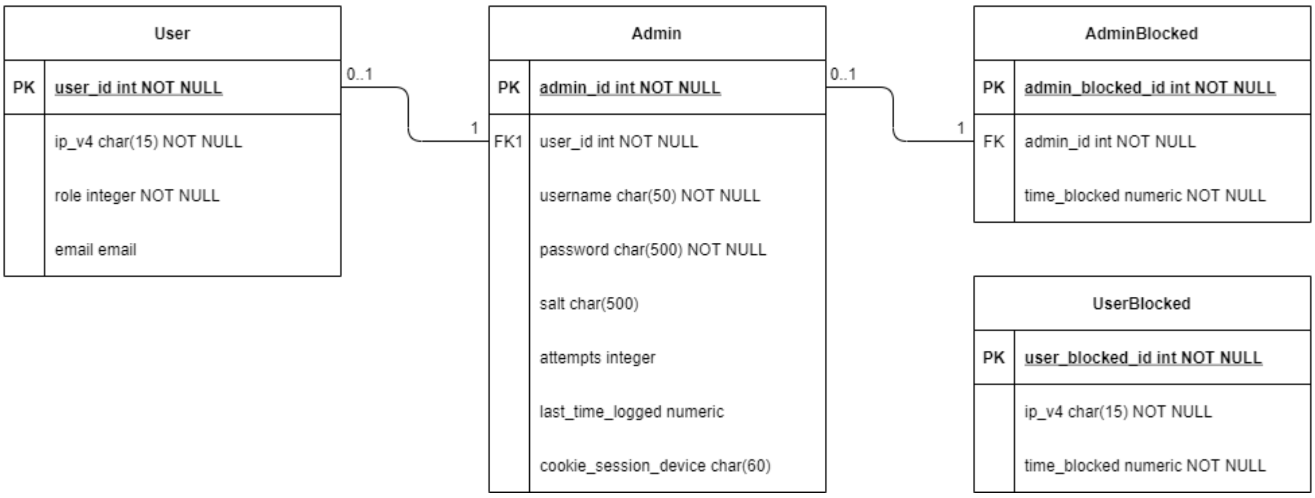


**FIGURE 4.1** Entity relationship diagram of the sets that are stored in the database.

Now, as for the Time Series, it has been intentented to be used to monitor the home page and thus see the traffic caused. The intention was to collect the data regardless of the origin of the packets that get into the home page. First of all, explain a bit in a general way that a Time Series is structured in a database that can contain various measurements. Each of these measurements contains a series of points which usually contain quantitative information, and also some tags which are used to store optionally metadata and qualitative information. A point is made up of a mandatory field called time, which is what is used to identify a specific point in a measurement. In addition to that field, we can add as many fields as we want with information that we find most useful. In our case, for one point we would have stored the packets information filtered by our sniffer. This, could have been valuable for us since we could identify anomalies in the traffic. For the visualization of this data we could have used a platform like Grafana which allows queries to the database and offers other tools to make a better analysis. Moreover, it allows to create an alert, which in our case would be useful when a certain number of packets exceeds a limit or, on the contrary, when it is below a limit. In this way, it will not be necessary for us to always be aware of what happens with this data and we save a lot of time on checking it every time.

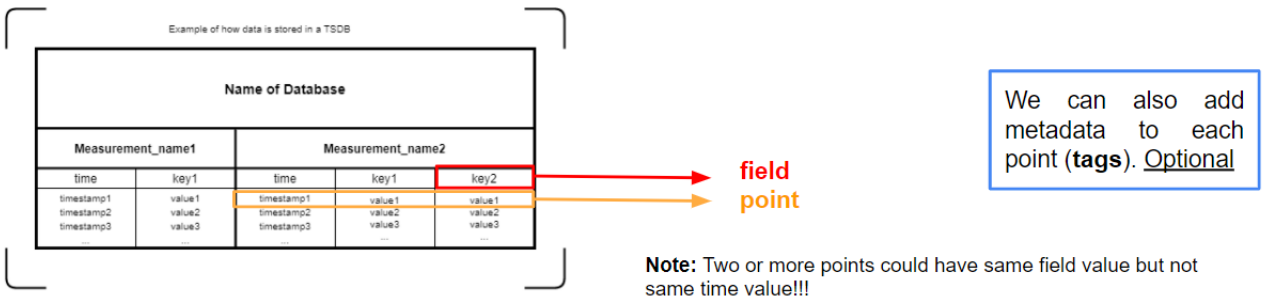


Figure 4.2 This is just a simplified version of the time series database. It is just an approach to show in a more basic way how the different elements are grouped.

## 4.7 User experience design

Although this project is based more on the development and research of DDoS attacks, we have tried to look for the best user experience (UX) to provide the best possible product. Thus, in the first place, the information architecture was thought, which includes the design of the navigation system and the tagging system to get a general idea of how the web page would be structured. Afterwards, usually before starting to implement the web page, different prototypes would be made with the intention of seeking feedback and thinking about the smallest details. However, for this project feedback from the prototypes was not sought, since even though it is an important step, it would involve investing a lot of time. So, we will go at a faster pace and go straight to the final hi-fi prototype.

> ***Note:*** *remember that administrators can do the same actions as users, and only they have access to restricted pages. Therefore, when we refer to users, although in the project we have defined that a user is another nomenclature for user bee, everything we refer to that they can do also indicates that administrators are capable of doing it. When, only the admins can do something in particular, this name will be specifically specified.*

### 4.7.1 Information architecture

The objective of the navigation system is to find the easiest and most optimal model for the user's movement in the informational space. It has been considered to make a mixed navigation, so that from the main screen to the subsections a hierarchical navigation combined with a hypertextual one will be used. On all pages there will be a navigation bar that will allow you to navigate to the main pages: home, attack, defense, analysis, about and settings. These pages can be accessed from any other page. Note that the attack and defense pages will be seen only if the user's IP address matches one of the administrators. In this way, we deny users permission to access these pages completely. In figure 4.3 the connection between these main pages is shown. Notice that a barrier is displayed before accessing the attack and defense pages called credentials. This barrier, although we will explain it in more detail in the next section, is to show that these two pages (attack and defense) are also connected to the others, but it will be necessary for the admin to enter their access data before.

From the home page, as shown in figure 4.4, users can access an information section where we will have everything necessary so that users can understand how our page works and another section of cookies, so that users can give their consent to different cookies that we will insert on their device. In addition, they will also have the possibility to join the Hive from that same page where they will have to read a code of conduct and accept it before continuing. On the other hand, if the user has already joined and wants to leave the Hive, they will be informed of what will happen to their account before continuing.

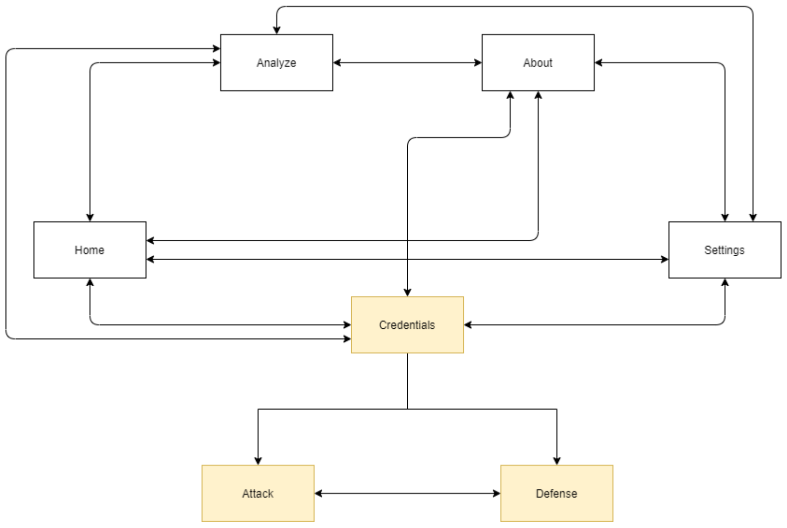


Figure 4.3 All main pages that can be accessed from any other web page. This can also be represented as a complete graph, since all nodes are strongly connected. In yellow, the pages that can only be accessed by administrators. Note that from the attack and defense pages we can also go to the other 4 pages, but we do not draw more arrows because it makes reading difficult.

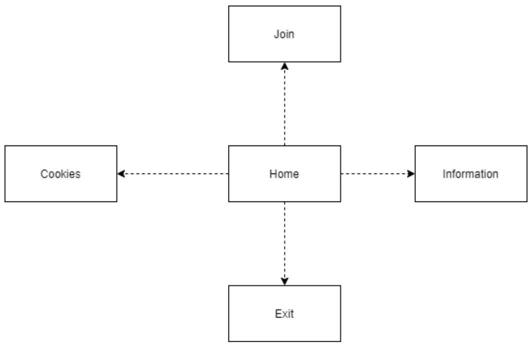


Figure 4.4 Possible navigation from the main page. The dashed arrows indicates that all the actions happens in the same page, so all these navegation options will be shown as modals (pop ups).

For the attack page, as the figure 4.5 shows, an administrator may come across different attack options whose names should already be familiar to us. The admin will have access to a test page for each attack where they can access a page to test the attack and see how it works or access to the actual DDoS attack page where the specific attack can be performed. The menu is simple, from the attack page the admin can access any of these attacks either for the test or to perform the attack, and from these pages they can only go back to the menu page.

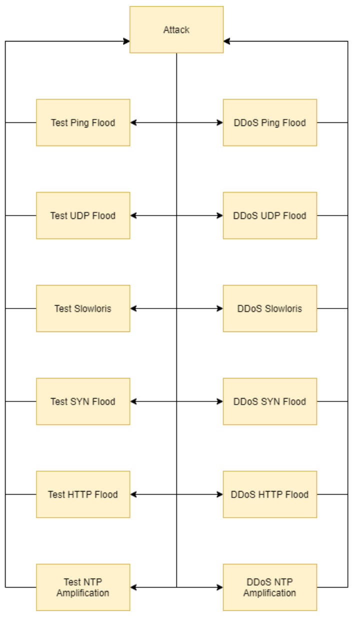


Figure 4.5 Possible navigation from the attack page. From each of these pages, we can access to the others 5 main pages explained in Figure 4.3. The yellow indicates that only admins can access these pages.

For the other main pages, there are no more navigation options, since all the actions will be carried out in their respective pages without the need to create new web pages or modals to see extra information.

Regarding the label system, each section of the information has been named using the hybrid card-sorting method. For this, I have had the help of only 3 participants that, although they only meet a target audience profile (IT costumer), I think it has been beneficial enough to take different points of view. This technique has been used in order to know how to organize the different elements between the 6 main pages that we discussed earlier (close sorting). Before the participants proceeded to order them, they were put into context the different types of users that the web page allows and that the attack and defense pages are only accessible to administrators. Within each category, the participants have been given freedom to classify how they would like to find the elements best (open sorting). I have also answered some questions that came up at the time in order to help them to understand some topics related to DDoS attack types.

After doing a pooling, the result that can be seen in figure 4.6 has been obtained. For this activity, we have fixed some categories which represents the different main pages that we explained in the navigation system. The participants were offered different cards (the yellow cards) which they had to classify within the fixed categories. However, they could create a subset of categories within them if they thought these subgroups would make more sense. In this way, in the attack and defense categories we can see that we have different groupings by subcategories.

