UNIVERSITY OF TWENTE

MASTER THESIS

Signature-Based DDoS Attack Mitigation: Automated Generating Rules for Extended Berkeley Packet Filter and Express Data Path

Author:
Huub VAN WIEREN

Examiners:
Dr. J. J. CARDOSO DE SANTANNA
W. VAN DER HOUVEN MSC (KPMG)
Dr. A. PETER

November 26, 2019

Abstract

Distributed Denial of Service (DDoS) attacks are malicious attempts to disrupt a service from the target by overwhelming it by network packets. DDoS attacks are continuously rising in size and diversity. In 2018, Netscout reported a peak of 1.7 Tbps in size [1] and Akamai's annual report of 2018 [2] states that those spikes are still growing with an increasing growth curve. As an example from the beginning of 2018, with the new memcached attacks, attackers are still finding new ways to perform DDoS attacks. Cloudflare is one of the biggest vendors on the market providing solutions the defend against DDoS attacks. Their defending methods include the filtering of malicious packets by generated rules from attack signatures. The extended Berkeley Packet Filter (eBPF) and eXpress Data Path (XDP) form an important part in those defending methods. With the ability to filter packets at a very high speed, eBPF and XDP prove in existing solutions that it can perform in the fight against DDoS attacks. With eBPF and XDP, malicious packets can be dropped based on rules specified inside the eBPF program. Studies show that eBPF and XDP are tools that are able to drop packets at higher rates than former tools. However those studies only show this with plain packets and not in the case of an actual DDoS attack. Altough eBPF and XDP are open-source, the tools can not directly be used to mitigate DDoS attacks. In practice a network operator has to know how to use this tools and what the implication of different scenarios can be. Therefore, the overall goal of this study is to research how to use eBPF and XDP to mitigate DDoS attacks and to research how effective the tools can be. A DDoS mitigation system is proposed in this study with the use of eBPF and XDP. With this system a network operator is able to drop packets up to a 100% accuracy when deep packet layers are considered. The XDP filter allows higher packet processing speeds than an Iptables filter with the same rules. The contribution of this study is two-fold. It adds new scientific findings on which new studies can build upon and the study can be put in practice by network operators in real network environments.

Contents

List of Acronyms

ACL	**A**ccess **C**ontrol **L**ist
AS	**A**utonomous **S**ystem
BGP	**B**order **G**ateway **P**rotocol
C&C	**C**ommand and **C**ontrol
CDN	**C**ontent **D**elivery **N**etwork
DDoS	**D**istributed **DoS**
DoS	**D**istributed **D**enial of **S**ervice
DPS	**D**DoS **P**rotection **S**ervice
eBPF	**e**xtended **B**erkeley **P**acket **F**ilter
IDS	**I**ntrusion **D**etection **S**ystem
IP	**I**nternet **P**rotocol
IPS	**I**ntrusion **P**revention **S**ystem
IRC	**I**nternet **R**elay of **C**hat
ISP	**I**nternet **S**ervice **P**rovider
RTBH	**R**emote **T**riggered **B**lack**H**oling
WAF	**W**eb **A**pplication **F**irewall
XDP	e**X**press **D**ata **P**ath

Chapter 1

Introduction

Distributed Denial of Service (DDoS) attacks are not a new threat in our society. The presence of studies against DDoS attacks from the early 2000's, prove that the fight against DDoS attacks is ongoing for already a long time [3]. Reports show that DDoS attacks still form a severe threat and are actually growing in number, intensity and diversity [4]. For example, Netscout [1], reports a threefold increase in the number of DDoS attacks against third party data centers and cloud services with a record breaking 1.7 Tbps in size. Also Akamai's annual report of 2018 shows that the size of DDoS attacks is growing with 9% quarterly and that attackers are still finding new ways to leverage their DDoS attacks. For instance the new memcached reflection attacks in 2017, formed the cause of some record breaking DDoS attacks [5]. All those facts together highlight the need for more research into DDoS mitigation, despite the many mechanisms that actually already exist as shown by Osanaiye et al. [6].

DDoS attack mitigation can be realised by recognising and dropping the DDoS network packets in a network. Dropping packets at high speeds has to be done carefully, because legitimate traffic should ideally not be dropped. To be able to classify whether a packet is legitimate traffic or malicious traffic, classification techniques have to be adopted. As defined by Osanaiye et al. [6], those techniques to detect DDoS attacks can be split into signature based detection techniques and anomaly based techniques. The Signature based detection techniques use a set of DDoS attack characteristics in a database. Those characteristics are derived from known attacks that occurred the past for example. Those characteristics can be used to match against monitored traffic to detect any malicious activity. Anomaly based detection methods use techniques without any prior known information and usually leverage some form of machine learning. Signature-based techniques usually have a higher accuracy on detecting attacks over anomaly based methods. New attacks however, are more difficult to detect with signature-based techniques as maintaining up to date signatures is challenging.

One of the biggest vendors on the market of DDoS mitigation solutions is Cloudflare. As a leader in this market they have adopted DDoS mitigation mechanisms that outweigh competitors. The automated DDoS mitigation system of Cloudflare named GateBot is using the features of the extended Berkeley Packet filter (eBPF) and the eXpress Data Path (XDP) [7]. Those features include the ability to drop network packets at high speeds. XDP and eBPF work on an operating system level, which is the level that forms a bridge between hardware and software in the userspace. Altogether, eBPF and XDP are tools that can help to build infrastructure to make signature based defense mechanisms. Fabre [8] from Cloudflare made an article about Cloudflare's tool named L4Drop, which works complementary with Cloudflare's GateBot. L4Drop uses packet dropping rules in combination with eBPF and XDP. They however don't mention how those rules are generated exactly. Facebook uses XDP not directly in the fight against DDoS, but as a signature-based firewall in general [9]. Their solution is capable of automatically updating rules, but again, unclear is how those rules are generated. Outside the solutions of Cloudflare and Facebook there are little to no other public production use-cases that use eBPF and XDP as a packet filter. Høiland-Jørgensen et al. [10] show with their study on eBPF and XDP that DDoS traffic filtering is possible without the need for special hardware. A search on Google Scholar with the search term 'DDoS XDP eBPF' returns 25 academic works, but none of these works study how to automatically generate rules for signature-based DDoS mitigation with eBPF and XDP. Our hypothesis is that by studying rule generation in this domain, an effective method can be created to mitigate DDoS attacks.

An existing platform containing DDoS signatures is called DDoSDB launched by dr. Jair Santanna from the Univeristy of Twente [11]. Those signatures open the possibility to study signature based detection techniques, including how to derive rules from those signatures. This platform also hosts attack traffic which can also be used for research against DDoS attacks. With DDoSDB as a source of DDoS attack signatures, it could potentially function as a base to automatically generate rules for a eBPF and XDP filter. As mentioned there are however no studies that show how to do this automatic generation for eBPF and XDP. Therefore the goal of this study is to design a method for automated rule generation inside eBPF and XDP together with signatures from DDoSDB.

To pursue our goal, the following research questions (RQ) are defined as the basis of this research:

- **RQ1:** What methods currently exist that automatically update rules in a signature based DDoS mitigation?
- **RQ2:** How to automatically generate eBPF rules for DDoS mitigation using DDoS attack signatures?
- **RQ3:** What is the effectiveness and the usability of the designed DDoS mitigation system using DDoS attack signatures?

The three given research questions define the structure of this study. First RQ1 will be answered in chapter 2. It will give insight into what DDoS attacks are, including possible mitigation methods. On answering RQ2, the design of our own mitigation method will be described in chapter 3. The method will make use of a signature based techniques using XDP and eBPF. It will be tested against real DDoS attacks from DDoSDB. Then, chapter 4 will discuss the results of those tests, which will give answer on to RQ3. Finally in chapter 5, the conclusion of this study will be given.

Chapter 2

Background of DDoS Attacks & Mitigation

The overall goal of this chapter is to give answer to ***what methods currently exist that automatically update rules in a signature based DDoS mitigation? (RQ1)*** *The answer of this question will be given by the structure of multiple sections. Section 2.1 will elaborate on what DDoS attacks are. By taking those DDoS attacks into account, section 2.2 will focus on possible DDoS attack detection and mitigation methods. Then, in section 2.3 the workings and possibilities of eBPF and XDP in a signature based DDoS mitigation method will be discussed.*

2.1 DDoS Attacks

For the first section in this chapter, we want to understand what DDoS attacks actually are and how they work. The goal of this section is to give some background on DDoS attacks, how are they performed and which kind of attacks actually form a threat nowadays.

2.1.1 DDoS Background and Taxonomy

DoS and DDoS attacks are performed with the purpose of stopping a certain target or multiple targets from delivering a service. In the case of a DoS attack, the attack is performed with the use of just a single machine. Kenig et al. [12] outlines the history of DoS and DDoS attacks and states that the first DoS attack dates back to 1974. Then one of the first large-scale Distributed DoS attack occurred in 1999. As a distributed attack it obtained its strength by harnessing the resources of many machines together. DDoS attacks were due to their distributed character way more powerful then DoS attacks and caused bigger problems in the early 2000's. In 2002 a targeted victim by a DDoS attack was a DNS server [13], which caused that many internet users were not able to browse the web. Since then DDoS attacks still causes unavailable services on the internet.

Understanding what the weaknesses are of DDoS victims is essential to understand how DDoS attacks work and why it is possible that an online services become unavailable. Back in 2004, Mirkovic and Reiher [14] classified DDoS attacks by several different features, including the classification by what weakness they try to exploit. They make a distinction between semantic and brute-force attacks. By sending requests semantic attacks misuse a certain feature or bug from a protocol or application at the victim, that causes the victim to consume all resources it has. This kind of attacks do not focus on the the amount of requests that is send to the victim, but about what the victim does with those requests. An example could be to sent very complicated database queries to a database server which causes the database server to waist all computational resources and therefore, becomes unavailable for other users of the database. The other type of attacks, brute-force attacks focus on generating and sending more traffic than the victim can process. This can also exhaust resources at the victim, but this time it is due to the amount of traffic the victim receives. The categorisation of flooding attacks and semantic attacks is commonly used in literature [15], other studies refer to brute-force attacks as flooding or volumetric attacks [16]. Note that, the two classes of attacks do not mutually exclude each other, which means that an attack can fall in both classes of flooding attacks and semantic attacks. Furthermore, also note that by definition even semantic DDoS attacks use multiple machines to target the victim, since the attack is distributed. Therefore the amount of requests to successfully perform an semantic DDoS attack also plays a role, but doesn't have the focus.

As mentioned, semantic attacks misuse the behaviour of applications and protocols. Flooding attacks however exhaust resources that are also bound to protocols. Zargar et al. [16] classifies attacks based on what protocol is targeted by its requests, attacks on a network or transport level on the one hand and attacks on an application level on the other hand. Those levels can be found in the Open Systems Interconnection (OSI) model [17], which defines 7 layers in the computer network architecture. The network layer is layer 3 (L3) is the OSI model and the function of the layer is to route packets. The internet protocol (IP) works on this layer. Layer 4 (L4), the transport layer has the function of segmenting data before it goes to L3 and L4 also reassembles data after it comes from L3. Furthermore, L4 includes port addresses to deliver messages to the correct processes in layer 5. TCP and UDP are examples of protocols functioning in L4. Application level attacks exploit protocols on layer 7 (L7) of the OSI model, where protocols as HTTP can be found. This way DDoS traffic that reaches this last layer by targeting the HTTP protocol for instance, has been trough all the other layers as well. A DDoS attack with requests targeted at the TCP protocol in L4, will however not reach L7.

Concluded can be that, as long as they exist, DDoS attacks are made possible by sending requests in a distributed manner to its victim. Those requests are exploiting weaknesses of its victim by using semantic attacks and flooding attacks. How these attacks exactly cause a victim to stop delivering a service is determined by what protocols are involved, occuring in different layers of the OSI model.

2.1.2 Attack motivations

Akamai specifies 6 common types of motivations for DDoS attackers [18]. Furthermore also Fenil and Mohan Kumar [19] sums up several incentives to perform DDoS attacks. Some DDoS attack incentives and motivation together with some examples attack motivations are listed as follows.

- **Hacktivism** includes all hacking acts that have political motivations by attacking companies or governmental institutions. DDoS attacks are a subset of those hacking acts. The DDoS attacks against payment companies as Mastercard and Paypal when they closed donations to wikileaks are examples of hacktivism attacks [20].

- **Extortion** is another motivation of attackers to use DDoS attacks. Attackers demand the victim to pay ransom, by threating them with DDoS attacks [21]. The group that called themselve DD4BC is an example of attacker that asked bitcoins to their victims via mail, in exchange for stopping DDoS attacking the victim.

- **Script Kiddies** form another threat on the internet. Young people, often children form a threat if they are playing around with freely available tools or booters. They perform attacks to just play around, trying to make news headlines or hitting their friends to gain advantages in online games [22].

- **Distraction** can also be an incentive to perform DDoS attacks. Sometimes this is called DDoS as a Smokescreen and is performed to hide other malicious activity. The Sony Playstation hack for example was masked by days long DDoS attacks. Most times companies find out days later that they have financial loses or that data is stolen [23].

- **Market manipulation** can be the goal a DDoS attacker. By DDoSing certain companies, attackers can gain advantages with their position on stock markets [24].

Revenue losses and reputation damage form the biggest threats for businesses as a consequence of DDoS attacks [25]. Depending on the type of business, revenue losses vary, but cost a business tens of millions of dollars per hour when services are totally down. With the risk for businesses of losing their reputation or revenues, DDoS mitigation mechanisms will have to applied at business level. Not only business are impacted by DDoS attacks however, state actors or critical infrastructures can be targeted as well. Electricity supply systems for example can become a target as well, since those kind of systems all become connected to the internet [26].

2.1.3 Architecture of DDoS attacks

A DDoS attacker is the initiator of a DDoS attack. To perform a DDoS attack an attacker has to have a set of machines to target his victim. In a typical DDoS attack those machines are malware infected machines that thereby become remotely controllable. Literature refers to those machines with different names, such as zombies, slaves, daemons, bots or agents [14, 19]. The number of bots used in a in a DDoS attack can be over thousands [27]. It is possible to command and control those bots either in a more manual way or in an automated controlled way [14]. The level of automation varies between semi-automatic and fully automatic communication. In an fully automatic method no direct communication with the bots is needed between the attacker and bots, since they are fully hard-coded with a specified with the start time, attack type, attack duration, and source IPs of the victim machines [28]. Semi-automatic attacks can offer more control to the attacker as the attack type, victims, time and duration can be specified by him. To command and control those bots a botnet has to be formed [29]. In a botnet the bots are not only controlled, but botnets can also adopt mechanisms to infect new bots in an automatic way. Hoque et al. [28] defines a botnet as a collection of many malware infected machines controlled by malicious entities. Note that this definition does not necessarily mean that botnets are used to perform DDoS attacks, but can also be used to perform other mailicous activities. In modern botnets, bots are not directly controlled by the attacker, but via controllers that operate on Command and Control (C&C) servers [30]. Figure 2.1 gives an overview of a botnet structure.

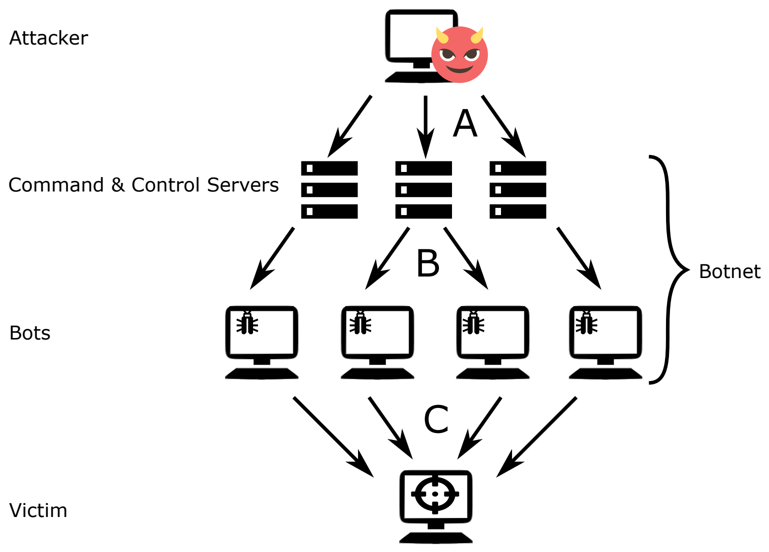

FIGURE 2.1: DDoS attack trough a botnet structure

The attack path from the attacker to the victim of a DDoS attack is displayed at Figure 2.1. The attacker controls the C&C server at A, after which the C&C server can send commands to the bots at B. The bots on their turn, direct their malicious traffic towards the victim at C. The C&C servers are responsible for infecting more bots and keeps communicating with them, therefore the C&C servers including the involved communication at A and B form the backbone of the botnet [30]. Using C&C servers in a botnet structure not only automates workflows of the attacker, but can also hide the identity of the attacker. The design of a botnet depends on which model is used. Alomari [31] defines several different botnet-models that are also used for DDoS attack. The first one is the 'Agent Handler' model, where the C&C servers are other compromised computers systems called handlers. Those handlers are infected with software that is able to command and control the bots called agents. In an Internet Relay Chat (IRC) model an attacker doesn't compromise machine other then the bots. IRC is normally used as a text-based chat system through communication channels. Those communication channels can also be used to create a botnet by sending commands in a private chatroom full of infected and connected bots. The final C&C model is the web-based model, where commands towards the bots are send trough the web protocol HTTP or HTTPS. The C&C web-based model for botnet became most common and popular nowadays [32].

Instead of directly sending traffic to the victim, bots can also send requests trough reflectors [33]. Figure 2.2 shows how those reflectors are operating is DDoS attack.

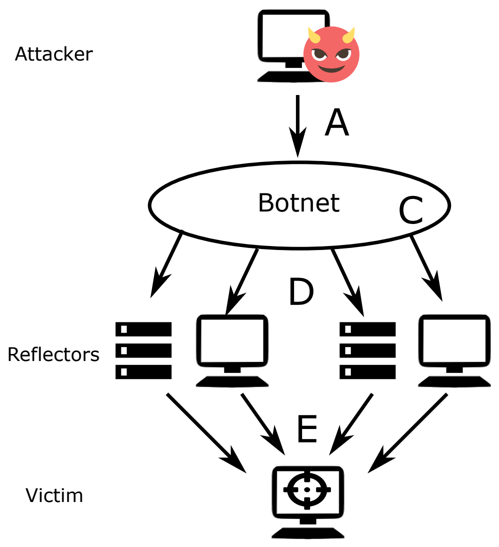

FIGURE 2.2: Reflected DDoS attack

In a reflection attack, bots can send their traffic to any device on the internet that gives a response. The traffic at D in Figure 2.2 is directly coming from the bots in the botnet and exists of requests for the reflectors. The reflectors respond on those requests with a response. Those responses are then send to the victim at E, made possible by IP address spoofing. IP address spoofing is done by, altering the source address of the traffic at D. The reflector then assumes the response has to be send to this false IP address, which is the IP address of the victim. Compared to non-reflected DDoS attacks, reflected attacks become even more distributed and even harder to trace back. Due to Reflection attacks, amplification attacks become possible which can increase the malicous traffic size. When the response of the reflector is bigger in size then the request, traffic size can be amplified. The amplification factor determines the ratio between the response size and request size and can easily be one order of magnitude big, Thomas et al. [34] shows this by using DNS requests for example. Note that those amplification attacks misuse the behaviour of normal innocent services that are just freely available such as DNS servers for example.

To sum up, DDoS attacks can be leveraged by the use of botnets existing out of C&C servers and bots. Furthermore reflectors enable the the possibility of attack traffic amplification to generate bigger traffic sizes.. The quarterly DDoS attack size growth of 9% doesn't grow continuously, but is determined by peaks in the growth curve as stated by Akamai [35]. Those peaks are set by the attackers discovery of new ways of building botnets or by the discovery of new reflection attacks.