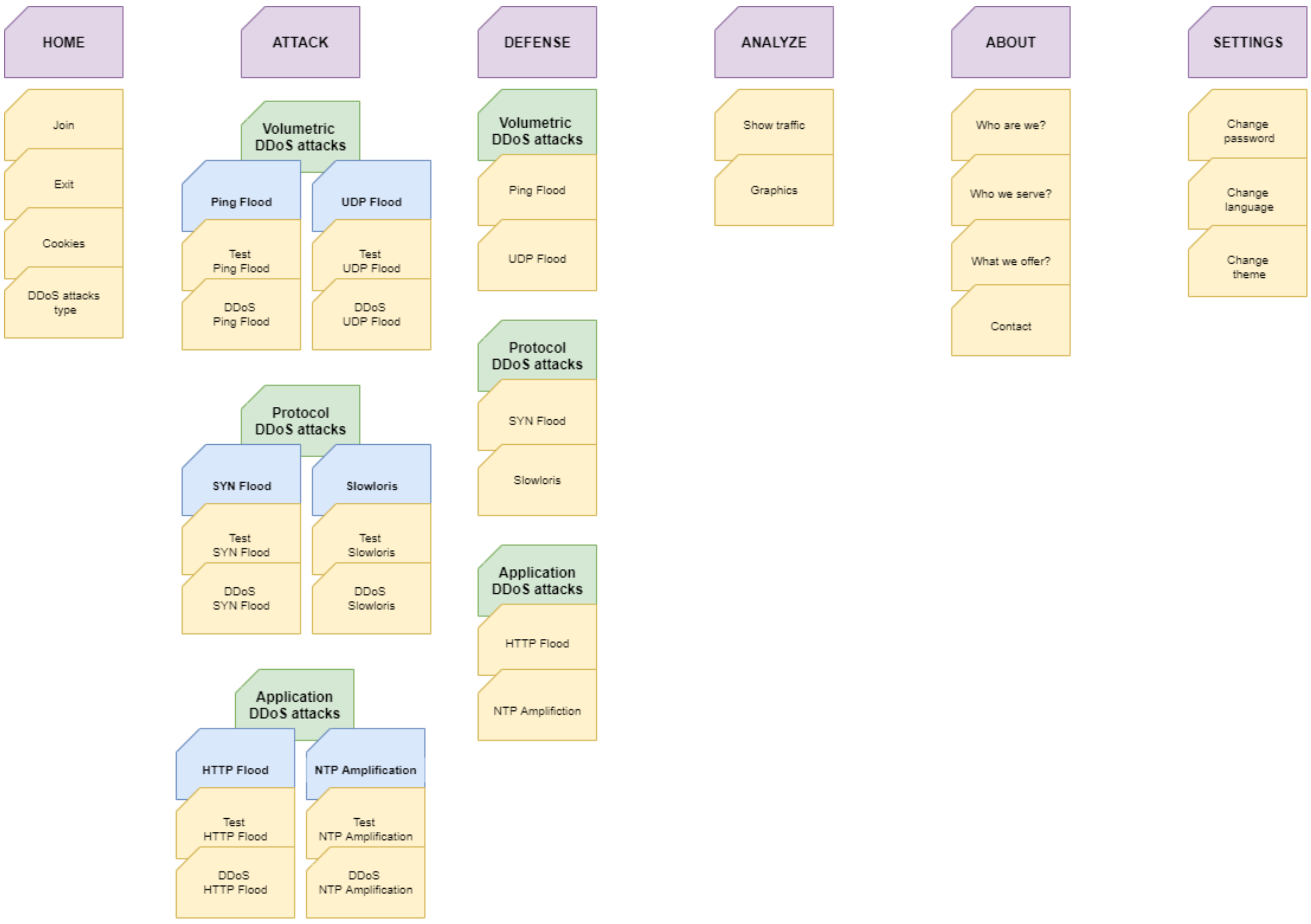
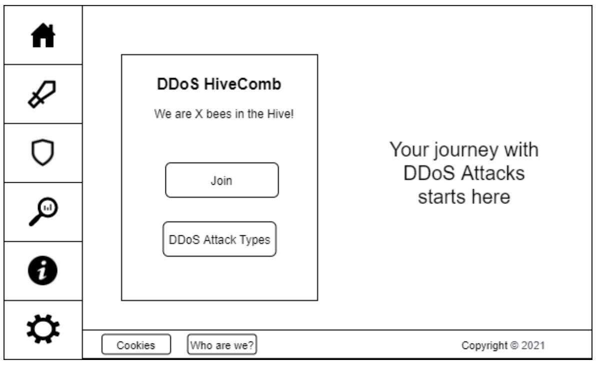


FIGURE 4.6 Hybrid card sorting method. In purple, the fixed categories that represents each of the main pages. In yellow, the cards that have been given to the participants to classify. In blue and green, subcategories that the participants considered more suitable for a better understanding.

### 4.7.2 Prototype

As we have already mentioned, conceptual prototypes have not been created since it would take too much time and I would not be able to do other tasks. For this reason, a single prototype has been created and that idea has been worked on. However, it can be seen that there were finally changes with respect to the prototype that we will show now, due to momentary feedback (at the time of development) from different people who have helped me to make the page more user-friendly. For this prototype, the navigation system and the label system already explained have been taken into account.

To begin with, we find the home page which will have two different views depending on whether the user has joined the hive or not. First of all, highlight the vertical navigation bar which will be found on all pages. In this way, with the vertical menu we can in a great way add an unexpected touch to the design of our website and give it a different style to what our competitors offer. Moreover, because of the way the user is going to have to navigate through our website, they will find navigation much easier since, as already seen, everything is focused on the 6 main pages. We have used icons instead of text to make the page more interesting and attractive to read. Plus, icons can communicate an idea in seconds and they can break the language barrier. In figure 4.7 it can be seen that visibility is given great importance, where all the elements are easy to read and identify. There are also not many distractions, and the main actions that the user can take on each of those pages can be easily identified.


DDoS HiveComb
We are X bees in the Hive!
Join
DDoS Attack Types
Your journey with DDoS Attacks starts here
Cookies
Who are we?
Copyright © 2021

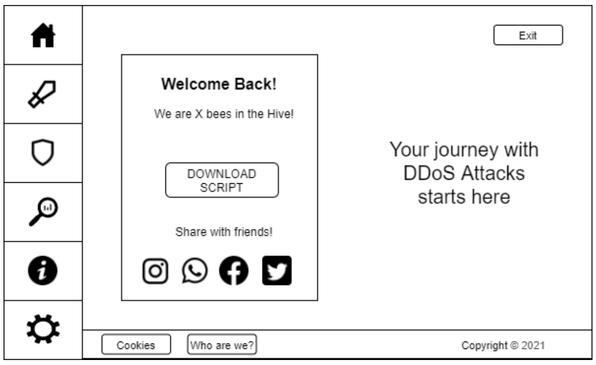


Figure 4.7 Home page. On the left you can see the screen in case the user is not registered. On the right when the user is registered.

Next, we find the credentials page which will be the page that after introducing the correct data will give to the admins the access to the attack or defense page. This will be a simple page where only the authorization of the administrator will be sought as shown in figure 4.8.

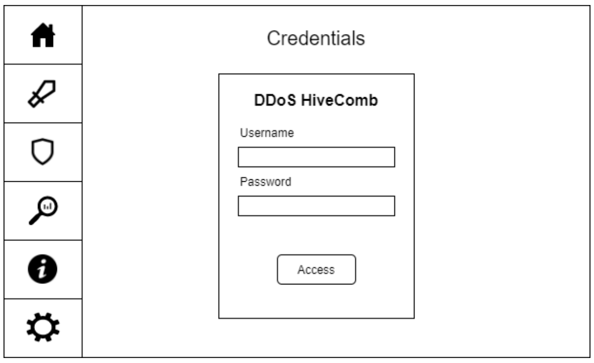


Figure 4.8 Credentials page. This page will be displayed before accessing the attack and defense pages.

When the attack page is accessed, the first thing that we will find is a main menu where the user can find the different attacks as it has been organized with the previously explained tagging system. For each attack we can see that there will be a small description of it and two buttons which will take the administrator to their respective pages. The use of the cards has allowed us to group all that information and separate it from the other elements. In addition, it allows us to summarize the content without seeing all the details of these attacks with the intention of creating a high information scheme to entice users to click for more details on a separate page. In these separate pages we will have a very simple view with a style very similar to the credentials page, giving great importance to whitespace to focus all the attention on the control panel. Note that on the testing page there will be a view to show the results of our actions and that these will not be serious attacks but rather simple for the administrator to understand how it works.

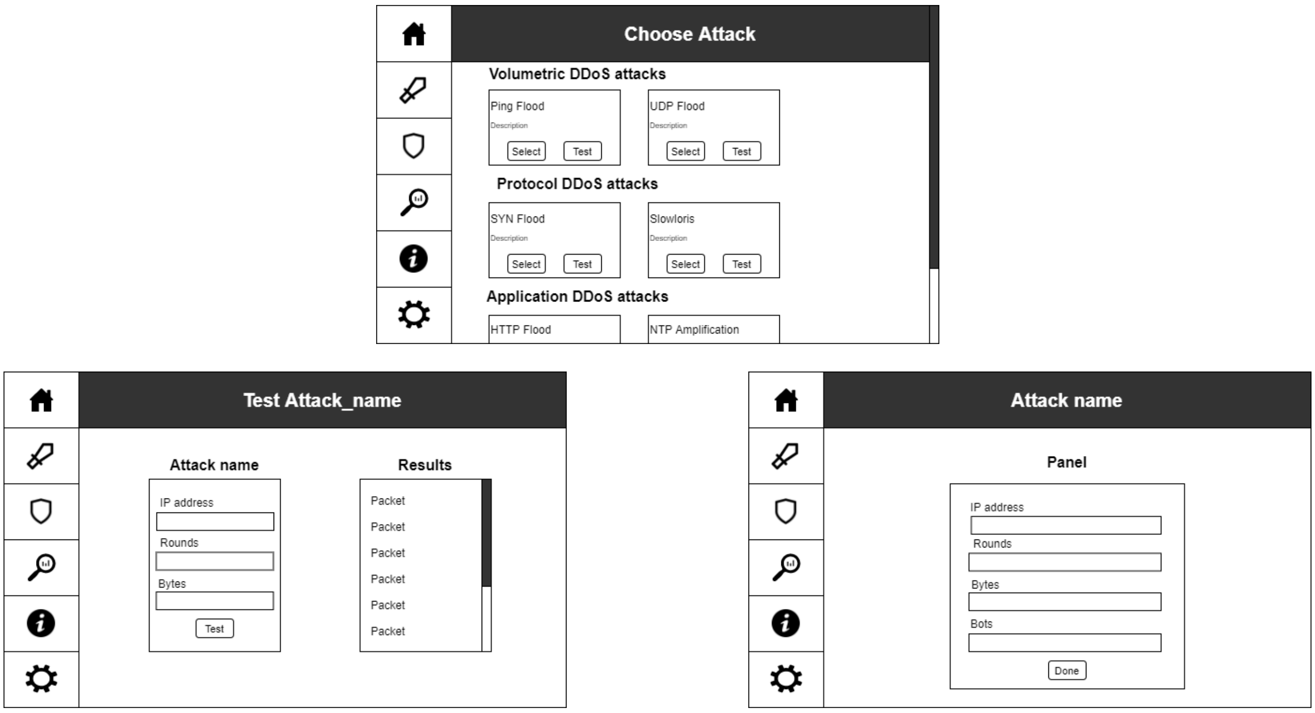


Figure 4.9 Attack page. The style of the page will be the same for each of the attacks. Above we can see the menu from where one of the two options offered per attack will be selected. On the left we see the case of a test of the attack. In the right in case we use the DDoS attack panel.