2.1.4 DDoS attack vectors

An attack vector of a DDoS is determined by the kind of messages sent by the bots of a botnet. Table 2.1 displays today's common attack vectors, together with its description. Roughly from the top to the bottom of the table, the attack vector generates less malicious traffic. As described in section 2.1.1, semantic attacks gain their strength by misusing a certain feature or bug in a protocol and flooding attacks gain their strength by the amount of traffic generated. Therefore, flooding attacks in OSI layer 3 and 4 generate the highest volumes of malicious traffic, semantic attacks in those layer are already effective with less traffic. Attacks in OSI layer 7 or the application layer tend to come in the lowest volumes of traffic, compared to the other attack vectors. In the table, one of the columns ′**S**′ or ′**F**′ is ticked respectively whether the attack is more semantic in nature or if the attack is more focused on flooding. The Protocol column shows which protocol is targeted with the attack and the ′OSI L′ column shows in which OSI layer this protocol operates. The final column shows if the vector typically uses reflectors as in Figure 2.2.

TABLE 2.1: DDoS attack vectors

Attack Vector	**Description**	**S**	**F**	**Protocol**	**OSI L**	**Can be reflected**
UDP Flood	An UDP attack targets random ports on a victim its computer or network to flood as much, not necesarrily valid, packets as possible. The system of the victim will try to determine what application is listening on this port and will eventually send back that the destination is not reachable. Spoofed ip address make sure that this response directed to the bots. By flooding the this traffic the victims network link can be overflooded [36].		x	UDP	L4	
UDP amplification	Protocols on top of the UDP protocol can be used to generated malicious response messages. Instead of TCP, UDP doesn't require to establish a connection between device before sending messages. Therefore messages on top of UDP can be spoofed and then be reflected towards the victim. The reflected messages are amplified and the amplification factor depends on the protocol [34].		x	UDP/*	*	x
ICMP flood	ICMP echo requests are used to ping a certain target. The attacker will try to send those requests in a higher number than the victim can handle [37].		x	ICMP	L3	
SYN flood	Every TCP connection is initiated with a SYN message, followed by a responding SYN-ACK from the receiver. The connection is then finally established by an ACK message from the initiator. To exploit the behaviour of this protocol an attacker can flood SYN messages to its victim, but then not respond on the SYN-ACK messages. This way the victim keeps open half-open connections in its memory which will eventually be exhausted [38].	x		TCP	L4	
SYN-ACK flood	An attacker can also choose to overload its victim by overloading it with SYN-ACK messages by sending SYN messages to reflectors on which the victim will exhaust its resources by trying to figure out why those messages are incoming [39].	x		TCP	L4	x
ACK flood	After establishing a TCP connection, the receiver of a messages from the other side acknowledges the receivement with an ACK message. In an ACK flood attack, the victim receives ACK messages and failes to link them to an active TCP session. As a result the victim resources can be exhausted by [16].	x		TCP	L4	
DNS flood	With this attack a DNS server is targeted by overloading it with DNS requests [40].	x		UDP/DNS	L7	

HTTP GET flood	The attacker sends legitimate looking HTTP traffic to the victims webserver containing GET requests. These request are normally used to obtain a webpage. When the Get request are flooded to the victim it will consume potentially more resources on the server than it can handle, sometimes also called an excessive VERB attack [41].		x	TCP/HTTP	L7	
HTTP fragmentation	The bots of the attacker establish a valid HTTP connection HTTP connections with the webserver of the victim. By fragmenting packets send to the webserver and sending them very slowly, the webserver will have to hold up many connection, which ultimately can exhaust its resources [41].	x		TCP/HTTP	L7	

Note that table 2.1 doesn't specify all available attack vectors, but structures some common ones and gives an idea on which different kind of attacks are possible. Singh et al. [41] for example specifies 20 different attack vector only using the HTTP protocol. Furthermore the UDP amplification vector in table 2.1 doesn't specify any targeted protocol. This is because several protocols can be used on top of UDP to amplify traffic. Next some examples of those amplifying protocols and how they amplify traffic.

- The **NTP** protocol is generally used to synchronise time between computer systems. An NTP server can also return up to the last 600 used IP addressees that have connected to the server by using the MONLIST command. The MONLIST command of 64 bytes can be amplified up to 100 responses of 482 bytes each. The typical UDP listening port of NTP is port 123. [42].

- The **DNS** protocol is used to translate domain names into IP addresses. The ANY request to a DNS server returns all records available for a certain domain name, which results in response message of over 4000 bytes with just a request message of 60 bytes. The typical UDP listening port of DNS is port 53 [43].

- The **SSDP** protocol part of the UPnP protocol, which is made for devices on a network to discover each other, such as as personal computers or printers. With a SSDP discovery request, all other device with UPnP services reply with a packet for each service they have configured. This way traffic amplification factor of 75 and higher can be reached. The typical UDP listening port of SSDP is port 1900 [43].

- The **CharGen** is an old protocol initially intended for testing purposes by generating random characters, but not really used for this purpose anymore. This protocol can be misused to generate amplified DDoS traffic return 358.8 bytes on average, with requests of just a single byte. The typical UDP listening port of CharGen is port 19 [43].

- Quote of the Day (**QOTD**) is a protocol that is old and barely used anymore. It has been used to generate a quote for users in the network and can also be misused to generate amplified traffic. The typical UDP listening port of QOTD is port 17 [43].

- The **CLDAP** protocol is the Connection-less Lightweight Directory Access Protocol and is designed to look up small amounts of information in a directory by complementing the LDAP protocol Response traffic can amplifiy traffic with a factor of over 50 [44]. The protocol works on port 389.

- The **Memcached** protocol is generally used to cache memory for database-driven websites. An open port on 11211 can result in an 15 byte request being amplified up to 750000 bytes [45].

According to DDoSMon [46] more than half of all DDoS attack vectors today are UDP amplification attacks. Other common UDP based attacks are UDP flood and DNS flood. The most common TCP attacks are the SYN floods and ACK floods. Other common attacks are against webservers using HTTP, but DDoSMon does not specify what attack vectors exactly. Apart from the QOTD protoco, the previously summed up amplification protocols are the most frequently used ones according to DDoSMon. Also Akamai reports those protocols as most frequently used in amplification attack [5]. Last sections the term 'attacker' is constantly referred to as a person that initiates a DDoS attack, in practice however, an attacker doesn't necessarily need any knowledge of any of the above technical aspects of a DDoS attack. Booters or stressers form platforms that offer DDoS attacks targeting a specified victim, in return of a payment. This way anyone with web access is able to start an actual DDoS attack [47].

Several aspects of DDoS attacks are summed up in this section; the working of DDoS attacks, different DDoS attack types and the kind of protocols that are used and misused within DDoS attacks. The essence of a DDoS attack is that is that it exhausts resources of its victim in such a way that the victim is not able to deliver a certain service properly. This means that as long as resources of a certain service are limited, the service will be prone to resource exhaustion and thus also vulnerable for DDoS attacks. The next section will look into existing ways to defend such a service against those DDoS attacks.

2.2 DDoS Attack Detection and Mitigation

In the previous section we gained insight in what DDoS attacks are and how they work. The goal of this section is to study existing ways to mitigate DDoS attacks. Also available DDoS attack mitigation studies will be evaluated in order to gain insight in the whole field of DDoS mitigation.

2.2.1 DDoS Defense Taxonomy

Where Mirkovic and Reiher [14] classify different DDoS attacks, they also classify defense mechanisms to protect against DDoS attacks. First of all, defense mechanisms are divided into preventive mechanisms and reactive mechanisms. Then the reactive mechanisms can be sub-classified by their detection method. Finally a classification can be made by the deployment location of the defense mechanism. All those different classes will be described in this section.

Preventive mechanisms try to prevent DDoS attacks before they can even start. For instance, Hardening IoT devices [48], can be used to prevent IoT devices from getting compromised and being exploited in a botnet, with the result that less IoT devices are used to perform DDoS attacks. Although those kind of preventive mechanisms can be really effective, they will never 100% guarantee that a DDoS attack will never happen at all. The reason that DDoS attacks will always be present is that attackers are constantly finding new ways to compromise systems and new ways to generate malicious DDoS traffic. Therefore, reactive mechanisms will always be needed to respond on an actual ongoing attacks, which is the scope of this study.

Reactive mechanisms are relying on DDoS detection methods in the first place. The detection has be done as fast and accurate as possible to be adequately respond on a DDoS attack. The detection of a DDoS attack is typically done by either an anomaly-based detection method or a signature-based detection method. Signature-based detection methods detect the occurrence of attack with already known attack patterns, whereas anomaly-based detection methods use machine learning for example to detect abnormal behavior. Besides detecting DDoS attacks as fast as possible, those detection methods should also identify malicious traffic as accurate as possible. This means that normal traffic and malicious traffic are classified in the correct class as much as possible. Normal traffic is often also called benign traffic. Making a distinction between benign traffic and malicious traffic can be challenging, as attackers can try to imitate benign traffic as close as possible to make malicious look like just normal traffic. On the other side benign traffic can have behaviour patterns that are very similar to DDoS attacks. Flash crowd events for example, in which a server receives a sudden peak in the amount of traffic, can cause a detection mechanism to think of a DDoS attack. Flash crowd events can happen because of breaking news on a news website for instance [49]. Signature-based mechanisms in general have a higher accuracy than anomaly-based mechanisms, but signature-based mechanisms can only detect attacks that are specified by a limited amount of rules. However, anomaly-based mechanisms in general are better in detecting all possible attacks including common attacks, rare attacks and new attacks. Also anomaly-based detection do not need a constant update of rules as with signature-based mechanisms. After the detection of an attack, a reactive defense mechanism responds to the attack. Typically those responses include filter mechanisms to filter out the malicious traffic, but they can also include a reconfiguration of a network.

Another classification in defense mechanisms is made by deployment location. For instance, Zargar et al. [16], classifies DDoS defense mechanisms by deployment location. For the deployment location three different places are defined. The 'source-based' mechanisms that are deployed as close as possible at the source of the attack and also the 'destination-based' mechanisms that try to the defend the system of the victim itself. Lastly, the 'network-based' mechanisms hit the networks involved in between the source and the destination, which is basically the rest of the internet apart from the the source and destination of the attack. The last type of methods are the hybrid mechanisms using a combination of the given mechanisms. Ideally DDoS attacks are mitigated by source-based mechanisms as close as possible at the source of the attack. This way the probability of any harm done will be as small as possible. Those kind of mechanisms however are in general difficult to deploy since the attack traffic is distributed and therefore difficult to detect and filter. Source-based mechanisms Also the sources of an attack can be difficult to trace and it can be challenging to determine a responsible party to pay for the deployment of such source-based mechanisms. Network-based mechanisms can monitor traffic on a large scale, but they require high storage and processing capabilities. Furthermore, with network-based mechanisms, classifying traffic in benign and malicious traffic is difficult, due to the large volume of traffic. Destination-based mechanisms are easier and cheaper than other mechanisms to deploy since it is easier to aggregate traffic near the victim. It also becomes more easy to deploy mechanisms against application-level attacks, since not only the data in the network layers can be read. The attack traffic, however, might already do damage before the destination-based mechanisms can do their work. At the end, Zargar et al. [16], states that defense mechanisms should be divided over multiple collaborative deployment locations. Manavi [50] categorises defense mechanisms in a similar way and also recommends to further study collaboration methods between different defense mechanisms. Overall, concluded can be that DDoS defense mechanisms can be broken down in preventive and reactive mechanisms, where reactive mechanisms use anomaly-based and signature-based detection methods. Further conclusion is that those reactive defense mechanisms can occur in different deployment locations, ideally having methods that enable defense mechanisms to collaborate with other defense mechanisms.

2.2.2 DDoS Attack Mitigation methods

In the previous section an overall idea is given on how and where DDoS defense mechanisms operate. This section will elaborate on some specific mitigation methods that are used in DDoS defense mechanisms. Source-based methods as Ingress and Egress filtering [51], D-Ward [52] and MULTOPS [53] are mainly focusing on network devices near the sources of an DDoS attack. Those devices are typically not reachable by entities that actively try to defend themselves against DDoS attacks. Given that source-based methods lay out of scope for many parties and that source-based methods have a low accuracy in general [54], source-based methods will not be further discussed here. Source-based methods as such however can still be relevant as they can and exist complementary to network-based and destination-based methods, but are not relevant for the scope of this research. In the next subsections, specific network-based and destination based mitigation methods will be further discussed.

2.2.2.1 BGP routing

The internet consists of a network of Autonomous Systems (ASs), which are collections of IP routing prefixes. By 2019 there are over 90000 autonomous systems with an unique number [55]. Internet Service Providers (ISPs) are the most typical owners of ASs, but also other large organisations such as universities can own an AS. The Border Gateway Protocol (BGP) is the protocol that helps ASs to exchange routing information with each other. Determining the shortest path between ASs is part of this protocol for instance. BGP is also involved at Internet Exchange Points (IXPs), that form the physical infrastructure through which ISPs can route traffic between their ASs. On the internet level of IXPs and ISPs, BGP enables methods to mitigate DDoS attacks with the following methods.

- **Blackholing** or more specific 'Remote Triggered Blackholing' (**RTBH**) [56], is used inside a single or multiple ASs to drop network traffic. Instead of directing traffic the through the correct route, blackholing causes network traffic to be dropped by directing it to a null route . BGP announcements towards all other BGP routers, cause that all traffic directed to or coming from a certain range of IP addresses is being dropped. After the detection of a DDoS attack, these announcements can be initiated by the victim, but also by an intermediate device in the network. Note that traffic that is being 'blackholed', is dropped purely based on an IP prefix of either the source or the destination of the traffic, no other properties of the traffic are used to drop traffic. Therefore, dropping traffic using RTBH can be effective to mitigate a DDoS attack, but it is likely that legitimate benign traffic is also dropped [57].

- **BGP Flowspec** [58] is an extension of BGP, which enables the possibility to filter and drop traffic based on a certain flow specification. With BGP Flowspec, Access Control Lists (ACLs) can be distributed among all BGP Flowspec enabled routers. These ACLs contain filter rules based on 12 different parameters with details available in OSI layer 3 and 4, such as IP addresses, ports and TCP flags for example. Filtering traffic based on more information than just IP address prefixes, BGP flowspec has the potential to mitigate DDoS attacks more accurately than RTBH. Studies show that with BGP flowspec, both the detection of DDoS attacks and mitigation can be automated in systems as RADAR [59] and Stellar [60] for example.

As stated, BGP operates on a high level at IXPs and ISPs, to provide communication between different autonomous systems. Therefore BGP processes internet data in high volumes that involves all different kind of internet traffic. Even with BGP Flowspec, the latter makes it challenging to accurately filter DDoS attack traffic from benign traffic. Furthermore, BGP Flowspecs rules are limited by the the fact that only 12 different parameters can be used and no further packet information can be used. Besides this limitation, there is also a limit on the amount of rules a BGP router can apply. For example, in the documentation of Ciscos BGP Flowspec enabled routers, a maximum of 3000 rules is stated [61]. Nevertheless, mitigating DDoS traffic with BGP Flowspec can potentially be done in a more granular way than with RTBH.

2.2.2.2 Network Firewall

The purpose of a network firewall is to form a barrier between two networks, by filtering network packets. With that definition of a network firewall, BGP Flowspec can be seen as a network firewall. However BGP Flowspec can only be deployed on BGP enabled routers available on ISP and IXP levels, whereas more traditional firewalls can occur on all sorts of network edges.

With a study by Mogul [62], the first generation of network firewalls dates back to 1989. It was a packet filter to filter network packets based on their source and destination addresses, ports and protocol. Since then all sorts of firewalls are developed with the purpose of filtering incoming and outgoing network traffic. In 1998 Julkunen and Chow [63] introduced a dynamic packet filter, which was able to keep track of connection states. From that moment, two sorts of packet filtering firewalls existed, the 'stateless' and 'stateful' firewalls. A stateful firewall is also able to determine if a packet is initiating a new connection or if the packet is part of an already established connection. Next some open-source firewalls for the Linux operating system will described.

- The **Iptables** [64] packet filter was introduced as part of the Linux kernel in 2001. Since then it has been the main Linux firewall for years and is also used in the fight against DDoS attacks [65, 66]. With Iptables it is possible to put rules in a table that specify what traffic should be filtered. To optimize the filtering process, those rules should be ordered in a way that the rules with the highest change to match should be on top of the table. Ip also provides the possible to track connection in a table table, which makes Iptables a stateful firewall.

- **Nftables** [67] was introduced for the Linux kernel in 2014. Where Iptables existed of multiple seperate modules, Nftables combines them all and shows performance benefits over Iptables [68]. In the performance of filtering malicious DDoS packets, Nftables shows similar results compared to Iptables [69].

- The **extended Berkeley Packet Filter (eBPF)** [70] is an extension of a former BPF [71] and improves on the performance. eBPF enables the possibility to put programs including filtering rules inside the Linux operating system. The filtering itself is can be done via the **eXpress Data Path (XDP)** hook, which enables higher packet processing speeds than other hooks in the kernel. Not only processing speeds are improved, but eBPF also adds possibilities to interact dynamically with other programs running in user-space. Furthermore, filtering rules are not bound to specific packet header fields, but rules can basically be made on all possible bytes in a network packet. Studies show that eBPF is a promising DDoS mitigation method since it has flexible filtering rules and packets can be dropped had higher speeds than Nftables and Iptables [72].

The use of network firewalls against DDoS attacks can be limited by the firewall configuration or processing speed. To mitigate a DDoS attack with a network firewall, the firewall must be able to process the packets at least as fast as they are incoming. If the network firewall is not able to process those packets in time, a congestion can take place resulting in a denial of service. Therefore, firewalls should be carefully configured and also be as fast as possible. As shown by Salah et al. [73], network firewalls can potentially be targeted by a DDoS attack depending on the behaviour and the properties of the network firewall.

2.2.2.3 IDS and IPS

As described by Liao et al. [74], the Intrusion Detection System (IDS) and the Intrusion Prevention System (IPS), are systems that detect and stop an intruder of bypassing security mechanisms. Such an IPS or IDS can either function on a computer or on a whole network of devices. Characteristic for an IDS or IPS is that they usually exist of multiple elements and they can analyse all sorts of events and objects on multiple places in a network. For example, Hwang and Gangadharan [75] shows that already in 2001, a former version of Iptables is used as a building block inside an IDS. Strictly an IDS is a system that purely monitors and detects intrusion without taking care of stopping it. In literature often the terms IDS and IPS are used as synonyms, since in practice most systems contain both detection functionalities as well as stopping functionalities. This research will generally refer to an IPS from now on, considering it also contains detection mechanisms. The term IDS will only be used to specifically refer to the detection part of a certain system.

IPSs can be categorized in signature-based and anomaly-based systems, similar to the categorisation of DDoS defense mechanisms as outlined in subsection 2.2.1. Signature-based systems use certain rules derived from known DDoS attack to detect attacks, whereas anomaly-based systems detect attacks based on irregular behaviour on all sorts of components in a system. Anomaly-based DDoS defense mechanisms do most often occur in IPSs instead of other DDoS defense mechanisms, since IPSs often have access to many resources in a network. Some specific IPS techniques and methods related to the detection and mitigation of DDoS attacks will be summed up and categorized by their detection method.

- **Signature-based Systems**. Snort [76] is an open source IPS based on signatures. It has been adopted in several IPS solutions proposed by several studies. For example, Bakshi and Yogesh [77] propose a DDoS defense system using Snort in a virtual machine. It is able to drop packets coming from given ip addresses to try to mitigate the attack coming from those addresses. Another study that adopts Snort is one from Lonea et al. [78], it uses a MySQL database to store and match known DDoS attacks. Furthermore, Lo et al. [79] propose an Snort based IPS in the cloud with their study. Their system adopts the cooperation with other IPSs, to be a able to alert other systems of ongoing attacks.

- **Anomaly-based Systems**. Statistical anomaly detection, first normal traffic is gathered which is then used to compare it with incoming traffic. Statistical tests on the data can be applied to label the traffic as malicious or benign. A study by Khamruddin and Rupa [80] proposes an statistical anomaly detection technique using a Gaussian model to defend against DDoS attacks that target the HTTP protocol. Another study by [81] uses a statistical anomaly detection technique by calculating using a statistical method called the 'Jensen-Shannon divergence' method. Data mining techniques are also adopted by anomaly detection methods to help detecting attacks in environments where big data is involved. For example, Shamsolmoali and Zareapoor [81] uses Hadoop to be able to monitor network traffic with big volumes.
- **Hybrid Systems**. Some methods, use the techniques of both, signature and anomaly based techniques in a hybrid system. For example Bro is an example of a open-source hybrid system [82]. In hybrid solutions a signature-based detection method usually comes first in line, to be able to detect and filter all known attacks first. After that, an anomaly-based can detect missed DDoS attacks and try to filter this traffic. Teng et al. [83] propose an IPS where the first layer of defense works similar to other Snort systems. After passing this layer, the second layer uses statistical methods to detect anomalies in the network traffic.

For a more comprehensive overview of available methods, the study of Osanaiye et al. [6] can be reviewed, including all advantages and disadvantages of the methods. In general, the biggest disadvantage of signature-based detection are that unknown attacks cannot be detected and that it can be challenging to maintain an up-to-date signature database. However, the advantage of signature-based detection is that it has a low false positive rate. At the other side, anomaly-based detection has the advantage that is can detect new DDoS attacks as well. The downsides of anomaly-based detection generally include that traffic is classified less accurately and that it can be challenging to respond on an attack [84]. Hybrid detection methods can provide the advantages of both methods and have therefore the focus in newer studies. An advantage of a hybrid method is that signatures can be made from detected anomalies, that can directly be applied as rules in the signature-based methods. The limitation is that a hybrid system can also get complex and challenging to implement. A complex system can cause an overhead, resulting in the exhaustion of resources. This exhaustion means that monitoring and handling the network traffic can cause a denial of service on itself, which is the opposite of the DDoS IPS its goals.

2.2.2.4 Web Application Firewall (WAF)

Web Application Firewalls (WAFs) defend webservers against attacks targeting protocols in the application layer, such as HTTP, SMTP, DNS and NTP [85]. The general purpose of a WAF is to defend against attacks such as, SQL injections, session hijacking and cross-site scripting. However, another purpose of a WAF can be to detect and stop DDoS attacks [41]. Next some methods will be described on how WAFs are able to mitigate DDoS attacks.

- **Request analysis** can be used to decide whether a user request is part of a DDoS attack or not. Those requests must be part of a certain application layer protocol. For example, a widespread used protocol on webservers is HTTP, which is also a target for all sort DDoS attack types [41]. With WAFs such as Modsecurity [86] or Modevasive [87], HTTP request inspection can be done to detect those types of DDoS attacks [88]. Those WAF examples are open-source and can be configured to be part of an Apache server.
- The use of **CAPTCHAs** is commonly used on web servers to distinct normal users from bots by giving the user a certain challenge [89]. Those challenges can also be used to filter DDoS attack requests coming from bots [90]. The challenges given to users can include a only human-readable pictures, but challenges can also involve a certain human behaviour detection [91]. An example of performing human behaviour detection could be the analyses of mouse movements.

The advantage of WAFs is that specific DDoS attacks can be detected, while they are not detected on other places. However, the limitation is that not all types of DDoS attacks can be mitigated, since WAFs are only able to detect certain DDoS attack in the application layer. Also handling requests on an application level is not as fast as doing it on a network level. One way of overcoming this problem, is to move the filtering part to a network level. For example, Endraca [92] propose a method to filter packets with Iptables after the detection of a DDoS attack with a WAF.

2.2.2.5 Overview mitigation methods

The previous subsections elaborate on different existing methods to mitigate DDoS attacks. This section will highlight the differences between the methods including their advantages and disadvantages. The methods are described under one one of the following names; BGP routing, Network Firewall, IPS and IDS or WAF. Generally, as a DDoS attack occurs, it passes those mitigation methods in the same sequence. Ideally a DDoS attack is mitigated as fast as possible, since closer to the victim it becomes more likely that a network congestion takes place as resources become more scarce. That congestion can result in a denial of service on itself. However, the further away from its victim it is more challenging to accurately detect the malicious traffic. The first reason for this, is that the amount of traffic is less closer to the victim, which makes it easier to analyse. Secondly, network traffic can be inspected more deeply closer to the victim, which results in more available data about the malicious traffic. Especially, attacks that do not occur in high volumes, typically attacks hitting the application layer, are more easily to detect with methods that are able to inspect network traffic with a high granularity. Concluded can be that with the choice of a mitigation method, a trade-off will occur between the accuracy of the detection and likeliness of resource exhaustion.

As seen in subsection 2.1.4 from the previous section, the most common attack vector in DDoS attacks is 'UDP amplification', which is an attack vector that generates high volumes of malicious traffic. To mitigate those kind of attacks, the mitigation method should therefore be able to handle large volumes of network traffic. However, to mitigate all sorts of DDoS attacks, some mitigation methods should also be able to detect DDoS attacks with lower volumes of malicious traffic. The mitigation method categories as outlined in the previous subsections of this chapter can be found in Table 2.2.

TABLE 2.2: Mitigation methods

Method category	OSI L	Open-source tools	Advantages	Limitations
BGP routing	3,4	BGP Flowspec	High speed network traffic processing possible, closely to the source of an attack.	Low accuracy due to: -limitation on the number of rules -strict rule specification requirements. -relative high quantity of benign traffic -malcious traffic is distributed
Network Firewall	3,4	Iptables Nftables eBPF & XDP	Network firewalls are often already part of existing network configurations. Methods can be used as part of an IPS/IDS. eBPF & XDP offer the possibility to filter on arbitrary packet data.	Challenging to find and update efficient rules that are able to filter all DDoS attacks. New DDoS attacks are difficult to detect.
IPS/IDS	3,4,7	Snort Bro	Can have access to all sorts of resources on systems and networks. IPS and IDS systems are the best methods for anomaly-based detection.	Systems can become too complex.
WAF	7	Modsecurity ModEvasive	Easier detection of low volumetric attacks in the application layer.	Only a suitable solution against attacks targeting OSI layer 7.

Table 2.2 shows with column 'OSI L', what OSI layers the methods usually can access. As discussed, some open-source tools can be found back in the table as well as the advantages and the limitations of the methods. Note that BGP Flowspec and eBPF and XDP are relatively new open-source tools that all overcome some limitations of their former methods. BGP routing is bound to only specific routers and high level network routers at ISPs for example, whereas eBPF and XDP can in theory operate on any device with a Linux kernel. The latter makes it possible for eBPF and XDP to operate on more different locations than what is possible with BGP Flowspec. Another advantage of eBPF and XDP over BGP Flowspec is that filter rules can be more granular, with even the possibility to inspect packet data destined to all other OSI layers.

WAFs have the limitation that they are not able to mitigate all sorts of DDoS attacks as they only operate in the application application layer. However in collaboration with other systems, WAFs can be an effective mitigation methods.

With the use of an IPS, anomaly-based detection methods can be used, which can be an advantage over signature-based only methods. Signature-based can only be effective if correct rules are maintained and applied, which exactly the main challenge for new methods using eBPF and XDP.

2.2.3 DDoS Protection Services

The previous section elaborated on what kind of mitigation methods one can use to protect itself against DDoS attacks. The defender can choose to implement those methods itself, but it can also choose to outsource it to a cloud-based DDoS Protection Service (DPS) [93]. This section will elaborate on those DPSs, how they operate and what the biggest DPS vendors are.

DPSs are commercial online services that offer protection against DDoS attacks. This protection is realised by implementing mechanisms against different kinds of DDoS attacks and also by adopting multiple techniques. Those techniques include partly the methods of the previous section, but also include the use of Content Delivery Networks (CDNs). A CDN is a distributed network network of servers and data centers, with the goal of delivering high availability and performance to its end-users [94]. Due to this distributed nature of a CDN it can be used divert internet traffic. This traffic diversion can balance loads and help to defend against DDoS attacks. For example, the DNS protocol or the BGP protocol can be used to redirect and divert the internet traffic, after which malicious traffic can be filtered. DPSs combine the features of CDNs and other mitigation methods to be able to mitigate DDoS attacks for their clients. Since, DPSs are commercial parties they typically do not fully disclose how their methods exactly work, but some of them publish how their methods work to some extend.

According to a IDC MarketScape assessment from 2019, Cloudflare, Akamai, Radware and Arbor Networks are all examples of leader vendors in their field [95]. Those vendors all have in common that they use CDNs and its features to mitigate DDoS attacks. Furthermore, all DPSs use a layered system using different sort of firewalls and scrubbing centers [96]. They also all stress the need for good collaboration amongst all sorts of DDoS mitigation methods, also including ISP level solutions. Akamai claims to serve between 15% and 30% of the global web traffic and offers a DDos mitigation service called Kona [97]. They deployed mitigation methods in the application layer as wel as in the network layer with network firewalls and WAFs. Arborer Networks offers a DPS with underlying systems called Pravail APS, Peakflow SP and Peakflow TMS [96].

Cloudflare implemented an automatic system called Gatebot [98] that analyses sampled incoming traffic from all their CDN edge servers. The central Gatebot is able to detect DDoS attacks and deploys mitigation rules on to the edge servers using L4Drop. On every edge-servers a network firewall is running which uses eBPF and XDP in order to mitigate DDoS attacks [8, 99]. Before dropping the malicous packet a small sample is send via Sflow packets back to Gatebot to keep track of the dropped traffic. Besides filtering network traffic, Cloudflare uses WAFs on their edge servers to protect against web application attacks. With their 'I'm under attack mode' [100], Cloudflare tries to mitigate DDoS attacks targeting the application layer. By using Javascript and cookies, Cloudflare checks if the traffic is coming from a legitimate user or a bot.

Concluded can be that DPSs can be helfpul for protecting online services against DDoS attacks. Especially due to the used CDNs and easy deployment, DPSs can have an advantages over self-implemented mitigation methods. Cloudflare publishes more technical articles about their DDoS mitigation methods than the other leaders in the market. From multiple blogposts it becomes clear that Cloudflare embraces the use of eBPF and XDP within their DPS.

2.3 eBPF and XDP

In the previous section we investigated how DDoS attacks are currently mitigated. We observed that signature-based solutions have a lower false positive rate than anomaly-based *solutions, which indicates that benign traffic is less often classified as malicious with signature based methods. We learned that eBPF and XDP form relatively new methods to mitigate DDoS attacks. Those methods are able to make a signature-based DDoS mitigation that shows packet dropping speeds that are higher then other available methods. Besides the speed, eBPF and XDP also have some features that offer the possibility to communicate with user-space. Furthermore, with eBPF and XDP, full packet inspection is possible. This section aims to study all available materials on eBPF and XDP.*

2.3.1 extended Berkeley Packet Filter (eBPF)

This section will elaborate on how eBPF works and how it interacts with other systems. First will be described how its predecessor, the Berkeley Packet Filter (BPF) works. BPF is a virtual machine model that runs inside the Linux kernel. It was designed for filtering and capturing network packets matching specific rules. It was introduced by Steve McCanne and Van Jacobson in 1992 [71] and it came with the ability to run programs inside the kernel of a Linux operating sytem. One well-known program that is able to capture network packets using BPF is tcpdump. After compiling tcpdump turns a readable expression into an BPF bytecode, which contains a set of instructions. This expression forms a rule for the BPF filter. A rule, for example, could be that all incoming UDP packets at a certain port and coming from a certain IP address have to be filtered. A rule can be broken down in all separate parts as IP addresses and ports, these are called parameters.

A rule can be created for a tcpdump expression that filters all outgoing packets with a certain destination IP address. In the next example the rule contains only one parameter which is 192.168.0.1 for filtering all IPv4 packets with this IP address. Listing 2.1 gives the set of instructions that represent the BPF bytecode of the rule.

LISTING 2.1: Tcpdump instructions

```
(000) ldh     [12]
(001) jeq     #0x800          jt 2     jf 5
(002) ld      [30]
(003) jeq     #0xc0a80001     jt 4     jf 5
(004) ret     #262144
(005) ret     #0
```

More details on the syntax and meanings of the code in listing 2.1 can be found at [101]. The first instruction (000) loads 2 bytes from the packet with offset 12 into memory. The second instruction (001) checks if the IPv4 protocol is used. If the latter is true the code jumps to instruction (002) and otherwise to (005). The third instruction (002) loads the the destination address. Then the next instruction does the actual check if this is destination address 192.168.0.1 and returns true at instruction (004) or false at the last instruction (005). Note that this filter exactly costs six instructions, for a rather simple rule with not so many parameters. A more complicated rule with more parameters will require a lot more instructions.

Alexei Starovoitov came up with a new BPF architecture BPF JIT in 2014 [70]. This is now known as the extended BPF (eBPF) and sometimes called just BPF as it replaced the classic BPF. The classic BPF is often named as cBPF in literature, but in this paper the the two names we will use are BPF for the old one and eBPF for the new one. eBPF is more efficient then the BPF thanks to this just in time (JIT) compiling of the eBPF code. eBPF is not only designed to filter network packets, but to process any general event inside the kernel. Furthermore eBPF comes with maps that make it possible to share data between applications in the user-space and kernel-space. For example, an application like an internet browser which can directly interact with the user, runs in the user-space, whereas the driver of a computer screen runs inside the operating system. Those maps are not restricted in size. The execution of eBPF code is hardware independent which makes it possible to run on any architecture.

The shown tcpdump example was just an example of a program that in this case uses the former BPF. The creation of an eBPF program with the purpose of eventually running inside the eBPF VM, starts typically with the creation of 'C source code'. Figure 2.3 shows this step in the up left corner at '1'. The 'LLVM Clang compiler' [102] is able to compile this code into the 'eBPF bytecode' in the form of an ELF file [103]. After this compilation the bytecode can be loaded into the kernel with a system call (syscall). A syscall is the typical way for a program in userspace to request a service from the kernel of the operating system. An eBPF program loaded into the kernel is done via the 'bpf syscall' [104]. Together with this syscall, maps can be initiated and the program type has to be specified. This program type essentially defines which kernel functions can be called from the eBPF program. As can be seen in Figure 2.3 the eBPF bytecode reaches the 'verifier'. Inside the kernel the eBPF verifier comes with some checks on the byte code to ensure some safety, for example no backward jumps are allowed to make sure no loops are possible. Furthermore an eBPF program can not have more than 4096 instructions and each instruction has a length of 64 bits. With the fact that simple filter rules can already cost multiple instructions as it is the case with tcpdump, it means that the bytecode can only contain a very limited amount of filtering rules. If finally the bytecode is considered safe by the verifier, the 'JIT' compiler compiles the bytecode into machine code that matches the right architecture. Finally the eBPF program runs inside the kernel where it can interact with other kernel functions and can also interact with userspace programs via 'maps'.

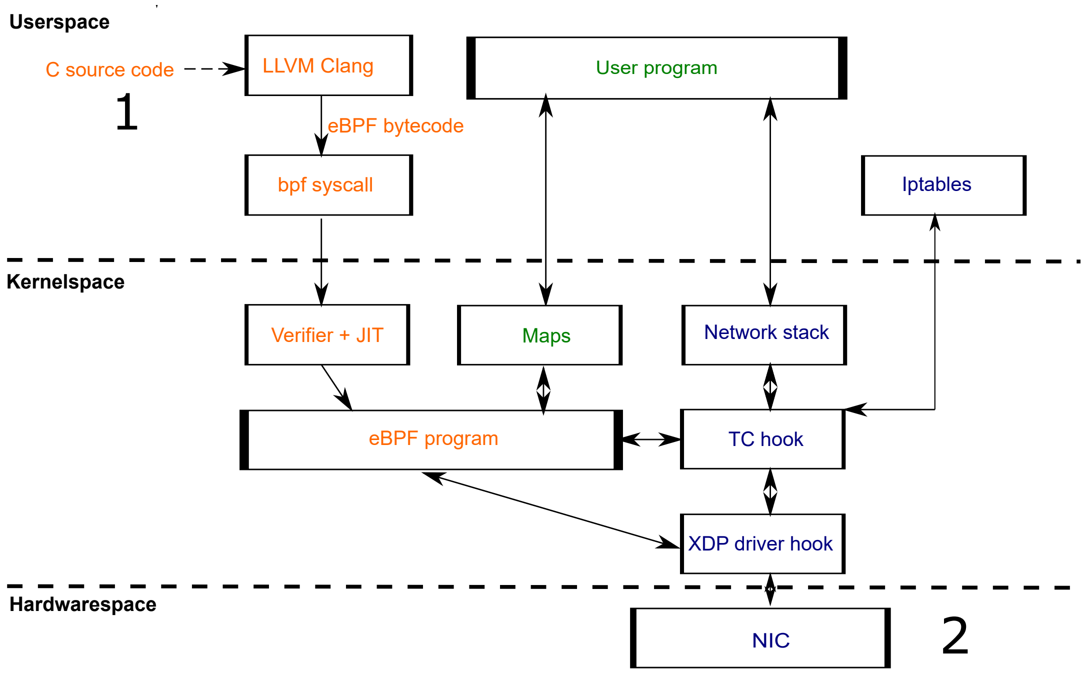

FIGURE 2.3: eBPF & XDP environment

2.3.2 eXpress Data Path (XDP)

This section will elaborate on XDP and how it is possible that it enables high packet processing speeds. To reach the userspace, normally a network packet coming from the Network Interface Card (NIC) flows through the kernel of an operating system. Figure 2.3 shows this NIC at the '2', after which an incoming packet will flow through all the elements inside the kernel. An alternative way is Kernel bypassing; a method to process those packets directly in user-space by skipping the kernel. The main advantage of doing this, can be that a high-speed packet processing performance is possible. Some toolkits already use a form of kernel bypassing to gain performance, such as the Data Plane Development Kit (DPDK) [105], Cisco's Vector Packet Processing (VPP) [106], Netmap [107] and Snabb [108]. Bypassing the kernel to gain speed has some disadvantages. All networking functionality normally provided by the kernel has become useless. There is also almost no ability to interact with other parts of the OS and the user-space application needs to manage their hardware resources directly. This all results in increased system complexity which could introduce some security problems. XDP is a solution that does basically the opposite of kernel bypassing. It actually adds programmability with eBPF programs directly at the operating system networking stack [10]. This is done by adding a new layer in the kernel network stack. An eBPF program can be directly executed at the NIC driver at the first possible moment after a packet is received. The latter is called the the 'XDP driver hook' as shown in Figure 2.3. As explained is the previous section, the bpf syscall allows eBPF bytecode to be executed as an 'eBPF program'. A declaration inside the syscall has to be given to specify that the 'XDP driver hook' should be used and the eBPF program therefore can also be called a 'XDP program'.

In the fight against DDoS attacks, an import feature of a network filter is that it should drop malicious packets as fast as possible. According to the performance evaluation from Høiland-Jørgensen et al. [10], the packet dropping performance of XDP is significantly faster then older methods, such as Iptables. This performance advantage of XDP over Iptables can be explained by the fact that XDP has the possibility to filter right at the network device driver, which is an earlier stage then where Iptables does its job. As shown in Figure 2.3, the fastest way for Iptables to drop packets is in the prerouting phase near the traffic control (TC) hook. At the TC, the incoming packets already have passed the driver space. This 'TC hook' however will come after the 'XDP driver hook' and is therefore also a lot slower. According to Høiland-Jørgensen et al. [10] packets an the 'XDP driver hook can be dropped with a speed of 24 Mpps on a single CPU against a dropping rate of 4.8 Mpps at the TC hook. This shows that packets can be dropped significantly faster with the use of XDP than former methods using Iptables. The speed improvement of XDP over Iptables is in the case of Cloudflare, one of the reasons for the replacement of an Iptables filter [7]. With Cloudflare's use of NICs that can handle two times the speed of 25Gbps in their edge-servers, the network links themselves are not the bottleneck when it comes to handling packets [109].

2.3.3 Related Work

This section aims on getting insight in all the available scientific work on eBPF and XDP related to the mitigation of DDoS attacks. Sections 2.3.3.1 and 2.3.3.2 give insight in possible future research directions based on the found studies on eBPF and XDP. In order to be sure no scientific materials have been missed, a Google Scholar crawler has been written [110]. First of all, the search terms 'eBPF', 'XDP' and also the full written terms 'extend Berkeley Packet Filter' and 'eXpress Data Path' have been used. Those searches resulted in up over thousand result for only 'eBPF' and just below hundred results for the combination with 'XDP'. Table 2.3 shows the coverage of the 25 most important papers found.