On the defense page you will find the attacks classified by the tag system and only the preset configuration can be activated/deactivated. Using the toggle switch will allow the admin to perform instantaneous actions that do not need a review or confirmation. Comment that by the time I had made the prototypes I did not have much idea how each of these attacks works in great detail or how they could be mitigated.

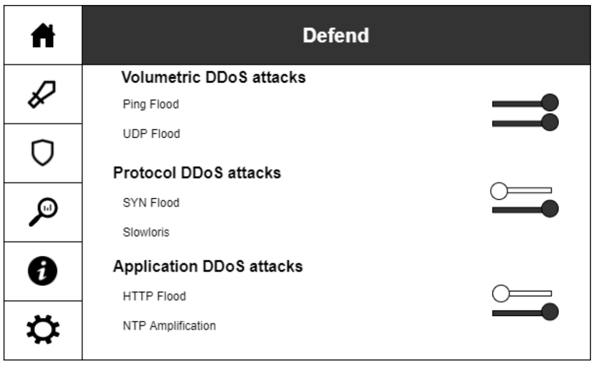
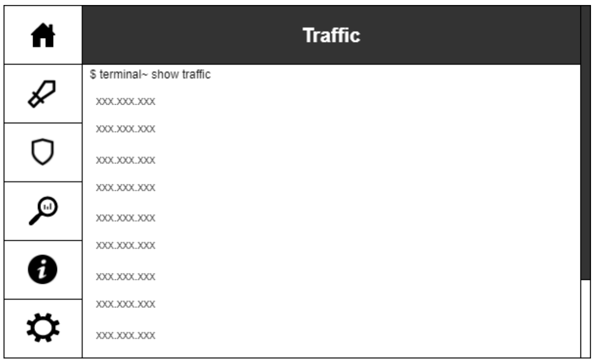


Figure 4.10 Defense page. For each attack, we can able or disable the defense cofiguration already established.

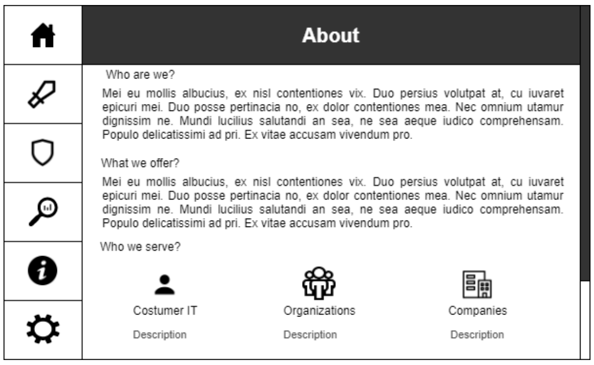
The analysis page will be to visualize different packages or users who visit the page using the style of a terminal to give a touch to the website. Furthermore, as the main costumers are related in the IT sector, we want to give a focus to their personal tastes and something that most of them will be familiar with. This page could contain more information apart from the IP address, since as in the previous case I was not aware of the information that could be taken and displayed.



```
$ terminal~ show traffic
 xxx.xxx.xxx
 xxx.xxx.xxx
 xxx.xxx.xxx
 xxx.xxx.xxx
 xxx.xxx.xxx
 xxx.xxx.xxx
 xxx.xxx.xxx
 xxx.xxx.xxx
 xxx.xxx.xxx
```

Figure 4.11 Analyze page. We see the IP address of each user that visits our home page.

The about page will only contain additional information about the project which will contain a lot of text. Here the micro whitespace is presented since the reading of the text will be simple and it will be possible to differentiate itself from the different sections without difficulties.


About

Who are we?

Mei eu mollis albucius, ex nisl contentiones vix. Duo persius volutpat at, cu iuvaret epicuri mei. Duo posse pertinacia no, ex dolor contentiones mea. Nec omnium utamur dignissim ne. Mundi lucilius salutandi an sea, ne sea aeque iudico comprehensam. Populo delicatissimi ad pri. Ex vitae accusam vivendum pro.

What we offer?

Mei eu mollis albucius, ex nisl contentiones vix. Duo persius volutpat at, cu iuvaret epicuri mei. Duo posse pertinacia no, ex dolor contentiones mea. Nec omnium utamur dignissim ne. Mundi lucilius salutandi an sea, ne sea aeque iudico comprehensam. Populo delicatissimi ad pri. Ex vitae accusam vivendum pro.

Who we serve?

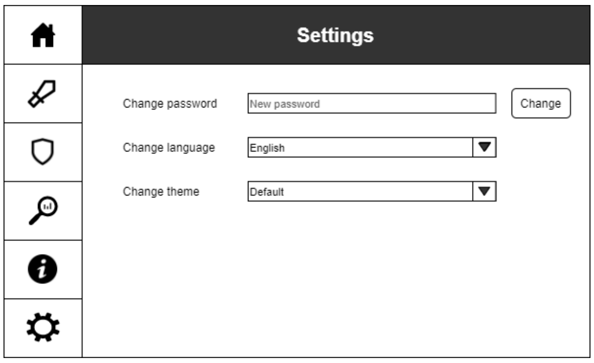
Costumer IT
Description

Organizations
Description

Companies
Description

**Figure 4.12** About page. The text is filler, not the actual description.

Finally, the settings page remains in the same style as the others, and will feature selection or insert text tools that will dynamically make changes. The aim is to ensure that the user makes the least possible effort to make the changes and at the same time that he can find the different options without any difficulty. It must also be said that not many features are offered, which makes this implementation easier. However, when we want to add more features, we will probably have to take another approach or use a labeling system to find the different options faster.


Settings

Change password: New password [Change]

Change language: English

Change theme: Default

**Figure 4.13** Settings page. There is no need for extra pages in order to perform the actions from settings at least for this project.

## 4.8 Website security

This section will cover the different implementations that were taken into account to protect administrators' access to restricted pages, which are only accessible to them. I think it is an important point since giving access to undesirable people could misuse this application and use it for their purposes or interests.

- This application will play a lot with users' public IP addresses. As we have already seen, we store in our database all the IP addresses of our users, whether they have the role of user bee or administrator. When a client visits our website, we will check if their IP address is registered in our database and if it is the case, we will check the role of this. If the role is administrator, all the pages shown in the prototype will be seen. Otherwise, if the role is user bee or the client's IP address is not found in the database, then this client will not be able to access the restricted pages. It will be impossible for the client to access by browsing the web page since the attack and defense icons will not even be displayed in the vertical menu. The client could only enter if they manually type the complete url, but in that case a 403 Forbidden error would appear, thus denying access. Constantly comparing the IP address for each user would add a lot of entries to the relational database, which could be a heavy workload. To check, a query is made to the database searching by role. When getting all administrators, then it looks for an IP address match. A better implementation of this section would be to use Graph noSQL databases of the IP addresses of only administrators. These provide fast data recovery for connected data, so IP address lookups will be much faster. Another way to improve it, would be using a key-value stores since once we have looked for the IP address for the first time we can save it in a Memcahced or Redis type database which will store this data in the memory of the machine instead of do exhaustive searches on the disk. These are possible improvements that I have thought could be made in case I had many users, but since this is not the case, it has been kept simple using only what was commented.

- In the credentials page that will be accesed only if the client's IP address matches with an admin, it will be needed to introduce an username and password. They will only have 3 attempts to do it. Therefore, after the third attempt the page will be blocked to the admin for a certain time. When the admin's account gets blocked, since we have his/her email we will send an email notifying it because an intruser could have tried to enter with the admin's account.
- Inserting the credentials every time the admin accesses the attack page or the defense page could be annoying, so we insert a device cookie session the first time the admin accesses to the respective page. This cookie session will allow to the admin to navigate for a certain time without needing to introduce again the credentials. The cookie is removed if the browser is closed so it will be requested to introduce again the credentials. Moreover, if someone tries to enter with a different browser or machine (but same IP) it will be requested to introduce again the credentials since it will not have the cookie session.
- Django framework already provides an User authentication that I could use to implement this part of the project. However, I wanted to try to do it for myself in order to understand better the risks and how some features are implemented. For example, the password in Django authentication is already hashed and encrypted, but I wanted to know better how it is done so I implemented my own functions to store admin's password adding a salt to it.
- When an user wants to exit from the hive, it will be notified with a message that he/she will not be able to join again for a certain time. This is, because my thoughts were a bit influenced by application DDoS attacks and since this operation of joining and exiting is kinda expensive for my database, I wanted to make sure that no one could take profit of this vulnerability.
- There are control files logs, that contain unexpected errors from the backend. Each line of the file contain the time when the error happened and a message informing the context of the issue. With this I can find some other vulnerabilities that I could get in my website and I can track better where the problem started. Therefore, I can fix or improve some functionalities of my website thanks to the information of this files. In order to not overload this files with information (making them heavy) we will reset the files automatically every certain time. For example, we could say that at 3 am the files should get reseted, so when the time reaches this process will be done automatically.

## 4.9 Attacks implemented

The attacks implemented in this project began by implementing them individually and progressively using sockets. Afterwars, I tested them and added them into the application. They were not all implemented at once, for this look at the planning section. Next, each of the attacks will be explained as they have been implemented (not what they do, this is already explained in chapter 3). We will focus on the testing implementation of these attacks (Dos attacks), since the explanantion of how to perform DDoS attacks using the botnet will de discussed in the next section.

### 4.9.1 Ping Flood

These attacks are based on sending a number of ICMP packets defined by the administrator. The message sent in each packet will be a character (representing 1 byte) repeated as many times as the administrator has specified that he wants the packets to be long (valued in bytes). That is, the message will be of the size specified by the number of bytes since each character represents one byte. Both the number of packets to send and the size of the message will be limited to a certain number so as not to exceed and not create more traffic than necessary since it is only a test. To make tracking more difficult, the source IP address will be randomized.

### 4.9.2 UDP Flood

The implementation of these attacks works as in the previous case regarding the sending of the packets, the message size and the randomization of the source IP address. However, it has some differences and it is that we will send UDP packets instead of ICMP. Moreover, in this case we will specify the port that we want to exploit. It would be advisable to explore if the port to be tested is open to verify that the packets are received. Normally, when attacking, a port that is known to receive UDP traffic is set, such as the port for DNS communication.

### 4.9.3 SYN Flood

This as in the UDP flood attack, we will use the same implementation idea. The idea will once again be to find a port to exploit, and thus try to exploit the vulnerability of the TCP protocol by creating semi-open connections.

### 4.9.4 Slowloris

For the slowloris we will also exploit a specific port and more precisely the port 80 which is where the HTTP communications of our application take place. For this we will send small TCP packages with a delay period of time so that the server interprets it as a bad connection. It will be important to look at the flags, since a SYN packet will be sent seeking to open the connection, and when the SYN-ACK packet is received from the server we will send ACK packets respecting the sequence of the packets.

### 4.9.5 HTTP Flood

Obviously, for these attacks we will have to exploit port 80 where we will send as many packets as the administrator has specified. For these packets we will generate a totally random URL made up of few characters. This will be the request that we will make to the server, which will normally reply that the page does not exist.

### 4.9.6 NTP Amplification

For the use of these attacks, NTP servers using the monlist command have been searched and saved to a list. The monlist is a debug command that allows to retrieve information about the monitored traffic associated with the NTP server. As you can imagine, the answer will not be exactly small, and this will depend on each NTP server. So we will randomly take an NTP server from that list and send as many UDP packets as the administrator specified. The destination port will be specifically specified as 123, since this is where this protocol communicates with the server and we will specify the target's source IP address in order to the NTP server sends the response to the target's server.

## 4.10 Server/client communication (Botnet)

As we have already discussed, users will have to download a file to consider their computer in the botnet. This communication between the server and the client was implemented separately from the main project as it was easier to test and verify that the communication was smooth. Once we had a solid base, where both the client and the server could send and receive messages, we began to add the necessary commands to adapt to the needs of the project. That is, the first step was simply to find that the connection and communication between these two was correct. And the second step was to start thinking about and implementing the commands that the client expected to receive. This will be discussed in more detail later in order to make it clearer to understand. Next, the implemented part of the client will be explained, but not to explain the code itself, but to understand the process and the importance of the status code which allows a fluid communication between the server and the client.

In the first place, we will explain the communication that occurs between these two in a very general way and little by little we will expand each of these steps to see what is happening. In figure 4.14, you can see the main logic of the communication that a given client will follow with the server. For this, the TCP protocol is used, so when the connection has been established we can say that the three-way handshake has been completed. From there, the server will send a specific command and the client after reading and processing it will send a response. The server will then receive the response and display it to the administrator on the web page screen. The administrator will then be the one who decides which command to process, since the client will always be attentive to his message. There will come a time when the administrator wants to close the connection or rather reestablish the connection (usually because there are new bots that want to connect with the server). Then, these new clients (bots) will make the connection while the ones that were already connected will close the connection and then try to reconnect. The server after sending the close command, closes the connection as well, and returns to the point of listening for new connections.

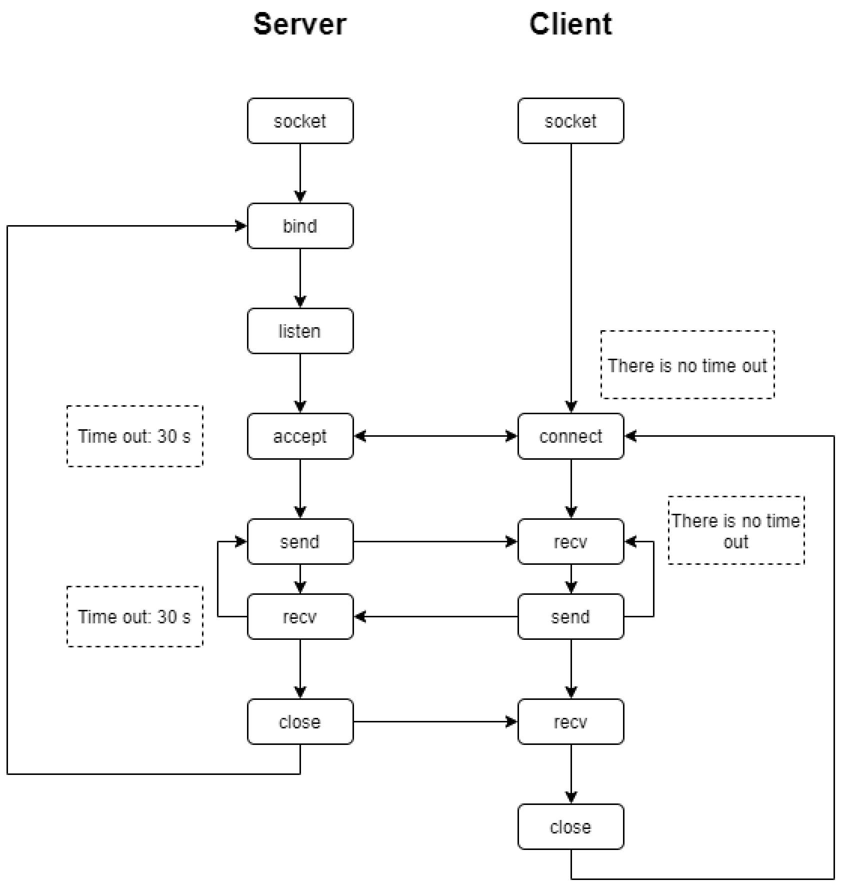


Figure 4.14 Client-server communication flow. The client will not have time outs since it will be always listening to the same server. The server will take some time because it has to listen the response from all the clients.

Now that we have seen how flow works in general, we will explain how the client manages to read and process the commands send by the server and then proceeds to give a response. In figure 4.15 the client logic can be observed in a very simplified way. It will try to read the command and process it. Finally, it will receive a specific message with the status of the processing (we will see it later). Depending on this status, the client will respond with a specific message to the server. Finally, it checks if the state we are talking about coincides with closing the connection. If this is the case, we will close the current connection and try to reconnect with the server. If it fails, we close the program, otherwise it repeats the process. The fact of closing the connection before reconnecting is because the client could open a different port and we do not want to leave ports open without any use.

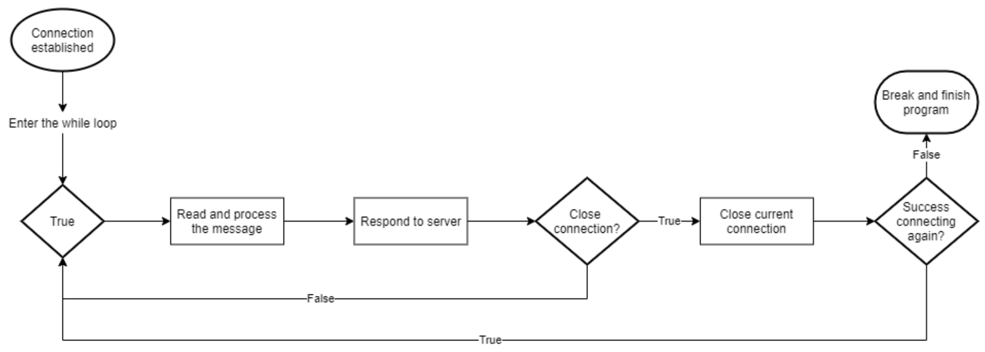


Figure 4.15 Client main logic. The work is delegated to a controller that will take care of the logic to receive the command and to respond to the server.

### 4.10.1 Read and process command

Now we will see how we manage the reading and processing of the command. To do this, we can see figure 4.16, which tells us that we will try to receive the message from the server. In the case that the message could not be received, we will return the EOL status which indicates that we had an end-of-line error. Otherwise, if we have been able to receive the message correctly, we will proceed to validate its format and evaluate it. Comment that although it is not seen in the figure, this whole process is wrapped in a try, except statement which will send the close connection status as a response if it finds an not treated error both in the reading and in the process.

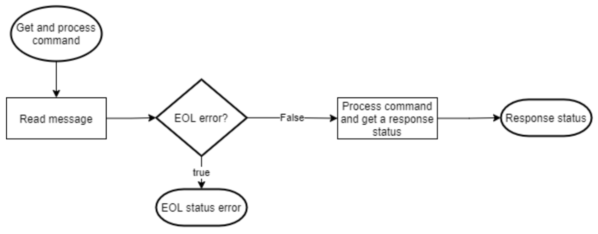


FIGURE 4.16 Reading main logic. The response status is an integer number. So, each status is represnted by a different number.

This message will be read as seen in figure 4.17. First we will initialize some necessary variables and will enter to the infinite while loop. The message will have a header of a specific size which both the client and server must know since it will contain how long the message will be (the part we are interested to receive). The end of the message will be marked with a end of line or better said with "\n". Now, after entering the while loop, we will receive a part of the message by requesting to the buffer a specific number of bytes. That number should be greater than the header in my implementation, because if not, the message length will likely not be read correctly. Then we check if we have received the message. In the case that we do not get a response, we will send an end-of-line (EOL) error status. Otherwise, we check if it is the first iteration of the while loop, that is, the first time we receive the message, because then that means that we have received the header. If it is the first time, we collect the value that is in the header and save it since it will allow us to know when we finish. In both cases, regardless of whether we have read the header or not, we proceed to decode (UTF-8) the partial message received and save it in a variable that will be the final result to be returned. Next, we will check what we said, if the size of the variable that contains the final message is the length specified by the header. If this is the case, we return the final message, otherwise we go to the next piece of message and repeat the process.

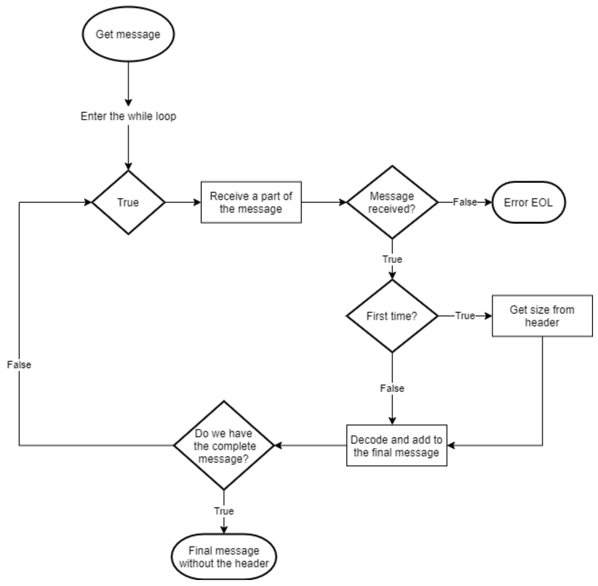

Get message
Enter the while loop
True
Receive a part of the message
Message received?
False
Error EOL
True
First time?
True
Get size from header
False
False
Decode and add to the final message
Do we have the complete message?
True
Final message without the header

FIGURE 4.17 Flowchart receive message. At this point, the client has already established the connection with the server. We return an error of end of line (EOL) or the message received without the header (which contains the message size).

Now we will proceed to process the command. For this we will see if the command contains a pattern of characters that we are sure it must have. At this point, **please look at the readme.md in the 'tfg / client' folder** to understand the format of the commands. To identify the command, first of all it will be need to convert it to lowercase so as not to have any problem with reading the characters. Then, depending on the case we will return a status response or we will call a common function that is responsible for evaluating DDoS attacks. The special cases in this part are the commands that will be used to close connection which will only return the close connection state; also the case of stopping the attack which will assign the global variable (which we will see later) to False (stop) and then it will return the success state; and finally if we do not recognize the command it will return the status of unknown command. For the other commands (DDoS attacks) the common function will be used but depending on the attack it will be passed different arguments. Therefore, from now on we will only focus on a specific command since it will be the same for all cases with some nuances. In any case, if you are interested in seeing all the commands please look at the code as each function and each step is commented in detail.