TABLE 2.3: Studies overview

Reference	**Paper Topics**			
	DDoS	**eBPF**	**XDP**	**Rule Generation**
Tran and Bonaventure [111]		✓		
Hong et al. [112]		✓		
Massimo [113]		✓	✓	
Bertrone et al. [114]		✓	✓	
Bertrone et al. [115]		✓	✓	
Hemminger [116]		✓	✓	
Ahern [117]		✓	✓	
Miano et al. [118]		✓	✓	
Lakhani and Miller [119]		✓	✓	
Rüth et al. [120]		✓	✓	
Chaignon et al. [121]		✓	✓	
Van Tu et al. [122]		✓	✓	
Xhonneux et al. [123]		✓	✓	
Tu and Ruffy [124]		✓	✓	
Viljoen and Kicinski [125]		✓	✓	
Brouer [126]	✓	✓	✓	
Bertin [7]	✓	✓	✓	
Fabre [127]	✓	✓	✓	
Høiland-Jørgensen et al. [10]	✓	✓	✓	
Miano et al. [128]	✓	✓	✓	
Miano et al. [129]	✓	✓	✓	
Scholz et al. [72]	✓	✓	✓	
Khamruddin and Rupa [80]	✓			✓
Aljuhani et al. [130]	✓			✓
Makita [131]	✓			✓

First of all, several studies found via Google Scholar did mention the use of XDP and eBPF in combination with DDoS mitigation, but the main topics of those studies involve other topics that are not relevant for this study [132, 133, 134].

Furthermore some studies found are are about different use-cases for eBPF. Apart from packet filtering against DDoS attacks, eBPF offers solutions to other problems inside to Linux kernel. For example Tran and Bonaventure [111], propose an extension on the TCP stack inside the linux kernel. Another use of eBPF is the proposal of a virtual network switch by Hong et al. [112].

Another category of studies found, are studies that are about eBPF and XDP, but with no focus on the mitigation of DDoS attacks. For instance purely about the replacement of Iptables by eBPF and XDP in general [113, 114, 115] or studies about a general eBPF and XDP solution with no focus on DDoS attacks [116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126]. Miano et al. [118] for instance, discusses on all sorts of network functionalities that can be built with eBPF and XDP, such as routers, NATs and bridges.

The third category of studies focus on the use of eBPF and XDP related to the mitigation of DDoS attacks. From 2017, the study of Bertin [7] is an explorative study for Cloudflare that concludes that XDP is a promising technology for the mitigation DDoS attacks. Two years later in 2019 is now clear that Cloudflare successfully adopted XDP. In 2019, Fabre [127] describes for Cloudflare how they adopted XDP by hard-coding filter rules inside the XDP program. The paper concludes that they are successfully able to mitigate DDoS attacks, but are still constrained by the number of rules that can be specified. Høiland-Jørgensen et al. [10] tests the capability of eBPF and XDP to drop packets at high speeds. They show a raw packet processing performance of up to 24 Mpps on a single CPU. The paper of Miano et al. [129], specify 'bpf-iptables' based on eBPF and XDP, which is able to replace the former Iptables. They tested their bpf-iptables performance under simulated DDoS attacks. Further, they achieved higher packet dropping rates than Iptables with the use of eBPF and XDP. Their matching was done on IP source, transport layer protocol and source ports. In another study, Miano et al. [128] shows the same performance advantage of using eBPF and XDP over Iptables.

No studies have been found in how filtering rules should be generated in order to mitigate DDoS attacks with XDP and eBPF. Due to the unique behaviour of the eBPF environment, studies into rule generation methods can be helpful in order to optimize DDoS attack mitigation. Existing studies such as [80, 130, 131] can be used to find similiar methods for the use with eBPF and XDP.

2.3.3.1 Number of rules restriction

With eBPF and XDP together, it is possible to fit filter rules into an single XDP program. There is however a limitation in the amount of rules that be used in a XDP program, since XDP program can not be larger then 4096 instructions. As can been seen in subsection 2.3.1 the number of instructions needed can add up pretty quickly if the number of rules are increasing, especially when the rules become more complex. Scholz et al. [72] managed to create a efficient filter containing up to 100 rules. Facebook's XDP packet filtering solution uses eBPF maps to store rules [9]. In theory those maps can be infinitely large which means that there is much more space for specifying rules. An extra advantage is that the XDP program doesn't need to be recompiled to add or remove rules. However, Fabre [127] reports that looking up certain rules in maps also comes with the minimal costs of 4 instructions. Facebook doesn't show how their use of maps affects their performance results and only little implementation details are given by Facebook.

2.3.3.2 Packet filter algorithms

The way packets are filtered by rules can be defined by an algorithm. Those algorithms can define how packets are matched against certain rule-sets [135]. 'Linear search' is the method Iptables uses that matches incoming packets rule for rule against certain properties of the packet. As defined by Varghese [135], this packet matching can also be done via 'Cross-Producting' or decision tree approaches for example. However, not all those algorithms are suitable to be used in a DDoS mitigation solution with eBPF and XDP [129]. Miano et al. [129] uses the 'Linear Bit-Vector Search' (LBVS) algorithm with eBPF and XDP in order to match packets against their rules. In a presentation, the same authors suggest that Tuple-Merge [136] might also work efficiently, but this is still open for research.

2.4 Conluding Remarks

The goal of this chapter is to answer the research question; **What methods currently exist that automatically update rules in a signature based DDoS mitigation?** The answer on this question has been realised by first looking into DDoS attacks in general in section 2.1. Then in section 2.2 the field of DDoS detection and mitigation has been explored. Finally section 2.3, elaborates on the related work, the advantages and the limitations of using eBPF and XDP as a DDoS attack mitigation method.

The second section 2.1 gives a brief overview of what to most important aspects of a DDoS attack are. It explained that victims of DDoS attacks are targeted to exhaust their resources. By exhausting the resources of a victim, it will not be able to deliver a service to other legitimate users of the service. Those resources get exhausted by flooding the victim with network traffic or by misusing the behaviour of a certain protocol to cause the exhaustion of the victim its resources. The conclusion is that as long as resources are limited, they will always be prone to exhaustion and thus vulnerable for a denial of service. Further observation is that DDoS attacks still form a threat with huge revenues losses and reputation damage, due to 5 different attack motivations. At the end section 2.1 elaborates on the distributed nature of DDoS attacks. Nowadays typical DDoS attacks are performed by built botnets existing out bots and C&C servers, where the bots are compromised machines. Depending on the attack vector, an extra layer in to a DDoS architecture is added with reflectors. Most common attack vector is UDP amplification through those reflectors and the 7 most misused protocols are given.

Section 2.2 starts with describing different kind of defense mechanisms. Argued is that reactive mechanisms will always be needed against DDoS attacks and that they can are ideally deployed on multiple collaborative locations. Detection mechanisms of DDoS attacks can be classified into signature-based and anomaly-based, with the main challenge for signature-based methods to maintain an up to date ruleset. However signature-based mechanisms, have a lower false positive rate. Further it describes different mitigation methods that are categorised by four different categories; BGP routing, Network Firewalls, IPSs and WAFs. Concluded was that the relatively new open-source tools eBPF and XDP have some features that could overcome the limitations of other methods. These tools can be deployed on a wide variety of devices and enable the possibility to collaborate with other applications via user-space. This collaboration can overcome the limitation that signature-based mechanisms have, since it enables the possibility to dynamically update filter rules, that are automatically generated by secondary program.

Furthermore eBPF and XDP are able to fully inspect incoming packets at a even higher processing rate than former network firewall methods. Finally section 2.2, describes is that DPSs are a popular way to outsource the defense against DDoS attacks. Those DPSs use CDNs to be able to redirect internet traffic, which can be used to mitigate DDoS attacks. Implementation details on further methods are generally not public, but Cloudflare posts openly about their successful method using eBPF and XDP.

In section 2.3, the capabilities of eBPF and XDP in order to mitigate DDoS attacks are shown. With the ability of eBPF to communicate with all sorts of functions inside the Linux kernel and user-space, it enables to possibility to create a dynamic and customizable solution. The XDP hook inside the Linux kernel is able to process packets on high speeds due to the fact that packets are filtered close to the NIC. Related studies show that no studies use eBPF and XDP in order to automatically update and generate rules with the use of attack signatures.

Chapter 3

Method & Design

In the previous chapter we saw what DDoS attacks are, how they work and what mitigation methods exist to be able to defend against them. We concluded that the features of eBPF & XDP can potentially help mitigating DDoS attacks. These features include the possibility to dynamically update filter rules from userspace into kernelspace and also the possibility to drop packets at faster speeds than former methods. With eBPF and XDP as tools, this chapter its aim is to study how those tools can fit in a DDoS mitigation system. Therefore, the goal of this chapter is to find an answer on ***how to automatically generate eBPF rules for DDoS mitigation using DDoS attack signatures (RQ2)****.*

In ***section 3.1*** *we will give the predefined requirements of the system we created, by taking RQ2 into account. Then, in* ***section 3.2*** *we will elaborate on the implementation of our own system and on how the requirements of the system are met. After that,* ***section 3.3*** *will describe how the effectiveness and performance of the designed system is going to be measured.*

3.1 Requirements

The requirements for the DDoS mitigation system of this study are specified in this section. We first take a look at how our built DDoS attack mitigation system is part of a bigger system. How this bigger system will look like on a high level has been shown in Figure 3.1. At the left side of the figure, normal traffic as well as attack traffic flows into the system. At the right side of the figure the filtered traffic flows out of the system and should ideally only exist out of normal traffic.

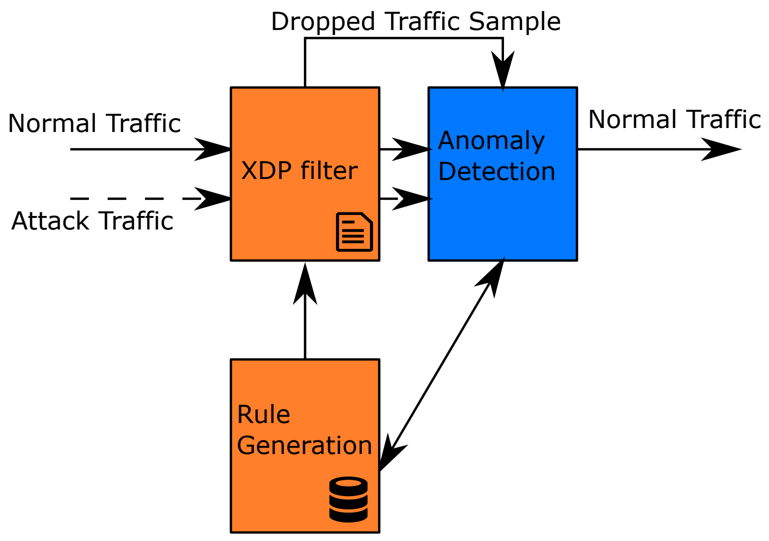

FIGURE 3.1: Design of DDoS Attack mitigation system

The parts in orange, will be a part in the system that is created for this study. The blue anomaly detection part the traffic is **NOT** part of this study, therefore further requirements apply on the orange part. To take full advantage of the characteristic of eBPF and XDP, we choose that our system should be part of a bigger collaborative system. The incoming traffic will first pass the XDP filter that is able to apply certain filter rules on the traffic. The traffic will than flow trough an anomaly detection system which should be able to detect new attacks. By dissecting the traffic the anomaly detection system should also be able to compare the traffic with already known attacks. As soon as the anomaly detection has warned the rule generation system with a certain incoming attack, the rule generation system can create rules to be applied inside the XDP filter. This way we have a hybrid mitigation system that uses both signature-based and anomaly-based methods to mitigate DDoS attacks. Within the system the XDP filter is able to sample parts of dropped traffic. From those samples can be derived how many and what kind of packets are being dropped. With that information can be approximated how big the attack is and how much normal traffic has been dropped.

The goal of the system is to drop DDoS Attack traffic by automatically generated XDP rules that filter the traffic. In a realistic scenario the system will be operated by a network operator, since the system will be part of a certain network. Those networks can differ in several ways and will therefore have different specific requirements when it comes to the mitigation of attacks. The network operator will have to asses what the critical parts of his network are and in what ways a DDoS attack can occur, since this will be different for every network. Therefore the network operator should be able to manually alter network traffic filter rules that are created by any automatic system. That's why the system can be operated from a network operator's perspective. To create a realistic system which is operated by a network operator, the requirements of the system take into account that the system can be part of different networks and different attack scenarios. Those factors are considered to be the context of the actual DDoS mitigation system. The requirements are grouped by 3 main requirements, that all include several sub-requirements in the following way.

1. The system should be able to filter network traffic using eBPF and XDP.
   - 1.1 The system should be able to load eBPF programs inside the Linux kernel.
   - 1.2 The eBPF programs should have a XDP hook that is able to filter network packets.
   - 1.3 The system should be able to store filter rules in eBPF maps.
   - 1.4 The system should be able to store hard-coded rules inside the eBPF program.
2. The system should be able to generate XDP rules in order to drop packets from DDoS attacks.
   - 2.1 The system should be able to derive filter rules from DDoS attack fingerprints.
   - 2.2 The system should be able to filter on all different features in OSI layers 3 and 4.
   - 2.3 The system should be able to filter on important features inside OSI layers higher than 4.
   - 2.4 The system should be able to filter on multiple features at once.
3. The system should be able to mitigate DDoS attacks in a realistic situation.
   - 3.1 The system should be able to take its own context into account when generating XDP rules.
   - 3.2 The network operator of the system should be able to manually change the type of rules that are used in the XDP filter.
   - 3.3 The network operator of the system should be able to see what the kind of packets are being dropped and how many of them are being dropped.

The first requirement directly specifies the use eBPF and XDP. The use of those tools are justified by chapter 2. The second requirement is set in order to drop packets from DDoS attacks specifically. The system should be able filter based on specific DDoS attack properties. As learned from chapter 2, DDoS attacks can have a variety of different features that describe the specific DDoS attack. Besides being able to mitigate different sorts of DDoS attacks, the third requirement specifies that the system should also be able to work in a realistic network environment. With the term realistic is meant that there are many variables in the context of the system that can influence its behaviour.

Note that the requirements not only specify main and sub-requirements, but are made on top of each other. The first requirement should be met in order to realise the second one and the second requirement should be implemented to be able to implement the third requirement. The next section will describe the implementation of the system and how the requirements are met.

3.2 Implementation

By following the requirements from the previous section, this section will describe how the DDoS mitigation system has been implemented for this study. All code used for the implementation can be found in an open source Github repository [137].

3.2.1 eBPF, XDP and the IOvisor BPF Compiler Collection

As known from section 2.3 from the previous chapter, eBPF programs can be deployed by first compiling C source code by LLVM Clang and then using a bpf syscall to load the program run into kernelspace. This process can also be done by IOvisor's BPF Compiler Collection (BCC) [138], which is a project that includes tools to compile and load those eBPF programs. It uses a C wrapper around LLVM and provides a front-end in Python which makes it easy to makes applications in user-space that are also able to make eBPF programs. The system implemented for this study uses IOvisor's BCC, which makes it possible to deploy eBPF programs that use a XDP filter. The filter is able to get values out of maps, but can also be hard-coded inside the program. The Python front-end of BCC has been integrated within the rest of the system. With the use of BCC, requirement number 1 and its sub-requirements are all met.

3.2.2 Rule Generation

The second requirement of the requirements in section 3.1 describes that the system should be able to generate XDP filter rules that are able to drop DDoS attack traffic. The third requirements describes that the system should be aware of its context and that the system should manually be able to be operated by a network operator. For both requirements, 2 and 3 is described how they are met in this section.

The rule generation algorithm uses DDoS attack fingerprints as an input. Each fingerprint functions as a summary of a specific attack. A fingerprint can also be called the signature of the attack. The fingerprint of each DDoS attack contains the values of all available features inside the DDoS attack. Inside the fingerprints, the values are given as sets of values for each feature that are present in the DDoS attack. A feature is a certain field inside a network packet e.g. the source IP address in the IP layer. Note that the way the fingerprints are build up, the values are not linked to each other, so not known is what the value combinations are existing inside the attack. Those features values can be used to drop the DDoS attack traffic by the XDP filter. The rule generation algorithm for this study assumes that the DDoS attack fingerprints used are 100% correct, Algorithm 1 is a general example of how all features in a fingerprint are used to create rules for the XDP filter.

Algorithm 1: General XDP rule generation & filter algorithm

```
   Input: DDoS_fingerprint<k,v[]>
   // k is the unique key of a specific feature and v[] is a list
      with the values for that feature
   Input: feature_whitelist[]
   Input: feature_value_whitelist[]

 1 initialize XDP filter // load eBPF program and attach XDP filter
 2 feature_map<k,v[]> ← NULL
 3 foreach k, v[] in DDoS_fingerprint<k,v[]> do
 4     if k in feature_whitelist[] then
 5         continue
 6     foreach v in v[] do
 7         if v in feature_value_whitelist[] then
 8             continue
 9         else
10             feature_map<k,v[]> ← <k,v> // eBPF map, shared between
                  userspace and kernelspace
11         end
12     end
13 end
14 while XDP filter is loaded do
15     packet ← incoming packet
16     foreach value in packet do
17         if value in feature_map<k,v[]> then
18             pass_packet ← False
19             return pass_packet // DDoS packets being dropped as long as
                  XDP filter is loaded
20         else
21             pass_packet ← True
22             return pass_packet
23         end
24     end
25 end
```

The algorithm in algorithm 1 shows how from a DDoS attack fingerprint, XDP filter rules are created. The input of the algorithm exists out of the DDoS attack fingerprints, a set of features that should be white-listed and a set of values per feature that should be white-listed. With those lists a network operator can both choose to deny a filter on a specific feature or to deny a filter on a specific value of a feature. For example a network operator could choose that the XDP filter should never drop packets based on a source IP address. Another example could be that network operator specifically chooses to not filter on UDP source port 53. On line number 3 and 6 the checks take place whether a feature or certain feature value should be added to the feature_map. The XDP filter reads each incoming packet and checks all values of the packet against the values in concerning feature_map. If a value exist in one of those maps the packet will be dropped.

In the example algorithm, except from the white-listed ones, all features and their values will be used to filter packets. However, not in all cases, all values are needed to still be able to effectively drop the packets of a DDoS attack. To optimally minimize the features and its values that are used in the filter, the challenge is to find the information inside malicious DDoS attack packets which is unique for those packets and is not existing inside the normal packets. This unique information can exists out of multiple network packets features. Only the features that should be filtered should not be added to the white-list. The selection of those features can either be done automatically or manually by the network operator. There are several ways to select the features automatically. First method is to select the feature with the fewest values inside the fingerprint as given in algorithm 2.

Algorithm 2: Select feature based on fewest values

```
Input: DDoS_fingerprint<k,v[]>
Output: feature_whitelist
1  $best\_feature \leftarrow NULL$
2  $smallest\_count \leftarrow -1$
3  $feature\_whitelist \leftarrow []$
4  foreach k,v[] DDoS_fingerprint<k,v[]> do
5      $count \leftarrow 0$
6      foreach v in v[] do
7          $count \leftarrow count + 1$
8      end
9      if $count < smallest\_count$ or $smallest\_count == -1$ then
10         $best\_feature \leftarrow k$
11         $smallest\_count \leftarrow count$
12 end
13 foreach k in DDoS_fingerprint<k,v[]> do
14     if k is not $best\_feature$ then
15         $feature\_whitelist \leftarrow k$
16 end
17 return $feature\_whitelist$
```

Algorithm 2 start with a DDoS attack fingerprint and returns the best feature based on the amount of values per feature. It returns the feature as a white-list which can be used in algorithm 1 to generate XDP rules. In anomaly detection algorithms often feature selection techniques are used to determine what the most effective features are to train the algorithm. Adjusted mutual information is an information-theoretic metric which can be used as such a feature selection technique [139, 140, 141]. For this study, the adjusted mutual information (AMI) score metric has been used to determine which features inside the attack traces and fingerprints are best in distinguishing the attack traffic from normal traffic. Assuming that the AMI scores have been pre-calculated and are inside the fingerprints, the algorithm to select the best feature based on those scores, has been given in algorithm 3.

Algorithm 3: Select feature based on AMI score

```
Input: DDoS_fingerprint<k,v[]>
Output: feature_whitelist
1  $best\_feature \leftarrow NULL$
2  $best\_AMI\_score \leftarrow 0$
3  $feature\_whitelist \leftarrow []$
4  foreach k in DDoS_fingerprint<k,v[]> do
5      $score \leftarrow k(AMI\_score)$
       // AMI_score is additional information per feature inside the
          fingerprint
6      if $score > best\_AMI\_score$ then
7          $best\_AMI\_score \leftarrow score$
8          $best\_feature \leftarrow feature$
9  end
10 foreach k in DDoS_fingerprint<k,v[]> do
11     if k is not best_feature then
12         $feature\_whitelist \leftarrow k$
13 end
14 return $feature\_whitelist$
```

Algorithm 3, again starts with the fingerprint and returns the list with features that should be white-listed. The AMI scores in the fingerprints are linked to the concerning features. With the possibility of the system of generating XDP rules that are able to drop packets on specific features of a DDoS attack, the second requirement has been met. The third requirement has been met by the possibility to alter the generated rules based on the selected features and values. Selecting specific features and values can not only be done to optimise the algorithm, but also to respond to situations in the context of the system. The latter necessary to be able to mitigate DDoS attacks in realistic situations.