This function can be observed in figure 4.18. First, we will validate the format of the command, which will be different depending on the type of attack. There will be no need to worry about identifying the format as the command itself already carries the name of the attack and is therefore easy to identify. This validation can return two error states: split error or invalid format. Therefore, if it is the case that the error has arisen, that status will be returned. Otherwise, the validation returns the data already extracted in a list and also the name of the attack type that will be useful later. Next, as the format is correct, we proceed to convert the hostname into an IP address (if it is already an IP address, it returns it to us). If we had an error converting it, we will return a status code to report it. Or else, we proceed to identify the attack with the data we were commenting on. For each of these attacks we have the option of using multi threading or not. If this is the case, we call a function that will create as many threads as specified. If not, we will create a single thread that will be in charge of launching the attack in the background of the application. In this way, we can continue with the logic of our program while the attack is running. In any case, we will return the success code since we managed to launch the attack.

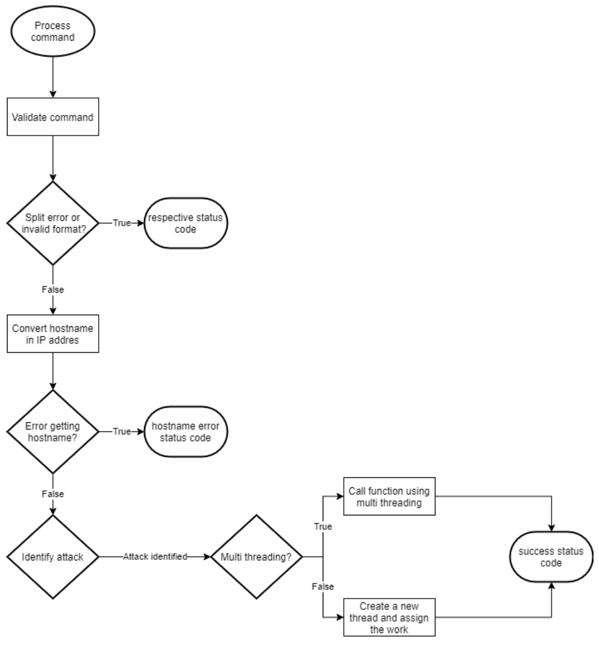


FIGURE 4.18 Flowchart process command. We ending by returning a status code.

And with this the section of reading and process of the command is concluded. Obviously, there are many more functions implemented in the validation and launching section of the attack, but for this you can look at the code, since as said each step is commented. The only thing I want to highlight is that when we launch the attack, we assign a global variable the value of "true" which implies that there is an attack in progress. This variable will be read by all the threads that are executing the attack and they will not stop until this value is "false". So, as we have already seen, the server will need to send a specific command to stop the attack in progress.

### 4.10.2 Respond to server

Now that we have received a status response, we can inform the server with a message reporting what has happened. The response process is much shorter, since we only have to identify the status code and encode the already defined message for each use. The message to be sent has the same format as the one sent by the server. That is, we will have the same header length with a numeric value that will tell us what the size of the message is, and then the specific message. The server to receive the message, will use the same reading strategy that we have seen in the client.

### 4.10.3 Pros and cons

The main advantage of this botnet, I think is that it has a fairly robust communication. By this I mean that even if we have errors the connection does not break, there is always a solution. In the worst case, if we run into an unexpected error (that I was not taking into account), the close connection status will be returned. Therefore, the connection between the client and the server will be re-established. Even if other packets are sent through that port, a clear structure has been defined, so, anything that is not fulfilled will be discarded or it will be reported that the format is not correct. Also, add that all errors will be stored in a specific file with the time when they happened and a message that gives context to what happened. The only bad thing is that although this file has helped me a lot to find bugs in my implementation at the time of development, at the moment I do not have any command created to request a report on the file. Another advantage is that this script consumes very few resources from the client machine. The sleep function has been used at some points in the code so that it does not consume a lot of CPU. Those points are where I am sure that it will not cause problems between the communication of the server and the client, such as when I finish informing the server, or after a period of rounds of attack.

On the other hand, this script is far from perfect. First of all, it is necessary to execute it manually (it does not have to coincide with the moment the server is running since it does not have time outs and there is no danger of it stopping by itself). Also, if the user restarts his computer, he will have to re-launch the script manually, we can say that this is linked to the previous issue. Another point, is that it is not tested (due to lack of time), that is, unit tests or functionality tests have not been created, so more bugs could have been found.

# Chapter 5

# Results

The results of this project can be separated into three main parts, where on the one hand we have the web application itself with its design and its implemented functionalities; the creation of DDoS attacks and the capture of these packets; and finally the communication of the botnet explained in the previous chapter. The set of these parts forms the end goal which we were looking for, the implementation of an IP stressor. Obviously, this is not all that has been worked out with this project, since along the way a great amount of knowledge was been obtained either by working with tools that had not been used before (nmap, wireshark, ntpq, netstat, etc.) or simply the fact of working with different libraries of programming languages, multithreading, databases, and so on.

For this chapter we will focus on explaining the results of these three parts that we have commented, since these secondary objectives will be indirectly reflected. To begin with, we have the website itself where we have based ourselves on a high-fidelity prototype, but in the long run it has undergone some adjustments either due to momentary feedback or due to having acquired new knowledge of the subject that I have need to apply. Given the user stories, it can be verified that we have fulfilled most of them, so the page complies with what we should have been done even though is still not complete. I have not fully reached what was expected because basically good estimates were not made, and the workload was higher than I initially thought. Still, users can perform the most important operations and also many of the less important ones. In figure 5.1 we can see some statistics that can allow us to understand what has been implemented.

| Category | Total US | Total SP | US completed (%) | SP completed (%) |
|---|---|---|---|---|
| DB | 4 | 12 | 75 | 25 |
| WEB | 3 | 9 | 100 | 100 |
| HIVE | 4 | 15 | 100 | 100 |
| T&C | 1 | 2 | 100 | 100 |

| Category | Total US | Total SP | US completed (%) | SP completed (%) |
|---|---|---|---|---|
| NOTIF | 2 | 4 | 0 | 0 |
| DEPLOY | 2 | 10 | 50 | 50 |
| ABOUT | 1 | 5 | 0 | 0 |
| TEST | 5 | 19 | 40 | 31.6 |
| CRED | 2 | 8 | 100 | 100 |
| ATTACK | 9 | 84 | 100 | 100 |
| SNIFFER | 1 | 5 | 100 | 100 |
| DEFENSE | [Cancelled] | [Cancelled] | [Cancelled] | [Cancelled] |
| BOTNET | 4 | 32 | 100 | 100 |
| RESEARCH | 7 | 29 | 100 | 100 |
| SECURITY | 1 | 2 | 0 | 0 |

**Figure 5.1** Work done per category.

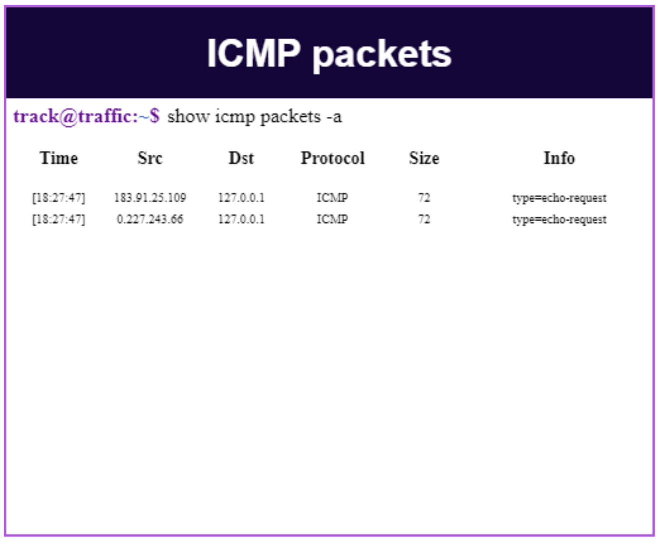
> ***Note:*** *Finally, we  have discarded the defense category as their user stories did not make much sense for reasons already stated. So for the next calculations we will not longer taking into account this category.*

So, as we can see in figure 5.1, the results are partially gratifying as a total of 57% of the categories were completely completed. Furthermore, if we count the total of US completed, this gives us a total of 80%, so it is a quite satisfactory number, especially if we think that those that remained have priority 5 (few), 8 (most of them) and 13 (only one). This of course, without taking into account the US of the deployment with priority 3, which is a different case because it was tried to carry out but could not be fulfilled. Regarding the SP, we obtain a total of 84% of the work concluded. So most of the work is achieved and only minor and low priority functionalities would need to be completed (based on my initial estimates).

As for the DDoS attacks implemented, we will show some images of the testing part since it is where these packages can best be seen. For all the attacks we have used the same destination IP address (127.0.0.1), the same number of turns (2) and the same number of bytes per packet (30).

> ***Note:*** *In each of the following images, we will find an column called "info". This will contain information provided by the protocol we have used and not by the IP frame.*

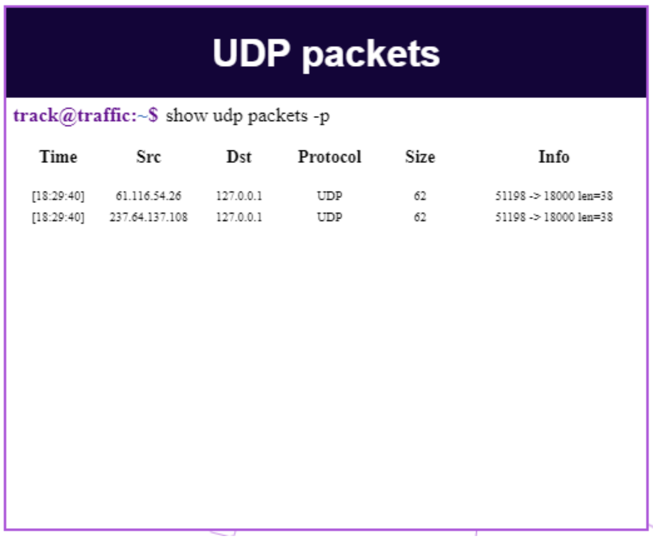
Starting with the Ping Flood, as we can see in figure 5.2 we find totally randomized source IP addresses which at first glance seem valid. I must admit that this part could have been better worked especially by not using private IP addresses or also adding IPv6 addresses. But, for the moment it is what has been implemented. Next, as we have already said, we will use the localhost as the destination IP address and in this case we use the ICMP protocol since we are sending pings. If we are based on the OSI model, ICMP packets work on layer 2, therefore no port is specified. The size of the packet in general is 72 bytes, that is because in addition to the 30 bytes that we have specified for payload, we must take into account the ethernet header (14 bytes), the IP header (between 20 and 60 bytes) and the ICMP header (8 bytes). So adding all these bytes gives us 14 + 20 + 8 + 30 = 72 bytes. In addition, we have extra information from the ICMP frame that indicates that the request type is echo-request. As we can see, we do not get any echo-reply as a response and that is because although we are capturing the packet, we are not responding to it. Likewise, if we answer the address is falsified and would not make sense.



```
track@traffic:~$ show icmp packets -a

   Time          Src           Dst       Protocol   Size          Info

[18:27:47]   183.91.25.109   127.0.0.1     ICMP      72     type=echo-request
[18:27:47]   0.227.243.66    127.0.0.1     ICMP      72     type=echo-request
```

FIGURE 5.2 Ping flood test.