3.2.3 Data Sources

The data that has been used to test the built system will be discussed in this subsection. The sources of the data and how the data has been ready for this study will also be discussed. The data used in this study is realistic data from real networks and real different sorts of DDoS attacks. This applies on the normal traffic traces, attack traffic traces and the corresponding fingerprints of those attack traffic traces. Santanna [11] created a platform called DDoSDB on which known attacks can be shared. The platform offers a database that is used for this study. The database exists out of 889 DDoS attack traces together with the corresponding fingerprints, that contain the most important information about the attack. The attack traffic comes in PCAP files and the fingerprints are formatted in JSON files. The available information in the fingerprint is depended on what protocols are used. In Listing 3.1 an example of a fingerprint is given of an attack with UDP traffic.

LISTING 3.1: UDP DDoS attack fingerprint

```
{
    "key":"d8ecc4556c6300355248ce2df77c49d3",
    "multivector_key":"420ebbc94e5ec5708143ac8774c08d92",
    "protocol":"UDP",
    "src_ips":[
      {
        "as":"23969",
        "cc":"TH",
        "ip":"125.27.146.179"
      },
      {
        "as":"23969",
        "cc":"TH",
        "ip":"125.26.151.154"
      }
   ],
   "src_ports":[
      24445
   ],
   "dst_ports":[
      31280,
      62696,
      41102,
    ...
   ],
}
```

As can be seen in Listing 3.1 in all the fingerprint summarizes some information available in the attack trace. All source IP addresses are listed as well as all the used ource and destination ports. In this particular case the the attack purely exists out of UDP packets with 24445 used as the source port. The out of the box fingerprints from DDoSDB only contain features up to a certain level. Since the fingerprints come from DDoSDB, note that the fingerprints are not made by this study. **The fingerprints from DDoSDB that are used for this study are assumed to be 100% correct.** Extra information to the fingerprints has been added that contains more in depth packet information about the DDoS attack. The DDoS attack traces have been compared the normal traffic and for each feature inside the attack traces the AMI has been calculated.

Out of all the 889 attack traces, some of the pcap files mistakenly either didn't contain any packets at all or contained attack vectors with multiple L4 protocols. After filtering all those bad attack traces, a number of 702 attack traces in total were left over. Table 3.1 shows the number of attacks per protocol.

TABLE 3.1: Number of attacks per protocol

OSI L4 Protocol	**#**	**OSI L7 Protocol**	**#**
TCP	278	HTTP	4
UDP	253	DNS	45
		NTP	19
		QUIC	9
		Chargen	7
		SSDP	7
		Other	50
ICMP	170	-	-

None of the attack traces from DDoSDB contained IPv6 packages. Further, in table 3.1 can be seen that the OSI L4 protocols TCP, UDP and ICMP are used in respectively 278, 253 and 170 of the attack traces. Every attack only exists out of one of those protocols. Some attack traces also contain packets with higher OSI level protocols as shown in the table.

To simulate normal traffic for this study, also realistic traffic has been used. The used "bigFlows" [142] network trace is a capture of real network traffic on a private network. This capture contains exactly 791615 network packets and 132 different applications are used. Those kind of network traces can be used to simulate the network traffic in a real network. The bigFlows network trace comes as an example network trace, with a tool that can be used for replaying network traces. This tool is called Tcpreplay [143]. Table 3.2 shows the statistics on what protocols are used in the capture.

TABLE 3.2: Statistics of bigFlows.pcap

OSI Layer	**Protocol**	**Number of packets**	**Percentage**
L3	IPv4	791179	99.9%
	IPv6	436	0.1%
L4	TCP	635017	80.1%
	UDP	153135	19.3%
	ICMP	3434	0.4%
L7	HTTP	28582	3.6%
	DNS	4466	0.6%

With 791179 out of the total 791615 packets, table 3.2 shows that the capture mainly contains IPv4 packets. Furthermore, the majority of the packets are TCP packets with a percentage of 80.1%. Most of the packets do not contain application data in OSI layer 7, but the protocols HTTP and DNS have a share of respectively 3.6% and 0.6% of the total amount of packets.

3.2.4 Setup

The test setup of this study contains of two workstations connected trough an UTP cable. The theoretic speed of both network interfaces is 1 Gbits per second and the throughput in practice is 950 Mbits per second as can been seen in Appendix A. Figure 3.2 shows the sequence of actions taken to test a given set of DDoS attacks against the XDP filter.

The sequence in Figure 3.2 starts with an attacklist of keys that represent certain attacks from the DDoSDB database. PC1 initiates an ssh connection with PC2 and from there on an iteration takes place over each attack in the attack list. PC1 then notifies PC2 on what attack is going to be tested together with the number of runs the specific is going to be tested. Next, PC2 prepares pcaps containing the DDoS attack traffic and normal network traffic as discussed in subsection 3.2.3. When the preparation of the pcaps on PC2 is done, PC1 attaches the XDP filter for a specific rule-set and start recording the network traffic with Tcpdump. Tcpdump [144] is a tool that is able to capture network traffic and is able to store a network trace file. In the meanwhile PC2 is able to send the prepared pcaps using Tcpreplay.

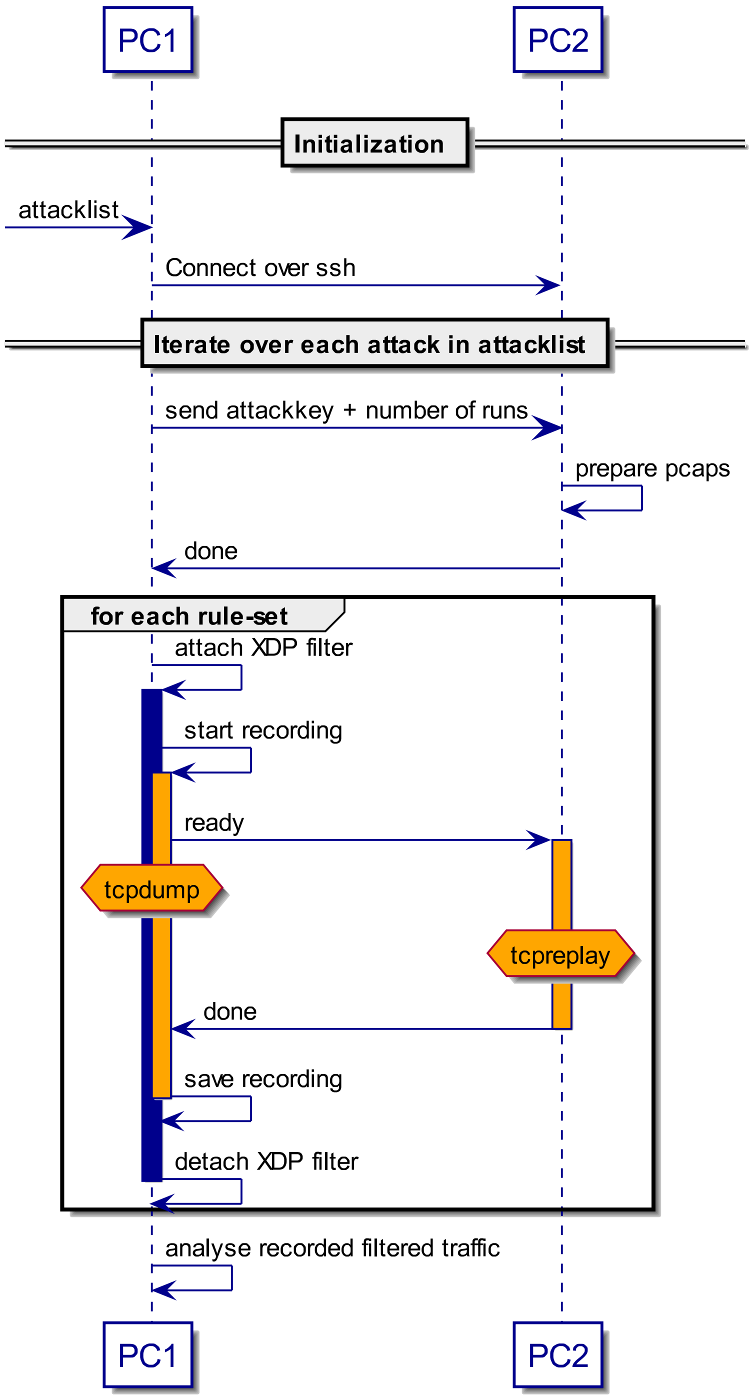

FIGURE 3.2: Testing setup

3.3 Evaluation Metrics

The build system has been described in the previous section. This section will elaborate on how this system has been tested on the effectiveness and ability to mitigate DDoS attacks. First of all the accuracy of the filter has been measured by checking how many of the total malicious packets are blocked and how many of the total normal packets are not blocked. For the input of the system, the amount of attack packets and normal packets are equally divided and therefore both 50% each run. A 'run' is defined as a specific test for a certain filter against a certain DDoS attack.

Since there are no IPv6 packets inside the attacks the IPv6 packets inside the normal traffic are left out. The prepared normal traffic pcap contains 1582358 packets, which is double the number of IPv4 in the bigflows pcap. The DDoS pcaps are replayed until the same number has been reached.

In the ideal case the output of the filter should exists of merely all normal packets and no attack packets. The accuracy metric is given by $\frac{tn+tp}{tn+tp+fn+fp}$ where $tp =$ the number of filtered malicious packets, $tn =$ the number of normal packets that are not filtered, $fp =$ the number of normal packets that are filtered and $fn =$ the number malicious packets that are not filtered. Another used metric is the false positive rate (FPR), which is given by $\frac{fp}{fp+tn}$. To measure if a recorded packet is malicious or not, a flag has been added in the preparation phase of the pcaps. The header checksum field of the IP layer of each packet has been used to store this flag.

Furthermore the limitations of the XDP filter have been measured. The first metric is the speed of applying all the rules for a specific attack. This metric shows how fast a DDoS attack can be mitigated after it has been detected by the anomaly detection part of the mitigation system. The metric will depend on the number of rules that are applied in the filter. Secondly the speed in which packets go through the filter has been measured. This way it can be compared against other former filter methods.

The effectiveness of a DDoS attack mitigation method heavily depends on the context of the system. It depends on the certain applications being used inside the to defend network, because this determines the normal type of traffic. It also depends on the set of IP addresses that are being used by the DDoS attack compared to the set of IP addresses that are used for normal traffic. For example, if a certain network uses a DNS server to resolve IP addresses and a DDoS attack is using the same DNS server to perform a reflected attack. Then blocking the attack by IP address will also drop a lot of normal traffic. Other features in the attacks packets are probably more distinguishable from normal packets than just the IP address. Measures has been done taking several contextual environments into account.

To sum up, the effectiveness of the system has been measured by using evaluation metrics for three different factors:

- XDP filter limitations
- XDP filter accuracy
- XDP filter context

3.4 Concluding Remarks

This chapter describes how our own build DDoS mitigation has been implemented. With the ability to automatically generate eBPF rules to mitigate DDoS attacks, the second research question has been answered; **How to automatically generate eBPF rules for DDoS mitigation using DDoS attack signatures?** In the first section 3.1 of this chapter the requirements for the DDoS mitigation system of this study are set. The requirements exist of four main requirements with sub-requirements. The first main requirement specifies the use of eBPF and XDP and the second main requirement specifies the ability to generate network traffic filter rules. The last requirement specifies that the system should be able to be used in a realistic environment, which includes that the system should be able to be operated by a network operator. Then in section 3.2 is shown how the different requirements have been implemented. The

IOvisor's BCC has been used to compile and load eBPF programs. BCC enables the use of Python programs in user-space that are able to communicate with kernel-space eBPF and/or XDP programs. A general XDP rule generation algorithm has been made in order to generate XDP filter rules from DDoS attack fingerprints. **This rule generation algorithm includes the possibility for a network operator to alter the rules.** This way the network operator will always be in charge of what is happening inside his network. The data that has been used for this study merely exists out of realistic network traffic, which means that the data comes from real scenarios. Santanna's DDoSDB [11] has been used for the DDoS attack fingerprints and DDoS attack traces. The bigFlows [143] attack trace has been used to simulate normal traffic in a network. Finally, section 3.3 explains how the created system has been tested and what exactly has been measured. The system DDoS mitigation effectiveness has been measured following several metrics. First of all the quantity of packets that are filtered under different circumstances are measured. Furthermore, the limitations of using an XDP filter are measured in terms of generation speed and rule size limits. Finally, all results will then be discussed considering the context of a system. The results on testing the build system will be given and discussed in the next chapter.

Chapter 4

Results & Discussion

The previous chapter elaborates on how our own build DDoS attack mitigation system has been implemented. The chapter specified what metrics were used to actually test the utility and effectiveness of the system. The goal of this chapter is to find an answer on the last research question; ***what is the effectiveness and the usability of the designed DDoS mitigation system using DDoS attack signatures?*** *As described in the previous chapter the evaluation metrics are based on three factors. The limitations of system has been measured, the accuracy has been measured and the context has been considered.*

First in ***section 4.1****, will discuss the limitations of the XDP filter. Further, the results are given on how many and what kind of packets the XDP filter was able to filter accurately. Then given all those results, in* ***section 4.2*** *discusses how effective the XDP filter can be under different circumstances. Different network properties can influence the effectiveness of certain XDP filter rules. The section also discusses how useful the filter is in a real scenario in a variable context.*

4.1 XDP filter Accuracy & Limitations

This section shows the results on testing the abilities of a XDP filter against the DDoS attacks in our data-set. In order to be able to assess the value of the accuracy results of the XDP filter, first the limitations of the XDP filter have been explored in subsection 4.1.1. Then, in subsection 4.1.2 shows the results and implications of having overlap between attack traffic and normal traffic. Finally, subsection 4.1.3 shows the accuracy results based on several different rule generation methods.

4.1.1 XDP filter limititations

One of the variables in a XDP program is the size of the maps. These maps reserve a certain amount of memory to be used between kernel-space and user-space. To filter on all values in a fingerprint, the size of a fingerprint determines how big the eBPF maps should be. The more information a fingerprint contains, the more time it takes to load eBPF maps of that are equal of size as the information in the fingerprint. Table 4.1 shows the statistics of the fingerprint with the most information inside the fingerprint.

TABLE 4.1: Statistics for fingerprint with most information.

Feature	**#**
Source IP addresses	5409107
Total lengths	3
Header lengths	3
Destination ports	1
Source ports	65535
Flags	4
Window sizes	3
Urgent Pointers	1

Table 4.1 shows that in the worst case, the fingerprint contains 5409107 source IP addresses, 65535 source ports and some extra values for the other features. It is a TCP DDoS attack with in this case no deeper packet information after the TCP layer. Note 'most information' in this case means the amount of bytes and not the number of values.

The influence of this size on the load speed of the program can be seen in Figure 4.1. It shows the curve for the size of an eBPF map plotted against the time it takes to load the specific eBPF program including the map into the system memory. The storage of IP addresses needs 32-bit hashmaps and for a port value it needs 16-bit hashmaps. All-together this means that a memory storage for a 5500000 32-bit hashmap, is the rounded up maximum size needed for the use of our dataset. This means that 3.37 seconds is the maximum time needed to load an eBPF program into memory. This is a reasonable time needed to mitigate a DDoS attack, especially when considered that DDoS attacks can result in a denial of service of over hours. In our tests the time needed to reserve the memory for the maps up to 4000 needed a minimum time of 1.16 seconds.

The size of the rules inisde a XDP filter can be loaded in a reasonable time. That is the main takeaway message of Figure 4.1, but note that this is the case for the worst case fingerprint in the dataset used for this study. It is possible that higher load times will be found for fingerprints outside the scope of this study.

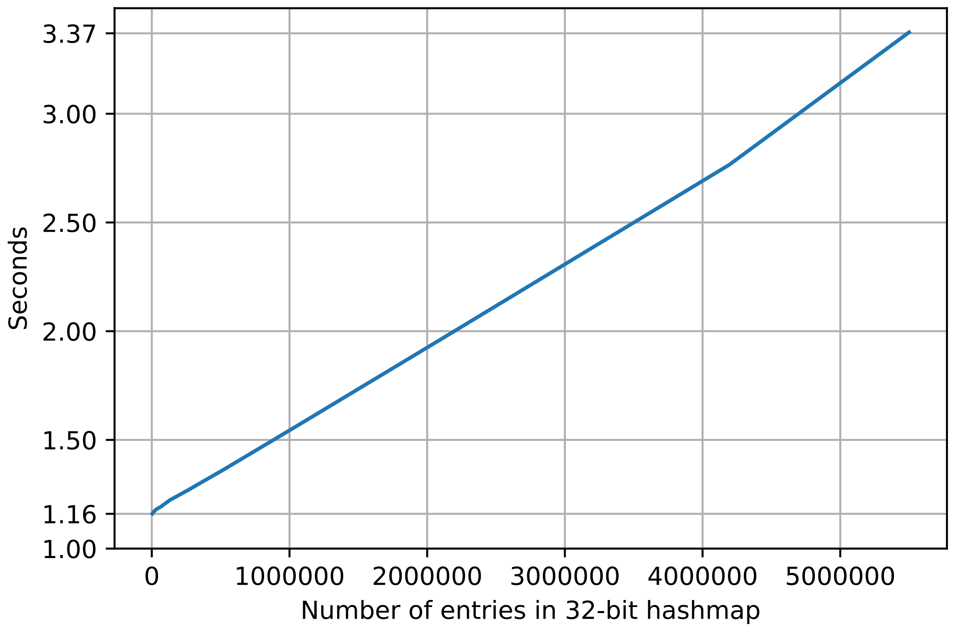

FIGURE 4.1: Load speeds for a single eBPF hashmap with 32bit entries

Not only the amount of rules can form a limitation, but also the filter speed of the filter can be a limitation. As we discussed in chapter 2, the speed advantage of eBPF and XDP over other tools is one of the reasons we choose for eBPF and XDP in our system. We tested if this advantage still holds when eBPF and XDP are used with different filter rules. Tested is how many packet per second the kernel of our set up is able to handle by the use of a tool developed by [145]. The tool is slightly modified in a way that it is sending DNS packets containing 32 bytes of data. This way the size of the packets is small enough to test the packet processing speed and the packets can also be tested against a filter that filters on DNS query types. For those tests four different scenario's are tested. The first scenario tests the baseline for one CPU with no filters at all. Then, a XDP and an Iptables filter were used filtering on 5409107 different IP addresses. Those IP addresses come from the fingerprint with the largest set of IP addresses. The last XDP filter, filters on the same set of IP addresses, all UDP packet features and the DNS query type. Figure 4.2 shows the results of those tests. It shows that the machine used in the setup is able to process 1.06 million packets per second when no filter at all is used. It also shows that with an Iptables filter on 5409107 IP addresses, the same CPU is able to process 0.39 million packets per second. With a XDP filter on the same set of IP addresses the CPU is able to filter 0.77 million packets per second. Finally the third bar shows that adding more checks on a few more features in the filter, doesn't influence the processing speed.

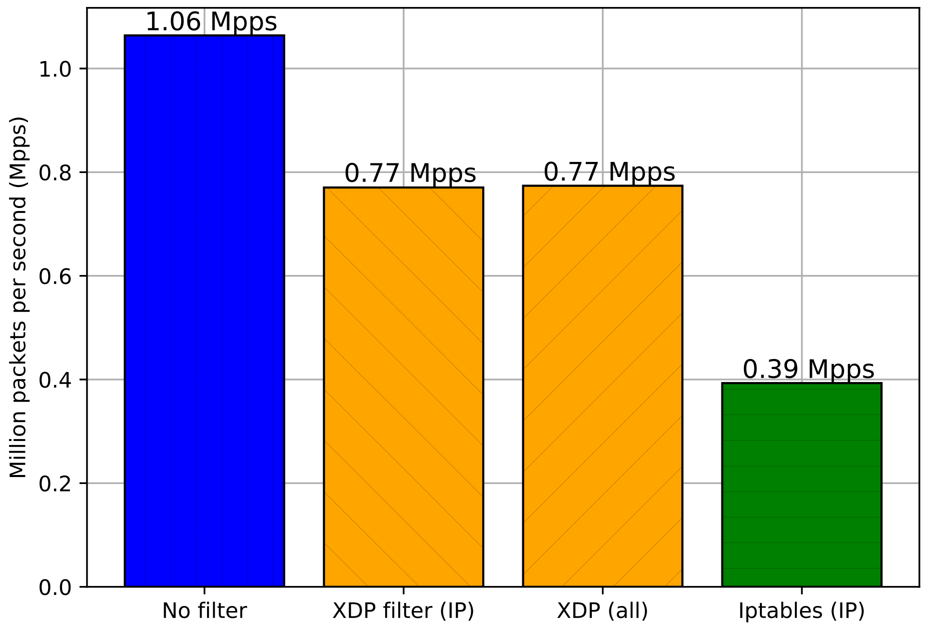

FIGURE 4.2: Kernel packet processing speeds on a single CPU core with different filters.

The results from Figure 4.2 say that processing a packets with a XDP filter is significantly slower compared to having to filter at all. However, the same results show that processing packets with a XDP filter is significantly faster compared to a Iptables filter. **A XDP filter is able to faster process packets than an Iptables filter with the same rule.** Note that those processing speeds rates are not the dropping speeds. Nevertheless, similar differences in rates can be expected when the packets are being dropped, since the packets follow the same path through the filter. Another finding on the results of Figure 4.2 says something about the impact on the processing speed when the complexity of rules increase. The more features used inside the filter, the higher the complexity of the rules inside the filter. The amount of different features in the filter does not influence the processing speed of the filter. **The amount of fingerprint information in the filter has a bigger influence on the packet processing speed than the complexity of the rules.**

4.1.2 Overlap between attack and normal traffic

This subsection shows the results and implications of a scenario in which the feature values of normal traffic and attack traffic overlap. One would expect that if the XDP filter drop packets based on a value that occurs in both normal traffic and attack traffic, the filter will also drop both kinds of traffic. This is an obvious implication of filtering on values that have overlapping feature values. To prove this hypothesis, the filter tested is an source IP address filter which blocks all the source IP addresses that are given in the fingerprint of the attack. This feature has been chosen because it is the first value in a network packets that is in a network packet. The normal traffic has been artificially modified to have an overlap in the values of those IP source addresses. This has been done under 5 different ratings, which are 0%, 25%, 50%, 75% and 100% overlapping source IP addresses between the DDoS attack and the normal traffic. Multiple overlapping percentages are linearly taken to prove that the behaviour of the filter is as we would expect. Since there were no false negatives, the false positive rates for all attacks are shown in Figure 4.3.

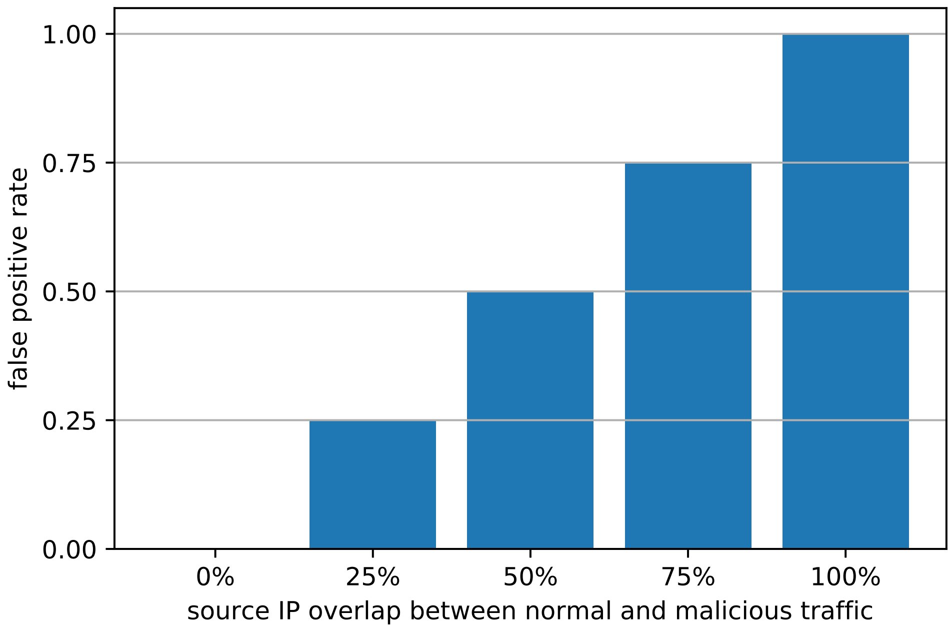

FIGURE 4.3: XDP filter accuracy for all attacks, with a source IP address filter

Figure 4.3 shows that the false positive rate of the filter perfectly matches the rate of overlapping IP addresses. With 0% overlapping IP addresses the filter works perfect, which means that it drops all the malicious packets and that all normal packets come through. With 100% overlapping IP addresses between normal and attack traffic, the false positive rate is exactly 1.0. This number means that all packets are dropped, including the normal the traffic. **This result is unwanted in a realistic situation since the service that is protected by the filter will still be unreachable.** The decision on what an acceptable false positive rate is, will be decided by the network operator. The results displayed in Figure 4.3 show that the filter is able to accurately drop DDoS attack packets as long as the to drop values in the filter are not overlapping with the normal traffic. The overlap of source IP addresses between normal and DDoS attack traffic will depend on the context of the system. The same overlap percentages can occur when other features are taken into account, but the more deeper in the packet, the more likely their might be some specific values that characterize a certain DDoS attack. The character of a DDoS attack is that they come from a distributed network of different machines with all different IP addresses. This means that it is more likely that an IP address will overlap with a normal traffic IP address. The next test wills be done considering that an operator wants to white-list the IP source feature, because of the big overlap between normal and attack traffic. The next subsections will test filters with more in depth packet filters.

4.1.3 One ore more feature(s) filter

There are two different methods specified in the previous chapter to select a feature for the filter. The first method picks a feature based on the amount of different values it has in the fingerprint. The second method picks a feature based on the Adjusted Mutual Information Score (AMI) score of each feature. The AMI score helps by determining which features should be selected for the filter. A feature with a higher AMI score is more likely to distinguish the attack from normal traffic. In Table 4.2, for each feature has been given how many times it is the highest AMI score for an certain attack.

TABLE 4.2: Number of times a feature has the highest AMI score for an UDP attack

Feature	**Number of UDP attacks**
packet length	2
UDP source port	144
UDP destination port	13
UDP payload length	94
Total	**253**

As can be seen in Table 4.2 the UDP source port has most often die highest AMI score, with 144 out of 253. This number can be explained due to the fact that UDP are often reflected. As described in chapter 2, the packets from those attacks will come from the same source port of the machine that reflects the attack packets. Furthermore, the destination port feature has significantly less times the highest score than the source port. Note that, all UDP attacks are considered here and therefore only possible features have been taken up until OSI layer 4.

The accuracy of the filter has been tested by going through different scenario tests. The total number of packets that were going trough the filter each test are 3164716 packets. Half of them were labeled as DDoS attack packets and half of them were labeled as normal. The setup has been tested without any filter and all the 3164716 packets came trough for each DDoS attack. For all 253 UDP attacks has been tested what the XDP filter accuracy is by filtering on different feature sets. For each XDP filter, each attack has repeatedly been replayed until 1582358 packets were send. In the meanwhile also the normal traffic has been replayed twice with a total of 1582358 normal packets. By analysing the packets that were received behind the XDP filter, it was possible determine the accuracy and false positive rate for that specific setting. Since there were no false negatives, the results are presented as false positives. Note that each attack has been tested four times under for different XDP filters. The results are given in Figure 4.4.

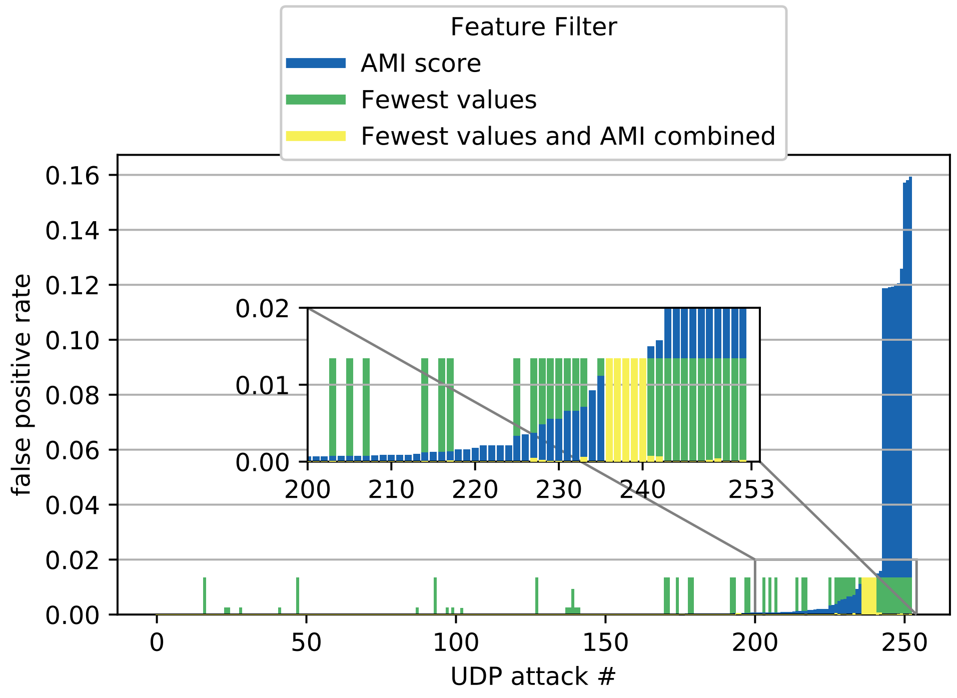

FIGURE 4.4: False positve rate for each UDP attack sorted on the number of false positives

Figure 4.4 shows the false positive rates of a filter with the fewest feature values in the fingerprint and the false positive rate of a filter with the highest AMI score for each individual test. The third test has been done with filtering on those both features combined. The latter test shows a lower or equal false positive rate for each run that has been done. Finally when all UDP features are considered an accuracy of 100% has been shown. **Taking all features values inside the UDP fingerprints is enough to drop all DDoS packets with zero false positives.** Considered has to be that this result has been achieved by using the bigFlows dataset for normal traffic, which means that this result cannot necessarily be achieved in every network.

In the figure, the 'AMI score' bars shows the highest peaks, with some attacks that resulted in a false positive score of 0.16. This number means that the filter dropped 16% of the normal traffic was dropped in the worst case scenario. This means that 84% of the normal traffic is able to reach the network behind the filter. Taking into account that all DDoS traffic has been dropped, this means that this is a good score considering the potential impact of a DDoS attack can be that 0% of the normal traffic reaches the network. This score was found when the filter was tested against one out of the 253 UDP DDoS attacks, which means that is was the worst case scenario of all the attacks considered in this study. The 'fewest values' bars show a maximum false positive rate of 0.013, which is lower than the peak of the 'AMI score' bars. However, the 'fewest values' bars shows that the there are more attacks that cause a false positive rate above the 0.01 compared to the 'AMI score' bar.

Table 4.3 shows how many times a certain TCP feature has the highest AMI score for a certain attack. Again all TCP attack are considered with features up until OSI level 4.

TABLE 4.3: Number of times a feature has the highest AMI score for an TCP attack

Feature	**Number of TCP attacks**
packet length	0
TCP source port	0
TCP destination port	1
TCP header length	30
TCP flags	103
TCP window size	2
TCP urgent pointer	142
Total	**278**

Table 4.3 shows that the packet length feature and the TCP source port feature never have the highest AMI score in our dataset. The packet length of a TCP packet without any deeper layers do have in most cases have the same length, whether the packet is from a normal packet or attack packet. Further, with 141 out of the 278 total TCP attacks, the TCP urgent pointer most often has the highest AMI score. Normally the urgent pointer specifies where the urgent data ends in the TCP stream. This is normally used to specify that this data should directly be delivered to the application instead of being queued in a data buffer. For DDoS attacks this can be used to force packets to be processed. However the TCP urgent pointer flag is often not set in the packets, which makes the urgent pointer field itself useless. Nonetheless, the value in this field is something that can identify DDoS attacks in our dataset. After the urgent pointer, the TCP flags are secondly most often the feature with the highest AMI score. Tests have been done with different selected filters as shown in Figure 4.5. The tests are done in the same way as done for the UDP attacks.

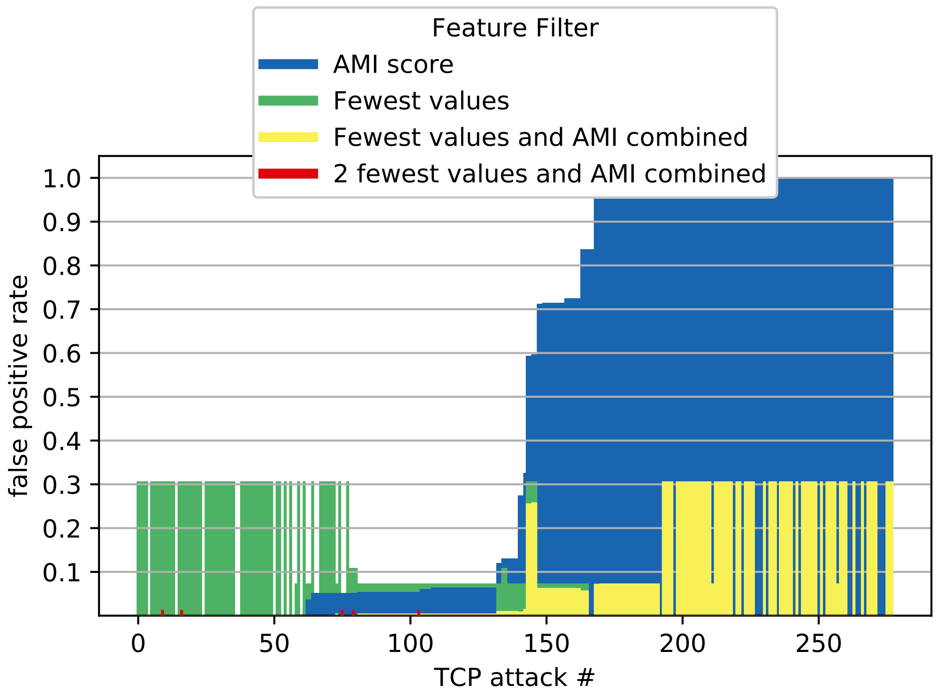

FIGURE 4.5: False positve rate for each TCP attack sorted on the number of false positives

Figure 4.5 shows the results of four different tests. In the first test the filter filters on the feature with the highest AMI score, in the second test the filter has been applied on the feature with the fewest elements in the fingerprint. Furthermore, the test shows the results when the filter drops packets based on values in both features and also a test with three different features; 2 fewest value features and a feature with the highest AMI score. Finally the results for a filter on all features. For the highest AMI scores feature can be seen that 60 TCP runs have no false positives at all and 110 of the tests have a false positive rate of 1. The 'fewest values' filter shows less false positives and even less false positives are reported with a filter that filters on both the feature with the highest AMI score and the feature with the fewest values in the fingerprint. The fourth type of bars with three features shows that in 5 out of the 278 attack there are just a few false positives at all. The attack with the highest false positive rate had a false positive rate of 0.01. This is a significant improvement over the previous tests. Lastly, when the filter is used on all TCP features, without the IP address, there were no false positives. A 100% accuracy has been reached when all TCP features in the fingerprint are considered in the XDP filter. Note that this under the circumstances used in this study. The bigFlows dataset considered as normal traffic might form a bias.

In the comparison between the results of TCP attacks and UDP attacks, it shows that the UDP filter shows better results when one or two features are taken to generate filter rules. TCP has more features to filter on so on beforehand it was expected that it was more likely to find a feature that resulted in less false positives than on filtering on UDP packets. This difference can partly be explained by the fact that the fraction of TCP is higher in the normal dataset that has been used. Another reason is that there are more attacks that contain deeper layers the UDP attack set. Therefore the UDP payload length in the UDP header becomes in many cases a distinguishing feature.

With eBPF and XDP it is even possible to filter on features in higher OSI levels. For the 45 DNS DDoS attacks the AMI scores have also been calculated, with the result that for all attack the question type of the DNS request always had the highest score. All DDoS attacks made use of the ANY request type, whereas in the normal traffic none of the DNS packets contained this type of request. The filter therefore showed 0 false positives and 0 false negatives. **A 100% accuracy has been reached when just one deep layer feature has been used in the filter.** The ability of eBPF and XDP to filter on deep layer packet features, has made this result possible. Note, that this result is true for this special case, by filtering DNS packets in a specific dataset. Result might differ when other protocols, other attacks or other normal traffic is used.

The results shown in this subsection might be sufficient to mitigate a DDoS attack. The results are all produced on two different feature selection methods, with the 'fewest values' method and the 'AMI score' method. Some of the results show that all packets of the DDoS attack are dropped and that there are certain percentages of falsely dropped packets. A potential problem for the network operator with these methods used, is that many values are used in the filter. This means that there is not much space to remove values from the filter to be able to reduce the false positives. For example if a feature is chosen based on the least values, there are per definition not many values the filter. If the filter contains just one value, than the network operator has no possibilities to whitelist any values. Furthermore, if there is a critical service between one the false positives, a the network operator might want to white-list specific values, with the risk of being vulnerable for the DDoS attack again. The next section discusses the operating choices of a network operator under different scenarios.

4.2 The usability of the filter

The built system for this study is open-source and can be found online on github [137]. Combining the system with DDoS fingerprints, it can be used by a network operator to defend his network against DDoS attacks. The effectiveness of certain rules to drop DDoS attack packets heavily depends on the context of the filter. Furthermore, the applied rules in the system have a potential impact on the system of the network operator. The effectiveness of a rule and the impact of a rule depend on the size of the network, the place of the network and what applications are used on the network. In order to make sure that a network operator is able to mitigate a DDoS attack without any further complications, he needs to be able to check the impact of certain rule on the network. As a network operator then has to decide how many false positives and how many false negatives he allows in his filter. How a network operator can use the system to obtain those those numbers has been discussed in subsection 4.2.1. The less features used in the XDP filter, the faster the network operator is able to deploy the rules, but the probability on false positives increases. A second possibility a network operator has, is choosing to white-list certain specific values from a feature. The advantage of this is that specific false positives can be targeted to make sure that specific normal traffic will not be dropped. Finding the optimal balance between false positives, false negatives and mitigating the DDoS attack has been discussed in subsection 4.2.2.

This study has been focused on the ability of XDP to drop packets with a high accuracy. Other studies already proved that XDP is able to drop packets on high speeds. This study shows that this speed advantage is still there if XDP rules are applied. However, those test were done with minimum sized artificially made packets to be able to fully stress the CPU of the machine in the setup. This was necessary because of the 1 Gbps network link in the setup. Future studies should measure the impact of the XDP rules on the dropping speeds with a higher speed network link and real DDoS attacks.

4.2.1 Determining the performance

In order to make sure that a DDoS attack gets mitigated and that there are no false positives, a network operator can act based on what is happening inside the network. Whenever a XDP filter has been applied against an incoming DDoS attack, the network operator can see how the filter performs. The first step for a network operator should be to map all bottlenecks in his network. He should know what the maximum size of incoming traffic is before a denial of service occurs, in terms of data size and amount of packets. As stated the detection a DDoS attack itself should be performed by the anomaly detection part of the system which is not part of this study. One of the simplest anomaly detection method could be to monitor the incoming traffic rate for example. This rate should also be the first step of a network operator that wants to check the performance of the filter. If the incoming traffic rate is the same as before applying the filter, it could mean that the DDoS attack is 100% accurately mitigated, the DDoS attack already has been stopped or that the fraction false positives is equal to the fraction of false negatives. If the incoming traffic rate is lower after applying the filter, it most certainly means that there are false positives. Finally if the incoming traffic rate is higher than in a normal situation, nothing can be really concluded except that the attack traffic has most probably not fully been filtered.

Comparing just the incoming traffic rates tells the network operator some information, but not everything he needs to know. That is why the built system gives some information about the dropped packets. **The drop information that the system gives is two-fold, the amount of packets that have been dropped and a sample of the dropped packets.** With this information the network operator is able to determine the false positives and false negatives of the applied filter. The traffic that has not been dropped should contain only normal traffic, but can be check on possible false negatives by comparing this traffic to the fingerprint of the attack. This is however only relevant if the DDoS is not mitigated yet. As long as the DDoS has been mitigated, the network operator can allow some false negatives. The false negatives can be determined by analysing the dropped packets sample. As the results show, a DDoS attack packet can be distinguished from a normal packet by comparing all features of the packets against the attack fingerprint.