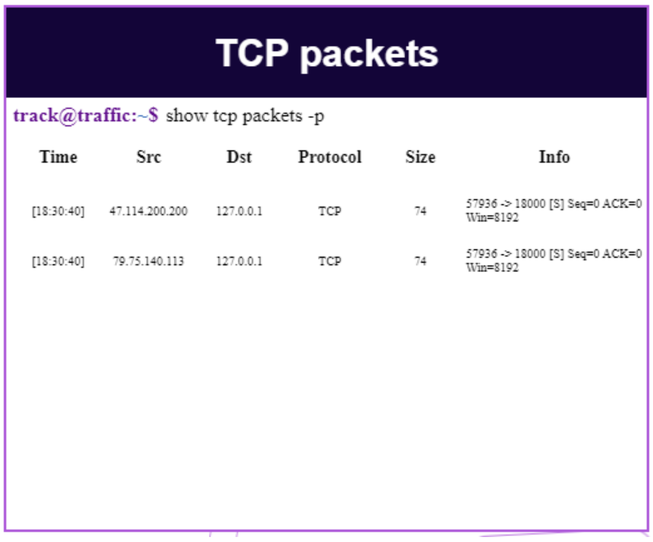
Next, we find the UDP flood attacks as shown in figure 5.3. Which, as in the ping flood, we have spoofed source IP addresses. The protocol in this case is obviously UDP and the size of the packet as in the previous case is made up of the bytes of the headers in addition to the packet itself that we have specified. In this case we have the 4 bytes of the loopback header, 20 bytes of the IP header, 8 bytes of the UDP header and the 30 that we have mentioned. In this case, as additional information we have the source port (51198) and the destination port (18000). It also returns the length of the UDP which contains those 8 bytes of the header plus the 30 that we have already mentioned.



```
UDP packets

track@traffic:~$ show udp packets -p

  Time          Src            Dst        Protocol   Size          Info

[18:29:40]   61.116.54.26     127.0.0.1     UDP       62     51198 -> 18000 len=38
[18:29:40]   237.64.137.108   127.0.0.1     UDP       62     51198 -> 18000 len=38
```

Figure 5.3 UDP flood test.

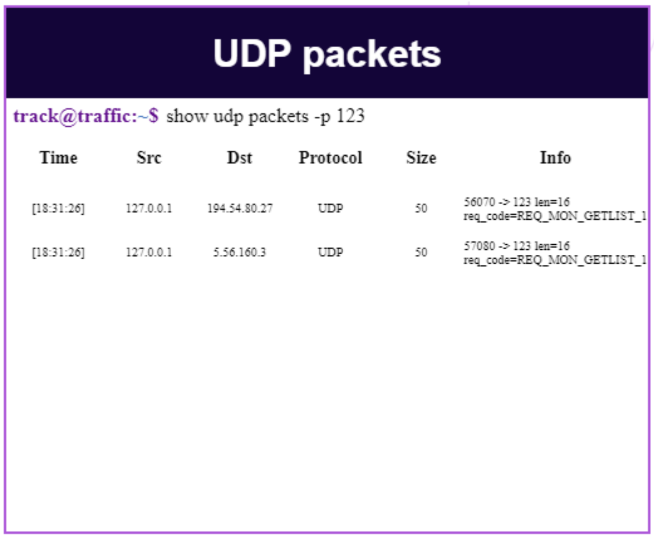
Continuing with the SYN flood attacks, we can see the same structure as in the previous two cases: the spoofed source IP addresses, we continue to use localhost as the target IP address and in this case we use the TCP protocol. The packet size is composed of the 4 bytes of the loopback, 20 bytes for the IP header, 20 more for the TCP header and the additional 30 that we have specified. So far we can obvsrve that in the case of ping flood we have used the ethernet interface and, in UDP flood and SYN flood the loopback interface is being used. This is because I have personally instructed the ping flood attack to use this interface, even if we are using the localhost we force to send the packet through ethernet. Meanwhile, in this attack and in the UDP flood, the interface is assigned automatically, so since we are using localhost, we are using this virtual network interface called loopback. Now, going back to the captured SYN flood packets, we can look at the additional information obtained from the TCP frame. First we see the port of origin and destination and then the type of flag that is assigned to it. In this particular attack, we are using only SYN packets as we do not want to complete the three-way handshake. As in the previous attacks, we do not expect any response (SYN-ACK packets from the target in this case). Moreover, this destination port (18000) that we are using right now is closed, so it basically does not make any damage to the target. I am using a closed port, since if I use an open port we would have seen other packets which are sent from other hosts. So it would be a little bit hard to identify our packets specially if we are using spoofed IP adresses and a random source port. Next, the sequence and acknowledge does not take on much importance in this case, since as we said, we will not complete the three-way handshake. The TCP window size means the number of bytes that the receiver is ready to receive. So in this case we expect a maximum of 8192 bytes.



```
TCP packets

track@traffic:~$ show tcp packets -p

  Time          Src            Dst        Protocol   Size          Info

[18:30:40]   47.114.200.200   127.0.0.1     TCP       74     57936 -> 18000 [S] Seq=0 ACK=0
                                                             Win=8192

[18:30:40]   79.75.140.113    127.0.0.1     TCP       74     57936 -> 18000 [S] Seq=0 ACK=0
                                                             Win=8192
```

Figure 5.4 SYN flood test.

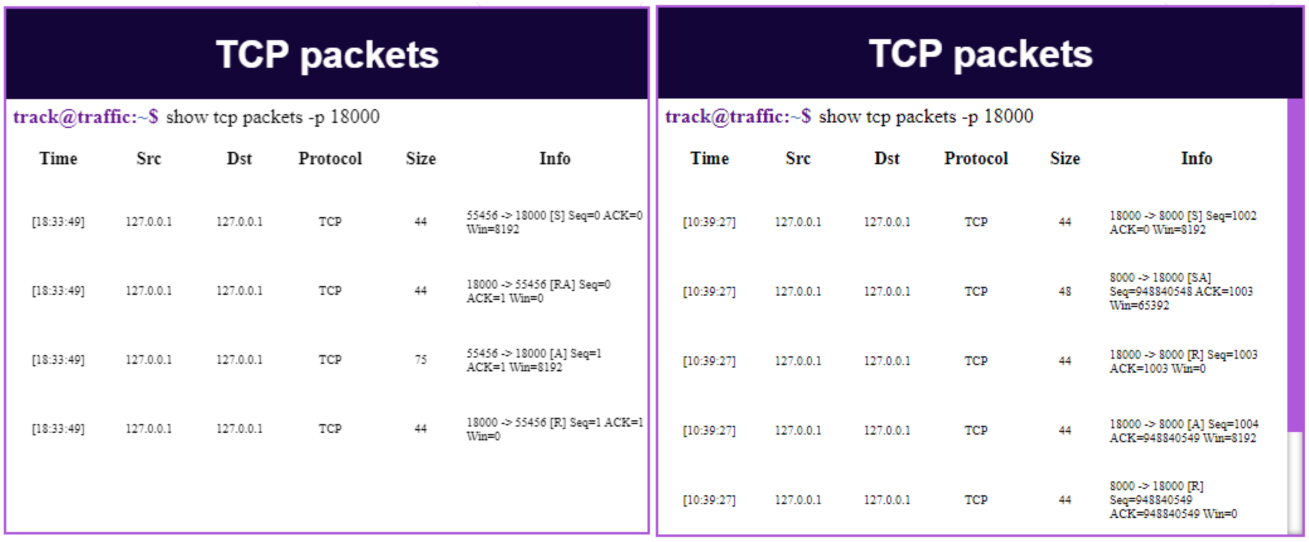
As for the NTP amplification, I have not had the expected result. In figure 5.5 we can see an example of capturing UDP packets destined for port 123. I think the packet itself is well structured, since both wireshark and my capturer show the data in the correct format and the request code is the monlist . So far so good, but as we see we do not get any response from the servers (not even in wireshark). Even if I specify my public IP address and WLAN interface, I do not get any response. After doing some more research, I have used another pattern that uses version 4 (which can be seen in the code) to check if I can receive a response between all the NTP servers that I have saved. The answer was affirmative (checked in wireshark since my capturer is not adapted to accept this packets), so my conclusion is that the servers that I have in the list are not configured to receive this type of monlist requests. Unfortunately I have not been able to find any NTP server that uses the monlist request, not even with the ntpq program. Removing this, in this case the packets have a length of 50 bytes marked by the 4 bytes of the loopback, 20 of the IP header, 8 by the UDP header and 10 by the message containing the monlist request. I probably should have switched to implementing DNS port based reflection attacks. After all, it is the same logic and I probably would have gotten more information. However, I was hoping to find a suitable NTP server for my request.



```
track@traffic:~$ show udp packets -p 123

    Time          Src           Dst         Protocol   Size          Info

[18:31:26]    127.0.0.1    194.54.80.27      UDP        50     56070 -> 123 len=16
                                                               req_code=REQ_MON_GETLIST_1
[18:31:26]    127.0.0.1    5.56.160.3        UDP        50     57080 -> 123 len=16
                                                               req_code=REQ_MON_GETLIST_1
```

FIGURE 5.5 NTP amplification test.

Regarding the slow loris, we have an example for only 1 turn sent in the figure 5.6. In this case, we cannot falsify the source IP address since we want to maintain a communication with the target. However, in the test part of this attack we are not managing to complete the attack, since if we look closely in the left image after sending the SYN packet, the target responds with an RA, which means that the port was not open. After our ACK, the server responds again with an R, which means that it has received an unwanted packet and has ignored it. Now, for an open port like we can see in the right image of the figure 5.6, we achieve to receive a SYN-ACK response. However, we also respond with a R packet. This packet is not send by me, I have not specified to send this packet after the SYN-ACK (check the code) it is done automatically. This is because scapy sends raw packets and we do not have an open socket where the communication can be completed. So, who is sending this RST packet? The answer is the kernel, since the kernel will receive the SYN-ACK first and it will see that there is no socket created to make the communication, so it just sends RST packets before we can do anything with scapy. Therefore, we are not completing the slow loris, we are just sending SYN and ACK packets each turn. This test is really just to see and understand the flow of communication between these two ports using the TCP protocol. To solve this issue, we can drop the RST packet through the iptables. However, I do not want to do a script for removing it since it might affect the clients network behaviour and we are trying to not harm our clients. In the DDoS attack, we will see that we do not have this issue since we are working directly with sockets.

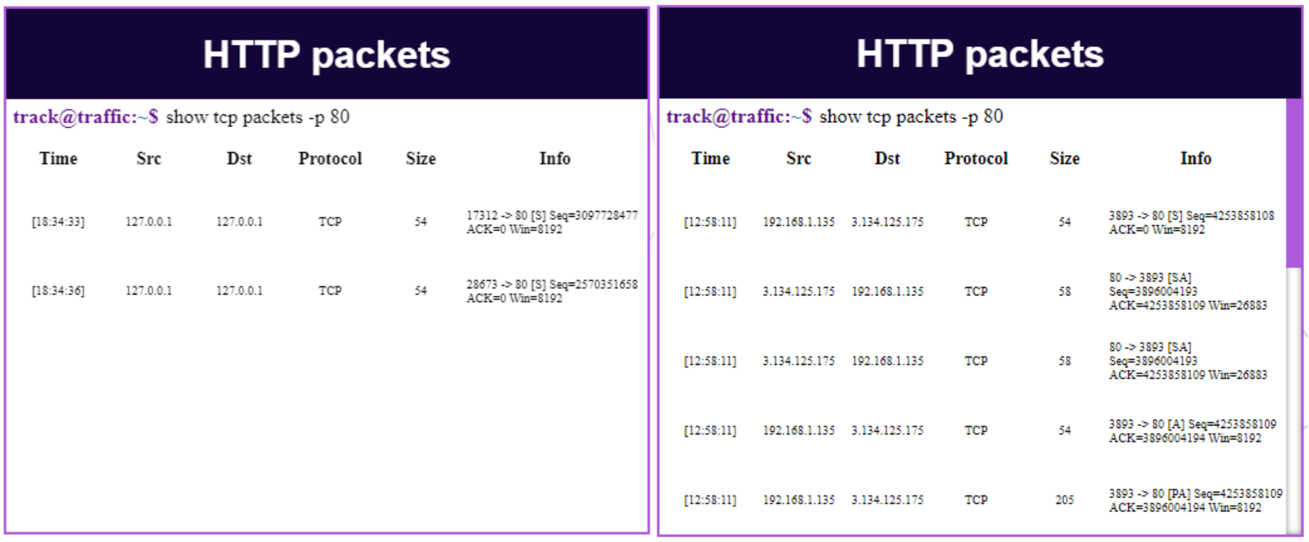


```
track@traffic:~$ show tcp packets -p 18000

Time        Src         Dst         Protocol  Size  Info
[18:33:49]  127.0.0.1   127.0.0.1   TCP       44    55456 -> 18000 [S] Seq=0 ACK=0
                                                    Win=8192
[18:33:49]  127.0.0.1   127.0.0.1   TCP       44    18000 -> 55456 [RA] Seq=0
                                                    ACK=1 Win=0
[18:33:49]  127.0.0.1   127.0.0.1   TCP       75    55456 -> 18000 [A] Seq=1
                                                    ACK=1 Win=8192
[18:33:49]  127.0.0.1   127.0.0.1   TCP       44    18000 -> 55456 [R] Seq=1 ACK=1
                                                    Win=0
```

```
track@traffic:~$ show tcp packets -p 18000

Time        Src         Dst         Protocol  Size  Info
[10:39:27]  127.0.0.1   127.0.0.1   TCP       44    18000 -> 8000 [S] Seq=1002
                                                    ACK=0 Win=8192
[10:39:27]  127.0.0.1   127.0.0.1   TCP       48    8000 -> 18000 [SA]
                                                    Seq=948840548 ACK=1003
                                                    Win=65392
[10:39:27]  127.0.0.1   127.0.0.1   TCP       44    18000 -> 8000 [R] Seq=1003
                                                    ACK=1003 Win=0
[10:39:27]  127.0.0.1   127.0.0.1   TCP       44    18000 -> 8000 [A] Seq=1004
                                                    ACK=948840549 Win=8192
[10:39:27]  127.0.0.1   127.0.0.1   TCP       44    8000 -> 18000 [R]
                                                    Seq=948840549
                                                    ACK=948840549 Win=0
```

FIGURE 5.6 Slow loirs test. Left image communication with a close port. Right image communication with an open port.

Finally, we have the HTTP flood as shown in figure 5.7. In this scenario we have HTTP requests that works on TCP port 80, but we have this port closed on our localhost (as we can see in the image on the left). Therefore, in this test we will only send HTTP requests without any response. However, we will see how it works if the website is deployed in the image on the right of figure 5.7. In this case, we can see that after the connection is established by the three-way handshake, the communication becomes fluid. We can observe 2 SYN-ACK packets sent by the server, and this is because the first packet might have been lost, so after a timeout, the server sends another SYN-ACK packet. Once the ACK packet was received, the HTTP communication began and basically consists of making requests for resources given a random URL. These attacks could be very effective if the victim's resources are known, and knowing how to exploit this application attack to request a large amount of data.

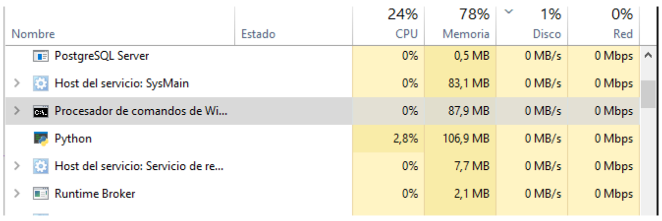


```
track@traffic:~$ show tcp packets -p 80

Time        Src         Dst         Protocol  Size  Info
[18:34:33]  127.0.0.1   127.0.0.1   TCP       54    17312 -> 80 [S] Seq=3097728477
                                                    ACK=0 Win=8192
[18:34:36]  127.0.0.1   127.0.0.1   TCP       54    28673 -> 80 [S] Seq=2570351658
                                                    ACK=0 Win=8192
```

```
track@traffic:~$ show tcp packets -p 80

Time        Src            Dst            Protocol  Size  Info
[12:58:11]  192.168.1.135  3.134.125.175  TCP       54    3893 -> 80 [S] Seq=4253858108
                                                          ACK=0 Win=8192
[12:58:11]  3.134.125.175  192.168.1.135  TCP       58    80 -> 3893 [SA]
                                                          Seq=3896004193
                                                          ACK=4253858109 Win=26883
[12:58:11]  3.134.125.175  192.168.1.135  TCP       58    80 -> 3893 [SA]
                                                          Seq=3896004193
                                                          ACK=4253858109 Win=26883
[12:58:11]  192.168.1.135  3.134.125.175  TCP       54    3893 -> 80 [A] Seq=4253858109
                                                          ACK=3896004194 Win=8192
[12:58:11]  192.168.1.135  3.134.125.175  TCP       205   3893 -> 80 [PA] Seq=4253858109
                                                          ACK=3896004194 Win=8192
```

FIGURE 5.7 HTTP Flood test. Left image communication with a close port. Right image communication with an open port and the webiste deployed with ngrok.

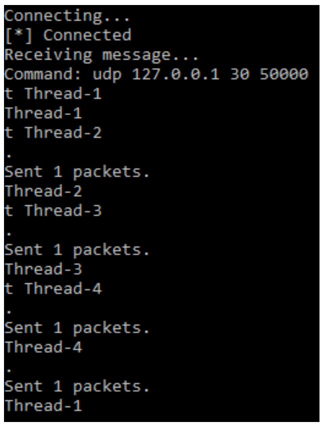
As for the botnet, the idea was to observe the attacks that have been implemented differently from what we have seen so far, and especially the use of multi threading. However, the execution of the code will be somewhat complicated to explain in the document, so I will try to comment on everything that can be demonstrated on paper. The first thing I want to highlight is that the script does not consume a large amount of resources when it is running (both waiting for the next command, and executing an attack). This can be seen in figure 5.8, in which it can be seen that it consumes very little CPU and although it consumes a good amount of RAM, it is not a very excessive value. As for the disk, the use will only be reflected when we have to write an error in the log file, and for the network in this case is not being used because we have executed in the loopback. The day of the presentation will be seen using the internet interface using ngrok (still do not expect a lot of network usage).



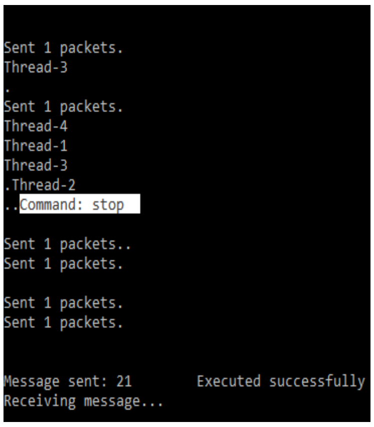
| Nombre | Estado | 24% CPU | 78% Memoria | 1% Disco | 0% Red |
|---|---|---|---|---|---|
| PostgreSQL Server | | 0% | 0,5 MB | 0 MB/s | 0 Mbps |
| Host del servicio: SysMain | | 0% | 83,1 MB | 0 MB/s | 0 Mbps |
| Procesador de comandos de Wi... | | 0% | 87,9 MB | 0 MB/s | 0 Mbps |
| Python | | 2,8% | 106,9 MB | 0 MB/s | 0 Mbps |
| Host del servicio: Servicio de re... | | 0% | 7,7 MB | 0 MB/s | 0 Mbps |
| Runtime Broker | | 0% | 2,1 MB | 0 MB/s | 0 Mbps |

FIGURE 5.8 Script performing UDP Flood with multi threading (4 threads).

Regarding the communication and processing of the attack, we can see figure 5.9 how the client receives the command from the server and proceeds with the attack until it receives the stop command. The screenshots can be a bit difficult to read in these cases. That is why the botnet demonstration will be left for the day of the presentation, since it is also necessary to explain how these commands are called from the server among many other things. So the last part of the results remains unfinished in the document and a demo will be seen where the operation of these attacks is demonstrated. Although it is difficult to visualize this part, observe that 4 threads are being created where each one is sending packets. Moreover, in the left image we can not see it because of the threads, but in the right image there is a response from our client to the server explaining that the command has been executed successfully. The number 21 means the length of the string which we have mentioned, and the spaces between the 21 and the string are because of the header size already explained in the Botnet implementation.



```
Connecting...
[*] Connected
Receiving message...
Command: udp 127.0.0.1 30 50000
t Thread-1
Thread-1
t Thread-2
.
Sent 1 packets.
Thread-2
t Thread-3
.
Sent 1 packets.
Thread-3
t Thread-4
.
Sent 1 packets.
Thread-4
.
Sent 1 packets.
Thread-1
```



```

Sent 1 packets.
Thread-3
.
Sent 1 packets.
Thread-4
Thread-1
Thread-3
.Thread-2
..Command: stop

Sent 1 packets..
Sent 1 packets.

Sent 1 packets.
Sent 1 packets.


Message sent: 21        Executed successfully
Receiving message...
```

FIGURE 5.9 Script that performs UDP Flood with multiple threads (4 threads). In the image on the left, the client is connected to the server and is waiting for a message. In the image on the right, the client has received the stop command that is causing the attack to stop working. In both cases the client is responding to the server (even if we can not see it in the image on the left it happens too).