4.2.2 DDoS attack scenarios

The setup that has been tested in this study made some assumptions on the context of the system. First of all, as shown in Figure 3.1 of the previous the chapter, the build system is considered to be part of a larger system. Ideally the overall system should contain an anomaly detection part which is able to detect an incoming attack and also able to match it against a certain DDoS attack fingerprint. This seems to be an essential part of the system, but doesn't mean that the system built for this study cannot operate on itself. The DDoSDB platform used for this study contains an open database which enables the sharing of DDoS attacks between network operators. With the system build for this study, XDP filters can be applied with any DDoS attack fingerprint in order to mitigate that specific attack. A simple DDoS detection system based on the incoming traffic rate can already be sufficient for the network operator to be notified about a DDoS attack.

A second assumption made on the context of the build system is that there are AMI scores inside the fingerprints. Those AMI scores define which features to selection in the AMI score filters. The AMI scores were obtained by comparing the normal network traffic against the DDoS attack traffic. Therefore, the training data in this study was the same set of data as the testing data, which means that the AMI scores are biased in this study. Ideally those AMI scores should be generated by each individual network operator based on their own 'normal traffic' sample. Network operators are open to share their findings on those AMI scores. With all the findings of network operators together, generic AMI features can be found for each attack. If those commonly found AMI features are shared via the fingerprints as well, a network operator can choose to rely on those values instead of doing his own analysis.

The results found in this study heavily depends on the normal network traffic dataset used. This dataset defines the context of the system, which of course can differ in other realistic networks. Also the impact of false negatives depends on the resources of the system and the strength of the DDoS attack. The filter should filter just enough packets to not cause an exhaustion of resources behind the filter. This means that a DDoS attack already is mitigated if there is no denial of service anymore, but there can still be many false negatives. As stated, the impact of a false positive depends on the context of the system. Note that the false positive rates in the results of the previous section are based on a 50/50 divided normal and attack traffic. As we known from the second chapter, we know that this is not a realistic fraction. DDoS attacks usually have a bigger share in the incoming traffic size, this means that the false positive rates should not be interpreted on itself, but relatively to each other. The rates will in practice be lower than the found rates in this study.

As an example, in a first scenario it could be possible that a network operator mitigates a DDoS attack by filtering on the source port of a packet. The fingerprint contains only one value for the source port which is 53. The attack packets might successfully dropped, but if there is a machine behind the filter that does DNS requests it, the DNS replies will also be dropped because this the protocol that uses this port. DNS in this case is common protocol which is used in many network all over the world. Therefore, more features should be considered to filter the packets.

In another scenario, it might be possible that among all the functions inside the network behind the filter is one specific function that is really critical. The false positive rates might be really low, but if this one essential function does not work anymore the network operator should decide to whitelist the values that cause this function normally.

Lastly, it is also possible in a scenario that a DDoS is successfully mitigated by a filter including a white-list and the system does not report any false positives or false negatives. The network operator should stay aware, since there might be some cases he want to add values to the white-list. This is because he might have forgot something to add to the white-list.

4.3 Concluding Remarks

The goals of this chapter was to answer; **what is the effectiveness and the usability of the designed DDoS mitigation system using DDoS attack signatures?** The answer of this question has been found, considering that the open-source built system is being operated by a network operator. Assuming that the DDoS attack fingerprints are 100% accurate, a network operator is able to mitigate DDoS attacks using the built system which uses eBPF and XDP. This chapter showed the accuracy of the built system and how it should be operated by a network operator.

The beginning of this chapter showed that the rules inside the XDP do not show any problematic limitations on the data used for this study. With a maximum of 3.37 seconds load speed, it is a reasonable time to load filter rules into an eBPF map. Then, the section gave different results for different feature selection algorithms. Furthermore, the packet processing speed while having rules in a XDP filter is faster than, having the same rules in an Iptables filter. It showed that in case of filtering UDP attacks, just a few attacks had a false positive rate of above 0.01 when just two features were selected. The first feature was selected based on the AMI score and the other one based on the amount values for that specific feature inside the fingerprint. To obtain the same false positive rate to filter TCP attack, a third feature was needed. In case of deeper packet values, a 100% accuracy score had been reached. However, in the second section those results were put, into a realistic perspective. From the discussion can be concluded that the optimal settings heavily depend on the context of the system. A network operator should always be the one who assesses what settings are needed for his network. The conclusion on the performance of eBPF and XDP is that it is able able to mitigate potential DDoS attacks with low false positive rates and low false negative rates. Especially since it is possible to filter packets based on deep packet features. However a network operator should always be there to judge the impact of the applied rules and if necessary put certain values in the white-list of the rule generator. This judgement can be done by the network operator by analysing the samples of dropped network traffic, which is provided by the system. It can be concluded that with the built system as a tool and with accurate DDoS attack fingerprints, a network operator is able to mitigate DDoS attacks effectively.

Chapter 5

Conclusion

The previous chapter gave answer to the last third research question. This chapter sums up the conclusions on all research questions as stated in the introduction:

- ***RQ1:*** *What methods currently exist that automatically update rules in a signature based DDoS mitigation?*
- ***RQ2:*** *How to automatically generate eBPF rules for DDoS mitigation using DDoS attack signatures?*
- ***RQ3:*** *What is the effectiveness and the usability of the designed DDoS mitigation system using DDoS attack signatures?*

The answers on those questions provide new insights for the scientific world and on top of that the answers can also help network operators to defend their systems against DDoS attacks. Section 5.1 concludes on the contribution and impact of this study.

Chapter 2 started on answering the first research question; **What methods currently exist that automatically update rules in a signature based DDoS mitigation?** Concluded is that the relatively new open-source tools eBPF and XDP have some features that could overcome the limitations of other methods. These tools can be deployed on a wide variety of devices and enable the possibility to collaborate with other applications via user-space. This collaboration can overcome the limitation that signature-based mechanisms have, since it enables the possibility to dynamically update filter rules, that are automatically generated by secondary program. Furthermore, eBPF and XDP are able to fully inspect incoming packets at a even higher processing rate than former network firewall methods. DDoS protection services (DPSs) are a popular way to outsource the defense against DDoS attacks. Those DPSs use content delivery networks (CDNs) to be able to redirect internet traffic, which can be used to mitigate DDoS attacks. Implementation details on further methods are generally not public, but Cloudflare posts openly about their successful method using eBPF and XDP.

With the ability of eBPF to communicate with all sorts of functions inside the Linux kernel and user-space, it enables to possibility to create a dynamic and customizable solution. The XDP hook inside the Linux kernel is able to process packets on high speeds due to the fact that packets are filtered close to the NIC. Related studies show that no studies use eBPF and XDP in order to automatically update and generate rules with the use of attack signatures. Future study directions have to include the limitations on eBPF and filtering algorithms that are used to match packets against rules.

Chapter 3 describes how our own build DDoS mitigation has been implemented. With the ability to automatically generate eBPF rules to mitigate DDoS attacks, the second research question has been answered; **How to automatically generate eBPF rules for DDoS mitigation using DDoS attack signatures?** In this chapter the requirements for the DDoS mitigation system of this study are set. The requirements exist of four main requirements with sub-requirements. The first main requirement specifies the use of eBPF and XDP and the second main requirement specifies the ability to generate network traffic filter rules. The last requirement specifies that the system should be able to be used in a realistic environment, which includes that the system should be able to be operated by a network operator. Then is shown how the different requirements have been implemented. The IOvisor's BCC has been used to compile and load eBPF programs. BCC enables the use of Python programs in user-space that are able to communicate with kernel-space eBPF and/or XDP programs. A general XDP rule generation algorithm has been made in order to generate XDP filter rules from DDoS attack fingerprints. This rule generation algorithm includes the possibility for a network operator to alter the rules. This way the network operator will always be in charge of what is happening inside his network. The data that has been used for this study merely exists out of realistic network traffic, which means that the data comes from real scenarios. Santanna's DDoSDB [11] has been used for the DDoS attack fingerprints and DDoS attack traces. The bigFlows [143] attack trace has been used to simulate normal traffic in a network. Finally, the chapter explains how the created system has been tested and what exactly has been measured. The system DDoS mitigation effectiveness has been measured following several metrics. First of all the quantity of packets that are filtered under different circumstances are measured. Furthermore, the limitations of using an XDP filter are measured in terms of generation speed and rule size limits.

The goal of chapter 4 was to answer; **what is the effectiveness and the usability**

of the designed DDoS mitigation system using DDoS attack signatures? The answer of this question has been found, considering that the open-source built system is being operated by a network operator. Assuming that the DDoS attack fingerprints are 100% accurate, a network operator is able to mitigate DDoS attacks using the built system which uses eBPF and XDP. This chapter showed the accuracy of the built system and how it should be operated by a network operator.

The beginning of this chapter showed that the rules inside the XDP do not show any problematic limitations on the data used for this study. With a maximum of 3.37 seconds load speed, it is a reasonable time to load filter rules into an eBPF map. Furthermore, the packet processing speed while having rules in a XDP filter is faster than, having the same rules in an Iptables filter. Then, the next section gave different results for different feature selection algorithms. It showed that in case of filtering UDP attacks, just a few attacks had a false positive rate of above 0.01 when just two features were selected. The first feature was selected based on the AMI score and the other one based on the amount values for that specific feature inside the fingerprint. To obtain the same false positive rate to filter TCP attack, a third feature was needed. In case of deeper packet values, a 100% accuracy score had been reached. However, in the second section those results were put, into a realistic perspective. From the discussion can be concluded that the optimal settings heavily depend on the context of the system. A network operator should always be the one who assesses what settings are needed for his network. The conclusion on the performance of eBPF and XDP is that it is able able to mitigate potential DDoS attacks with low false positive rates and low false negative rates. Especially since it is possible to filter packets based on deep packet features. However a network operator should always be there to judge the impact of the applied rules and if necessary put certain values in the white-list of the rule generator. This judgement can be done by the network operator by analysing the samples of dropped network traffic, which is provided by the system. The built system is publically available on github [137]. It can be concluded that with the built system as a tool and with accurate DDoS attack fingerprints, a network operator is able to mitigate DDoS attacks effectively.

5.1 Contribution

As stated in the introduction, studies about the mitigation of DDoS attacks are not new. DDoS attacks already exists since the early 2000's. The eBPF and XDP as tools to migitate DDoS attacks have been studied since 2016, which means that they are relatively new tools. Those studies only showed that eBPF and XDP are in potential very effective tools to mitigate DDoS attacks in terms of speed. However, no study up on so far showed how those tools can actually be used to mitigate DDoS attacks. No study used real DDoS attack in combination with the tools XDP and eBPF. This study shows how to create filter rules in order to mitigate DDoS attacks for eBPF and XDP. This study shows that a XDP filter with real applied rules is faster than Iptables, which is a new finding the scientific world. This study shows also the potential accuracy of the filter which can be up to 100% when deep layer packets features are considered. Future studies can build upon those findings, for example on the optimization of the rules or testing the system in a real network setup. This study can also be used in practice for network operators. This study offers a open-source system that can be used as a tool to mitigate DDoS attacks in real network scenario's. The system offers similar techniques that are used by big DDoS protection services, such as Cloudflare. However cloudflare, can use of the abilities of a big Content

Delivery Network (CDN) in the fight against DDoS attacks. This can not directly be compared to the abilities of the system created for this study.

Appendix A

Setup specifications

```
iperf3 -b 1000M -u -c 10.0.0.50
Connecting to host 10.0.0.50, port 5201
local 10.0.0.41 port 48591 connected to 10.0.0.50 port 5201
Interval Transfer Bandwidth Total Datagrams
0.00-1.00 sec 102 MBytes 855 Mbits/sec 13052
1.00-2.00 sec 113 MBytes 950 Mbits/sec 14501
2.00-3.00 sec 113 MBytes 950 Mbits/sec 14499
3.00-4.00 sec 113 MBytes 950 Mbits/sec 14504
4.00-5.00 sec 113 MBytes 950 Mbits/sec 14497
5.00-6.00 sec 113 MBytes 950 Mbits/sec 14496
6.00-7.00 sec 113 MBytes 950 Mbits/sec 14496
7.00-8.00 sec 113 MBytes 950 Mbits/sec 14497
8.00-9.00 sec 113 MBytes 950 Mbits/sec 14496
9.00-10.00 sec 113 MBytes 950 Mbits/sec 14503
- - - - - - - - - - - - - - - - - - - - - - - - -
Interval Transfer Bandwidth Jitter Lost/Total Datagrams
0.00-10.00 sec 1.10 GBytes 941 Mbits/sec 0.082 ms 0/143540 (0Sent 143540 datagrams
iperf Done.
```

Bibliography

[1] Netscout. NETSCOUT's 14th Annual Worldwide Infrastructure Security Report, 2019. URL `https://www.netscout.com/press-releases/netscout-releases-14th-annual-worldwide-infrastructure`.

[2] Akamai. state of the internet / security Ret ail Attacks API Traffic Letter from the Editor. 5(2), 2019.

[3] Puneet Zaroo. A survey of DDoS attacks and some DDoS defense mechanisms. *Advanced Information Assurance (CS 626)*, 2002.

[4] Verisign. Verisign Distributed Denial of Service Trends Report. 5(1):1–8, 2015. URL `https://www.verisign.com/assets/report-ddos-trends-Q32015.pdf`.

[5] Attack Sp and O T Li. State Of The Internet: Security | Summer Attack Spotlight | Akamai. 2018.

[6] Opeyemi Osanaiye, Kim-Kwang Raymond Choo, and Mqhele Dlodlo. Distributed denial of service (DDoS) resilience in cloud: review and conceptual cloud DDoS mitigation framework. *Journal of Network and Computer Applications*, 67:147–165, 2016.

[7] Gilberto Bertin. XDP in practice: integrating XDP into our DDoS mitigation pipeline. In *Technical Conference on Linux Networking, Netdev*, volume 2, 2017.

[8] Arthur Fabre. L4Drop: XDP DDoS Mitigations, 2018. URL `https://blog.cloudflare.com/l4drop-xdp-ebpf-based-ddos-mitigations/`.

[9] Anant Deepak, Richard Huang, and Puneet Mehra. eBPF / XDP basedfirewall and packet filtering. In *Linux Plumbers Conference 2018 NetworkingTrack*, 2018.

[10] Toke Høiland-Jørgensen, Jesper Dangaard Brouer, Daniel Borkmann, John Fastabend, Tom Herbert, David Ahern, and David Miller. The eXpress data path: fast programmable packet processing in the operating system kernel. In *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*, pages 54–66. ACM, 2018.

[11] J J Santanna. DDoSDB, 2019. URL `https://ddosdb.org`.

[12] Ronen Kenig, Deborah Manor, Ziv Gadot, and Daniel Trauner. DDoS Survival Handbook. *Radware*, pages 1–56, 2013. URL `http://security.radware.com/uploadedFiles/Resources_and_Content/DDoS_Handbook/DDoS_Handbook.pdf%5Cnhttps://security.radware.com/uploadedfiles/resources_and_content/ddos_handbook/ddos_handbook.pdf`.

[13] William F Slater III. The Internet outage and attacks of October 2002. *Chicago Chapter of the Internet Society*, (October):1–14, 2002.

[14] Jelena Mirkovic and Peter Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2): 39, 2004. ISSN 01464833. doi: 10.1145/997150.997156. URL `http://portal.acm.org/citation.cfm?doid=997150.997156`.

[15] K Munivara Prasad, A Rama Mohan Reddy, K Venugopal Rao, By K Munivara Prasad, KVenugopal Rao, K Munivara Prasad α, A Rama Mohan Reddy σ, and K Venugopal Rao ρ. DoS and DDoS Attacks: Defense, Detection and Traceback Mechanisms -A Survey DoSandDDoSAttacksDefenseDetection andTraceback Mechanisms-A Survey DoS and DDoS Attacks: Defense, Detection and Traceback Mechanisms -A Survey. *Global Journal of Computer Science and Technology: E Network*, 14(7), 2014. URL `https://globaljournals.org/GJCST_Volume14/3-DoS-and-DDoS-Attacks-Defense-Detection.pdf`.

[16] S T Zargar, J Joshi, and D Tipper. A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Communications Surveys Tutorials*, 15(4):2046–2069, 2013. ISSN 1553-877X. doi: 10.1109/SURV.2013.031413.00127.

[17] H. Zimmermann. OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications*, 28(4): 425–432, 4 1980. ISSN 0096-2244. doi: 10.1109/TCOM.1980.1094702. URL `http://ieeexplore.ieee.org/document/1094702/`.

[18] Akamai. DDoS Attacks | Akamai. URL `https://www.akamai.com/us/en/resources/ddos-attacks.jsp`.

[19] E. Fenil and P. Mohan Kumar. Survey on DDoS defense mechanisms. *Concurrency Computation*, (July 2018):1–19, 2019. ISSN 15320634. doi: 10.1002/cpe.5114.

[20] Marco Deseriis. Hacktivism: On the Use of Botnets in Cyberattacks. *Theory, Culture and Society*, 34(4):131–152, 2017. ISSN 14603616. doi: 10.1177/0263276416667198.

[21] Steve Mansfield-Devine. The growth and evolution of DDoS. *Network Security*, 2015(10):13–20, 10 2015. ISSN 1353-4858. doi: 10.1016/S1353-4858(15)30092-1. URL `https://www.sciencedirect.com/science/article/pii/S1353485815300921`.

[22] Steve Mansfield-Devine. DDoS goes mainstream: how headline-grabbing attacks could make this threat an organisation's biggest nightmare. *Network Security*, 2016(11):7–13, 11 2016. ISSN 1353-4858. doi: 10.1016/S1353-4858(16)30104-0. URL `https://www.sciencedirect.com/science/article/pii/S1353485816301040`.

[23] Brian Ricks, Bhavani Thuraisingham, and Patrick Tague. Lifting the Smokescreen: Detecting Underlying Anomalies During a DDoS Attack. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 130–135. IEEE, 11 2018. ISBN 978-1-5386-7848-0. doi: 10.1109/ISI.2018.8587338. URL `https://ieeexplore.ieee.org/document/8587338/`.

[24] Matti Mantere. Stock Market Manipulation Using Cyberattacks Together with Misinformation Disseminated through Social Media. In *2013 International Conference on Social Computing*, pages 950–954. IEEE, 9 2013. ISBN 978-0-7695-5137-1. doi: 10.1109/SocialCom.2013.149. URL `http://ieeexplore.ieee.org/document/6693446/`.

[25] onemon Institute LLC. TRENDS IN THE COST & DENIAL OF SERVICE ATTACKS : ASIA-PACIFIC. (June), 2018.

[26] Mrs.S Thilagavathi and Dr.A Saradha. Impact Analysis of Dos & DDos Attacks. *IOSR Journal of Computer Engineering*, 16(6):24–33, 2014. ISSN 22780661. doi: 10.9790/0661-16642433.

[27] An Wang, Wentao Chang, Songqing Chen, and Aziz Mohaisen. Delving Into Internet DDoS Attacks by Botnets: Characterization and Analysis. *IEEE/ACM Trans. Netw.*, 26(6):2843–2855, 12 2015. ISSN 1063-6692. doi: 10.1109/TNET.2018.2874896. URL `https://doi.org/10.1109/TNET.2018.2874896`.

[28] Nazrul Hoque, Dhruba K Bhattacharyya, and Jugal K Kalita. Botnet in DDoS Attacks : Trends and Challenges. *IEEE Communications Surveys & Tutorials*, 17 (4):2242–2270, 2015. doi: 10.1109/COMST.2015.2457491.

[29] Zhu Zhaosheng, Judy Fu Zhi, Lu Guohan, Roberts Phil, Chen Yan, and Han Keesook. Botnet research survey. *Proceedings - International Computer Software and Applications Conference*, pages 967–972, 2008. ISSN 07303157. doi: 10.1109/COMPSAC.2008.205.

[30] Sheharbano Khattak, Naurin Rasheed Ramay, Kamran Riaz Khan, Affan A Syed, and Syed Ali Khayam. A Taxonomy of Botnet Behavior, Detection, and Defense. *IEEE Communications Surveys & Tutorials*, 16(2):898–924, 2014. ISSN 1553-877X. doi: 10.1109/SURV.2013.091213.00134. URL `http://ieeexplore.ieee.org/document/6616686/`.