# Chapter 6

# Project closure

With this last section we conclude the work, where we will explain the stricken objectives, the different problems and inconveniences that I have encountered throughout the work and it has caused me some delays. Finally, the approach to be given to this long-term project will be explained, since so far it has only been implemented as a demo to demonstrate the potential it can be used for.

In general, the main objective has been carried out, which consisted in creating this web application that facilitates the use of DDoS attacks to test how servers perform under stress. However, not all initially proposed user stories have been implemented as they were either discarded or complicated to implement and other alternatives were chosen.

## 6.1 Problems solved

With this project we have tried to create a web page that unites a community to face DDoS attacks. The application created will make it possible to put servers or network infrastructures under stress using different types of DDoS attacks and tools. In this way, the capacity of these servers can be tested and thus be able to adapt new mitigation techniques. In addition, it is intended to inform and make understood the functionalities of the implemented protocols and at the same time offer guidance on their execution.

## 6.2 Issues during the project

First of all, when it came to design decisions and implementation orders, I was pretty misguided. This can be clearly seen in the estimation of user stories, since although I have started to study web protocols and tried to understand how the network is structured before starting to implement the application, I was not very sure how I would implement these DDoS attacks and many other functionalities. For this reason, throughout the project I have been making constant changes and improvements, since as I was learning more ideas emerged. Even so, thanks to the initial planning I was able to make the base of my application and from there start making the modifications that I thought were most convenient.

Regarding the implementation of the attacks, I started using only low-level sockets. However, I had conflicts with Django, and to work in better conditions I have seen that it is better to use the Django Channels extension. In spite of that, I decided to use a library that allowed me to manipulate network packets called Scapy. The main reason I opted for this option was because it made my work easier and using Django Channels would have taken me a long time to adapt. The biggest problem I had working with this library was when it came to capturing the sent packets. Although the use of this library seems very simple, I was not making good use of filters and did not understand very well the importance of network interfaces. Also, I was wasting a lot of resources on my machine trying to capture those packets and some of them were getting lost because I could not filter them properly. So, I got to a point of blaming the library that it was not working properly (I withdraw it). After researching the documentation, I found the necessary tips to be able to carry out my tasks. Moreover, in the documentation I observed that there is an asynchronous function to capture those packets, with which I managed to lower the consumption of the application.

Another issue I would summarize it in many ideas but little time. As I mentioned before, throughout the project I understood better what I was doing and my eyes were opened to new possibilities. However, that caused me to lose my way a few times because I was deviating from what I was initially trying to do (an IP stresser). In order not to get out of the way, I tried to continue with the initial planning, and before trying to add a new functionality I wondered if it really added value to what I was doing in this project.

Now, my biggest problem during the project was the fact of working in short periods of time (1 week sprints). The first months I think I was going at a good pace and was complying with the plan, but in the long run, due to issues not related to work, I was wasting time and I could not be attached to the computer to perform my tasks. In order not to waste too much time, I have spent some time just doing research and studying new technologies or methods that could be useful to speed up my thinking. Nevertheless, I have always tried to act in accordance with the initial terms and try to fit in with the demands of the job. Except for the last sprint, the tasks have been fulfilled in all the sprints and the methodology described in the planification section has been followed in all of these sprints.

Finally, as I just said, the last sprint I could not complete the marked tasks (related to deployment using Apache and Microsoft IIS). At that point I deviated a bit, and ended up opting for another alternative (Ngrok). Subsequently, the tasks that were pending I have carried out in a single final sprint since it would have taken me too long to restructure the planning within a short time margin.

## 6.3 Future features and plans

I think this project is just the beginning of something great. I think it has enough potential to continue working on it and to create a larger community. Opting for the help of new people to come up with ideas and sharing the work with other developers would greatly improve the application and attract more people thereby growing the community even faster.

On a personal level, there are some functionalities that I wanted to implement during the project but I felt that they were not important enough to give more priority to what I had marked up to that moment.

One of the most important things that I think I have not done are the unit tests and the functionality tests on client-server communication. Although it is working properly, I think that creating these tests would allow me to find possible bugs of which I am currently not aware and even improve the robustness of it. This would be one of my highest priorities when continuing with the project. Continuing with this topic, we should improve the script in general, but, being more specific, make it run in the background of the user bee computer without having to execute it every time we want to use it. Obviously, for this, even our server would need to be running constantly, so we should think of a better way to do the final deployment. Third party services could be used, but this could be very expensive as we would normally be generating a lot of traffic. Therefore, the most convenient thing is either to look for special plans and agree a price with a third party, or to have our own servers. In any case, it is required to have a source of income. Since the application is free, there is no source of monetization to carry out these balances. One way to solve this would be to seek donations or make product exchanges.

So far, I am the only one who has access as administrator to the website, but the idea is to give access to other entities or individuals who want to use it. In order to avoid misuse issues, we should think carefully about which measures to take and make sure that they are only capable of testing the IP address or host name that they requested. Then we would have to think of some way to claim that the owner of that IP gives consent to test their network and restrict the target IP address field of the attacks.

Another improvement has to do with the frontend part. The web application is currently not fully responsive and this can detract from the user experience. In addition, it is not tested to work with other browsers other than Google Chrome or Mozilla Firefox. Likewise, we need to seek feedback and improve or modify the functions that we already have in place to see what direction to take and where more work is required. In any case, we offer our email so that clients can come up with their criticisms or suggestions to helps us to offer the best possible resources to our users.

Finally, enhance the currently implemented DDoS attacks and create new ones. Since so far, I have been learning step by step how these packets work, understanding what each bit is, and exploring the particularities of each one to lay a firm foundation. Now, I should be able to get more juice out of it and put in more creativity to have unique and more detailed attacks that stand out from what has been implemented so far.

# Bibliography

[1] Denise Berard. (February 22, 2018). *DDoS Breach Costs Rise to over $2M for Enterprises finds Kaspersky Lab Report*. Recovered from https://www.kaspersky.com.

[2] Oleg Kupreev, Alexander Gutnikov and Ekaterina Badovskaya. (October 28, 2020). *DDoS attacks in Q3 2020*. Recovered from https://securelist.com.

[3] Oleg Kupreev, Ekaterina Badovskaya and Alexander Gutnikov. (February 16, 2021). *DDoS attacks in Q4 2020*. Recovered from https://securelist.com.

[4] (May 7, 2020). *Ddos-for-hire: Teenager sold cyber attacks via website*. Recovered from https://www.bbc.com.

[5] (n.d.). *Booters, Stressers and DDoSers*. Recovered from https://www.imperva.com.

[6] Luke Plunkett. (December 31, 2013). *Hackers Claim Takedown Of Battle.net, League Of Legends, EA.com*. Recovered from https://www.kotaku.com.

[7] Kevin Rawlinson. (March 31, 2015). *Evidence links China to GitHub cyber-attack*. Recovered from https://www.bbc.com.

[8] Taylor Gadsden. (April 16, 2020). *Mbps vs. Gbps: What internet speeds do you really need?*. Recovered from https://www.allconnect.com.

[9] (February 26, 2020). *What Is A Botnet & How Does It Work?*. Recovered from https://outpost24.com.

[10] Nick Babich. (November 24, 2020). *The UX Design Process: Everything You Need to Know*. Recovered from https://xd.adobe.com.

[11] Django (Version 3.1) [Computer Software]. (2013). Retrieved from https://www.djangoproject.com/.

[12] Scapy (Version 2.4.5) [Computer Software]. (June 16, 2021). Retrieved from https://scapy.readthedocs.io/.

[13] Ngrok (Version 2.0) [Computer Software]. (n.d.). Retrieved from https://ngrok.com/docs.

[14] Netstat [Computer Software]. (October 16, 2017). Retrieved from https://docs.microsoft.com/es-es/windows-server/administration/windows-commands/netstat.

[15] NTP distribution (Version 4.0) [Computer Software]. (March 31, 2014). Retrieved from https://www.eecis.udel.edu/.

[16] Pablo Fredrikson [Pelado nerd]. (May 1, 2019). *NGROK - Exponé tu app con SSL GRATIS!* Retrieved from https://www.youtube.com/.

[17] [__Drunk Engineer__]. (January 25, 2017). *OSI and TCP IP Models - Best Explanation*. Retrieved from https://www.youtube.com/.

[18] Chuck KeithNugget [NetworkChuck]. (August 6, 2020). *what is TCP/IP and OSI? // FREE CCNA // EP 3*. Retrieved from https://www.youtube.com/.

[19] Chuck KeithNugget [NetworkChuck]. (August 27, 2020). *how the OSI model works on YouTube (Application and Transport Layers) // FREE CCNA // EP 5*. Retrieved from https://www.youtube.com/.

[20] Chuck KeithNugget [NetworkChuck]. (December 6, 2020). *let's hack your home network // FREE CCNA // EP 9*. Retrieved from https://www.youtube.com/.

[21] Chuck KeithNugget [NetworkChuck]. (July 4, 2020). *FREE CCNA // What is a Network? // Day 0*. Retrieved from https://www.youtube.com/.

[22] Chuck KeithNugget [NetworkChuck]. (October 3, 2020). *i bought a DDoS attack on the DARK WEB (don't do this)*. Retrieved from https://www.youtube.com/.

[23] Ignacio Gentile [Nate Gentile]. (April 5, 2020). *El CIBER-ATAQUE más grande de la historia... ¿Empezó con MINECRAFT? | Ciberseguridad EP 2*. Retrieved from https://www.youtube.com/.

[24] [Computerphile]. (November 9, 2016). *Slow Loris Attack - Computerphile*. Retrieved from https://www.youtube.com/.

[25] [Computerphile]. (May 31, 2017). *How TOR Works- Computerphile*. Retrieved from https://www.youtube.com/.

[26] [Computerphile]. (May 31, 2017). *Hashing Algorithms and Security - Computerphile*. Retrieved from https://www.youtube.com/.

[27] [Computerphile]. (November 20, 2013). *How NOT to Store Passwords! - Computerphile*. Retrieved from https://www.youtube.com/.

[28] [Computerphile]. (December 30, 2013). *The Problem with Time & Timezones - Computerphile*. Retrieved from https://www.youtube.com/.

[29] [F5 DevCentral]. (April 10, 2018). *What is a Web Application Firewall (WAF)?*. Retrieved from https://www.youtube.com/.

[30] [F5 DevCentral]. (July 3, 2019). *How To Inspect Secure Traffic*. Retrieved from https://www.youtube.com/.

[31] [F5 DevCentral]. (March 12, 2018). *OSI and TCP/IP Model Overview*. Retrieved from https://www.youtube.com/.

[32] [F5 DevCentral]. (November 1, 2017). *What is DDoS?*. Retrieved from https://www.youtube.com/.

[33] [F5 DevCentral]. (July 19, 2017). *Attack Mitigation with F5 Silverline*. Retrieved from https://www.youtube.com/.

[34] [F5 DevCentral]. (January 11, 2018). *BGP Overview*. Retrieved from https://www.youtube.com/.

[35] IIS [Computer Software]. (December 6, 2018). *Configuración de aplicaciones web de Python para IIS* . Retrieved from https://docs.microsoft.com/.

[36] Django (Version 3.2) [Computer Software]. (2013). *How to use Django with Apache and mod_wsgi*. Retrieved from https://www.djangoproject.com/.