[31] Esraa Alomari. Botnet-based Distributed Denial of Service (DDoS) Attacks on Web Servers : Classification and Art. 49(7):24–32, 2012.

[32] Polly Wainwright. An Analysis of Botnet Models. pages 116–121, 2019. doi: 10.1145/3314545.3314562.

[33] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Computing Surveys*, 39(1):3–es, 2007. ISSN 03600300. doi: 10.1145/1216370.1216373.

[34] Daniel R. Thomas, Richard Clayton, and Alastair R. Beresford. 1000 days of UDP amplification DDoS attacks. *eCrime Researchers Summit, eCrime*, pages 79–84, 2017. ISSN 21591245. doi: 10.1109/ECRIME.2017.7945057.

[35] Akamai. [state of the internet] / security. 4(5), 2018.

[36] Anchit Bijalwan, Mohammad Wazid, Emmanuel S. Pilli, and R.C. Joshi. Forensics of Random-UDP Flooding Attacks. *Journal of Networks*, 10(5):287–293, 2015. ISSN 1796-2056. doi: 10.4304/jnw.10.5.287-293.

[37] Vijay L. Kumar, Suresh Kumar, Anurag Srivastava, and Vijay Kumar. PING attack – How bad is it? 94(4):332–337, 2006. ISSN 10969098. doi: 10.1016/j.cose.2005.11.004.

[38] L. Kavisankar and C. Chellappan. A mitigation model for TCP SYN flooding with IP spoofing. *International Conference on Recent Trends in Information Technology, ICRTIT 2011*, pages 251–256, 2011. doi: 10.1109/ICRTIT.2011.5972435.

[39] R.K.C. Chang. Defending against flooding-based distributed denial-of-service attacks: a tutorial. *IEEE Communications Magazine*, 40(10):42–51, 10 2002. ISSN 0163-6804. doi: 10.1109/MCOM.2002.1039856. URL `http://ieeexplore.ieee.org/document/1039856/`.

[40] Zheng Wang. Analysis of Flooding DoS Attacks Utilizing DNS Name Error Queries. 6(10):2750–2763, 2012. doi: 10.3837/tiis.2012.10.018.

[41] Karanpreet Singh, Paramvir Singh, and Krishan Kumar. Application layer HTTP-GET flood DDoS attacks: Research landscape and challenges. *Computers and Security*, 65(October):344–372, 2017. ISSN 01674048. doi: 10.1016/j.cose.2016.10.005.

[42] L Rudman and B Irwin. Characterization and analysis of NTP amplification based DDoS attacks. In *2015 Information Security for South Africa (ISSA)*, pages 1–5. IEEE, 8 2015. ISBN 978-1-4799-7755-0. doi: 10.1109/ISSA.2015.7335069. URL `http://ieeexplore.ieee.org/document/7335069/`.

[43] Christian Rossow. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. (February):23–26, 2014. doi: 10.14722/ndss.2014.23233.

[44] Suk June Choi and Jin Kwak. A study on reduction of DDoS amplification attacks in the UDP-based CLDAP protocol. *Proceedings of the 2017 4th International Conference on Computer Applications and Information Processing Technology, CAIPT 2017*, 2018-Janua:1–4, 2018. doi: 10.1109/CAIPT.2017.8320670.

[45] Kulvinder Singh and Ajit Singh. Memcached DDoS Exploits: Operations, Vulnerabilities, Preventions and Mitigations. *Proceedings on 2018 IEEE 3rd International Conference on Computing, Communication and Security, ICCCS 2018*, pages 171–179, 2018. doi: 10.1109/CCCS.2018.8586810.

[46] Insight - DDoSMon. URL `https://ddosmon.net/insight/?last=90`.

[47] J.J. Cardoso de Santanna. *DDoS-as-a-Service: Investigating Booter Websites*. 2017. ISBN 9789036544290. doi: 10.3990/1.9789036544290. URL `http://purl.org/utwente/doi/10.3990/1.9789036544290`.

[48] Seul Ki Choi, Chung Huang Yang, and Jin Kwak. System hardening and security monitoring for IoT devices to mitigate IoT security vulnerabilities and threats. *KSII Transactions on Internet and Information Systems*, 12(2):906–918, 2018. ISSN 22881468. doi: 10.3837/tiis.2018.02.022.

[49] Computer Science and Software Engineering. International Journal of Advanced Research in Techniques to Differentiate DDOS Attacks from Flash Crowd. 3(6):295–299, 2013.

[50] Mousa Taghizadeh Manavi. Defense mechanisms against Distributed Denial of Service attacks: A survey. *Computers and Electrical Engineering*, 72:26–38, 2018. ISSN 00457906. doi: 10.1016/j.compeleceng.2018.09.001.

[51] Paul Ferguson. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing, 4 2000. URL `https://tools.ietf.org/html/rfc2827`.

[52] Jelena Mirkovic and Peter Reiher. D-WARD: A Source-End Defense against Flooding Denial-of-Service Attacks. *IEEE Transactions on Dependable and Secure Computing*, 2(3):216–232, 3 2005. ISSN 1545-5971. doi: 10.1109/TDSC.2005.35. URL `http://ieeexplore.ieee.org/document/1510618/`.

[53] Thomer M Gil and Massimiliano Poletto. Proceedings of the 10 th USENIX Security Symposium MULTOPS : a data-structure for bandwidth attack detection. 2001.

[54] B. B. Gupta and Omkar P. Badve. Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a Cloud computing environment. *Neural Computing and Applications*, 28(12):3655–3682, 2017. ISSN 09410643. doi: 10.1007/s00521-016-2317-5.

[55] Regional Internet Registries Statistics - RIR Delegations - World - Autonomous System Number statistics - Sorted by number, 4 2019. URL `https://www-public.imtbs-tsp.eu/~maigron/RIR_Stats/RIR_Delegations/World/ASN-ByNb.html`.

[56] Danny McPherson and Warren Kumari. Remote Triggered Black Hole Filtering with Unicast Reverse Path Forwarding (uRPF), 4 2009. URL `https://tools.ietf.org/html/rfc5635`.

[57] Christoph Dietzel, Anja Feldmann, and Thomas King. Blackholing at IXPs: On the Effectiveness of DDoS Mitigation in the Wild. In *Academic Writing*, pages 319–332. 2016. ISBN 9783319305042. doi: 10.1007/978-3-319-30505-9{_}24. URL `http://link.springer.com/10.1007/978-3-319-30505-9_24`.

[58] Pedro Marques, Robert Raszuk, Danny McPherson, Jared Mauch, Barry Greene, and Nischal Sheth. Dissemination of Flow Specification Rules, 4 2019. URL `https://tools.ietf.org/html/rfc5575`.

[59] Ziad El Jamous, Sohraab Soltani, Yalin Sagduyu, and Jason Li. RADAR: An automated system for near real-Time detection and diversion of malicious network traffic. *2016 IEEE Symposium on Technologies for Homeland Security, HST 2016*, pages 1–6, 2016. doi: 10.1109/THS.2016.7568889.

[60] Christoph Dietzel and Matthias Wichtlhuber. Stellar : Network Attack Mitigation using Advanced Blackholing. *CoNEXT*, (iv):152–164, 2018.

[61] Cisco. Implementing BGP Flowspec. pages 1–28. URL `https://www.cisco.com/c/en/us/td/docs/routers/crs/software/crs-r6-2/routing/configuration/guide/b-routing-cg-crs-62x/b-routing-cg-crs-62x_chapter_011.pdf`.

[62] Jeffrey C Mogul. WRL Research Report 89 / 4 Simple and Flexible Datagram Access Controls for Unix-based Gateways.

[63] Heikki Julkunen and C. Edward Chow. Enhance network security with dynamic packet filter. *Proceedings - 7th International Conference on Computer Communications and Networks, ICCCN 1998*, 1998-Octob:268–275, 1998. doi: 10.1109/ICCCN.1998.998786.

[64] iptables(8) - Linux man page, 2019. URL `https://linux.die.net/man/8/iptables`.

[65] M. Šimon, L. Huraj, and M. Čerňanský. Performance Evaluations of IPTables Firewall Solutions under DDoS attacks. *Journal of Applied Mathematics, Statistics and Informatics*, 11(2):35–45, 2018. doi: 10.1515/jamsi-2015-0010.

[66] L Huraj. Performance Evaluations of IPTables Firewall Solutions under DDoS attacks. 11(2):35–45, 2015.

[67] netfilter/iptables project homepage - The netfilter.org "nftables" project, 11 2018. URL `https://www.netfilter.org/projects/nftables/index.html`.

[68] Tomas Jonsson. LATENCY AND THROUGHPUT COMPARISON BETWEEN IPTABLES AND NFTABLES AT DIFFERENT FRAME AND RULE-SET SIZES LATENS- OCH Abstrakt. 2018.

[69] Petr Blazek, Tomas Gerlich, Zdenek Martinasek, and Jakub Frolka. Comparison of Linux Filtering Tools for Mitigation of DDoS Attacks. *2018 41st International Conference on Telecommunications and Signal Processing, TSP 2018*, pages 1–5, 2018. doi: 10.1109/TSP.2018.8441309.

[70] A thorough introduction to eBPF [LWN.net], 2018. URL `https://lwn.net/Articles/740157`.

[71] Steven McCanne and Van Jacobson. The BSD Packet Filter: A New Architecture for User-level Packet Capture. In *USENIX winter*, volume 46, 1993.

[72] Dominik Scholz, Daniel Raumer, Paul Emmerich, Alexander Kurtz, Krzysztof Lesiak, and Georg Carle. Performance Implications of Packet Filtering with Linux eBPF. In *2018 30th International Teletraffic Congress (ITC 30)*, volume 1, pages 209–217. IEEE, 2018.

[73] K. Salah, K. Sattar, M. Sqalli, and Ehab Al-Shaer. A potential low-rate DoS attack against network firewalls. *Security and Communication Networks*, 4(2): 136–146, 2 2011. ISSN 19390114. doi: 10.1002/sec.118. URL `http://doi.wiley.com/10.1002/sec.188http://doi.wiley.com/10.1002/sec.118`.

[74] Hung Jen Liao, Chun Hung Richard Lin, Ying Chih Lin, and Kuang Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013. ISSN 10848045. doi: 10.1016/j.jnca.2012.09.004. URL `http://dx.doi.org/10.1016/j.jnca.2012.09.004`.

[75] Kai Hwang and M. Gangadharan. Micro-firewalls for dynamic network security with distributed intrusion detection. *Proceedings - IEEE International Symposium on Network Computing and Applications, NCA 2001*, pages 68–79, 2001. doi: 10.1109/NCA.2001.962517.

[76] Martin Roesch. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th USENIX Conference on System Administration*, LISA '99, pages 229–238, Berkeley, CA, USA, 1999. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=1039834.1039864`.

[77] Aman Bakshi and B. Yogesh. Securing cloud from DDOS attacks using intrusion detection system in virtual machine. *2nd International Conference on Communication Software and Networks, ICCSN 2010*, pages 260–264, 2010. doi: 10.1109/ICCSN.2010.56.

[78] Alina Madalina Lonea, Daniela Elena Popescu, and Huaglory Tianfield. Detecting DDoS attacks in cloud computing environment. *International Journal of Computers, Communications and Control*, 8(1):70–78, 2013. ISSN 18419844.

[79] Chi Chun Lo, Chun Chieh Huang, and Joy Ku. A cooperative intrusion detection system framework for cloud computing networks. *Proceedings of the International Conference on Parallel Processing Workshops*, pages 280–284, 2010. ISSN 15302016. doi: 10.1109/ICPPW.2010.46.

[80] Md Khamruddin and Ch Rupa. A rule based DDoS detection and mitigation technique. *3rd Nirma University International Conference on Engineering, NUiCONE 2012*, pages 1–5, 2012. doi: 10.1109/NUICONE.2012.6493216.

[81] Pourya Shamsolmoali and Masoumeh Zareapoor. Statistical-based filtering system against DDOS attacks in cloud computing. *Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2014*, pages 1234–1239, 2014. doi: 10.1109/ICACCI.2014.6968282.

[82] Vern Paxson. Bro: A system for detecting network intruders in real-time. *Computer Networks*, 31(23):2435–2463, 1999. ISSN 13891286. doi: 10.1016/S1389-1286(99)00112-7.

[83] Shaohua Teng, Chaoyu Zheng, Haibin Zhu, Dongning Liu, Wei Zhang, and North Bay. A Cooperative Intrusion Detection Model for Cloud Computing Networks. 8(3):107–118, 2014.

[84] Yudha Purwanto, Kuspriyanto, Hendrawan, and Budi Rahardjo. Traffic anomaly detection in DDos flooding attack. *Proceedings of 2014 8th International Conference on Telecommunication Systems Services and Applications, TSSA 2014*, pages 1–6, 2015. doi: 10.1109/TSSA.2014.7065953.

[85] Mohammed Babiker, Enis Karaarslan, and Yasar Hoscan. Web application attack detection and forensics: A survey. *6th International Symposium on Digital Forensic and Security, ISDFS 2018 - Proceeding*, 2018-Janua:1–6, 2018. doi: 10.1109/ISDFS.2018.8355378.

[86] SpiderLabs/ModSecurity, 2019. URL `https://github.com/SpiderLabs/ModSecurity`.

[87] jzdziarski/mod_evasive, 2019. URL `https://github.com/jzdziarski/mod_evasive`.

[88] Mohamed Aly Mohamed and Nashwa Abdelbaki. HTTP Application Layer DDoS Attack Mitigation Using Resources Monitor. pages 213–221. 2018. ISBN 9783319648606. doi: 10.1007/978-3-319-64861-3{_}20. URL `http://link.springer.com/10.1007/978-3-319-64861-3_20`.

[89] Luis von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. CAPTCHA: Using Hard AI Problems for Security. In Eli Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, pages 294–311, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-39200-2.

[90] Hakem Beitollahi and Geert Deconinck. Tackling Application-layer DDoS Attacks. 10:432–441, 2012. ISSN 1877-0509. doi: 10.1016/j.procs.2012.06.056. URL http://dx.doi.org/10.1016/j.procs.2012.06.056.

[91] Silvia Bravo and David Mauricio. DDoS Attack Detection Mechanism in the Application Layer Using User Features. *2018 International Conference on Information and Computer Technologies (ICICT)*, pages 97–100, 2018. doi: 10.1109/INFOCT.2018.8356848.

[92] Alexander Endraca. Web Application Firewall (WAF). *International Journal of e-Education, e-Business, e-Management and e-Learning*, 3(6):451–455, 2013. ISSN 20103654. doi: 10.7763/IJEEEE.2013.V3.277. URL http://www.ijeeee.org/index.php?m=content&c=index&a=show&catid=41&id=595.

[93] Mattijs Jonker, Anna Sperotto, Roland van Rijswijk-Deij, Ramin Sadre, and Aiko Pras. Measuring the Adoption of DDoS Protection Services. pages 279–285, 2016. doi: 10.1145/2987443.2987487.

[94] Cheng Huang and Keith W Ross. Measuring and evaluating. *Keeping Found Things Found*, pages 210–241, 2008. doi: 10.1016/b978-012370866-3.50010-2.

[95] Martha Vazquez. IDC MarketScape: Worldwide DDoS Prevention Solutions 2019 Vendor Assessment, 2019. URL https://www.idc.com/getdoc.jsp?containerId=US43699318.

[96] Arbor Networks. Layered Intelligent DDoS Mitigation Systems.

[97] DDoS Protection Akamai, 2019. URL https://www.akamai.com/uk/en/resources/ddos-protection.jsp.

[98] Marek Majkowski. Meet Gatebot - a bot that allows us to sleep. *Cloudflare Blog*, 8 2017. URL https://blog.cloudflare.com/meet-gatebot-a-bot-that-allows-us-to-sleep.

[99] Marek Majkowski. Cloudflare architecture and how BPF eats the world. *Cloudflare Blog*, 2019. URL https://blog.cloudflare.com/cloudflare-architecture-and-how-bpf-eats-the-world.

[100] Matthew Prince. Introducing: I'm Under Attack Mode. *Cloudflare Blog*, 2 2012. URL https://blog.cloudflare.com/introducing-im-under-attack-mode.

[101] No Title, 11 2018. URL https://www.kernel.org/doc/Documentation/networking/filter.txt.

[102] Clang C Language Family Frontend for LLVM, 2019. URL https://clang.llvm.org.

[103] Tool Interface Standard. Executable and linking format (ELF) specification version 1.2. *TIS Committee*, 1995.

[104] bpf(2) - Linux manual page, 2019. URL http://man7.org/linux/man-pages/man2/bpf.2.html.

[105] Data Plane Development Kit, 2018. URL https://www.dpdk.org.

[106] Leonardo Linguaglossa, Dario Rossi, Salvatore Pontarelli, Dave Barach, Damjan Marjon, and Pierre Pfister. High-speed Software Data Plane via Vectorized Packet Processing (Extended Version). *Tech. Rep.*, 2017.

[107] Luigi Rizzo. Netmap: a novel framework for fast packet I/O. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 101–112, 2012.

[108] Dominik Scholz. Diving into Snabb. *Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)*, 31, 2016.

[109] Robert Dinh. A Tour Inside Cloudflare's G9 Servers. *Cloudflare Blog*, 2018. URL `https://blog.cloudflare.com/a-tour-inside-cloudflares-g9-servers`.

[110] HuubvanWieren/google_scholar_crawler, 2019. URL `https://github.com/HuubvanWieren/google_scholar_crawler`.

[111] Viet-Hoang Tran and Olivier Bonaventure. Beyond socket options: making the Linux TCP stack truly extensible. 2019. URL `http://arxiv.org/abs/1901.01863`.

[112] J Hong, S Jeong, J Yoo, and J W Hong. Design and Implementation of eBPF-based Virtual TAP for Inter-VM Traffic Monitoring. *2018 14th International Conference on Network and Service Management (CNSM)*, (HiPNet):402–407, 2018. ISSN 2165-963X.

[113] Tumolo Massimo. Towards a faster Iptables in eBPF. 2018.

[114] Matteo Bertrone, Sebastiano Miano, Jianwen Pi, Fulvio Risso, and Massimo Tumolo. Toward an eBPF-based clone of iptables. (October), 2018.

[115] Matteo Bertrone, Sebastiano Miano, Fulvio Risso, and Massimo Tumolo. Accelerating Linux Security with eBPF iptables. (August):108–110, 2018. doi: 10.1145/3234200.3234228.

[116] Stephen Hemminger. XNetEm Network Emulation using XDP.

[117] David Ahern. Leveraging Kernel Tables with XDP.

[118] Sebastiano Miano, Matteo Bertrone, Fulvio Risso, Massimo Tumolo, and V Mauricio. Creating Complex Network Services with eBPF : Experience and Lessons Learned. (October), 2018.

[119] Zeeshan Lakhani and Heather Miller. Clippy(ing) Network Functions: Towards Better Abstractions for Checking and Designing Network Programs. 2018. URL `http://arxiv.org/abs/1812.11145`.

[120] Jan Rüth, René Glebke, Klaus Wehrle, Vedad Causevic, and Sandra Hirche. Towards In-Network Industrial Feedback Control. pages 14–19, 2018. doi: 10.1145/3229591.3229592.

[121] Paul Chaignon, Diane Adjavon, Kahina Lazri, Jérôme François, and Olivier Festor. Offloading Security Services to the Cloud Infrastructure. pages 27–32, 2018. doi: 10.1145/3229616.3229624.

[122] Nguyen Van Tu, Kyungchan Ko, and James Won Ki Hong. Architecture for building hybrid kernel-user space virtual network functions. *2017 13th International Conference on Network and Service Management, CNSM 2017*, 2018-Janua: 1–6, 2018. doi: 10.23919/CNSM.2017.8256051.

[123] Mathieu Xhonneux, Fabien Duchene, and Olivier Bonaventure. Leveraging eBPF for programmable network functions with IPv6 Segment Routing. pages 67–72, 2018. doi: 10.1145/3281411.3281426. URL `http://arxiv.org/abs/1810.10247%0Ahttp://dx.doi.org/10.1145/3281411.3281426`.

[124] William Tu and Fabian Ruffy. Linux Network Programming with P4.

[125] Nicolaas Viljoen and Jakub Kicinski. Using eBPF as an Abstraction for Switching. URL `http://vger.kernel.org/lpc_net2018_talks/eBPF_For_Switches.pdf`.

[126] Jesper Dangaard Brouer. XDP – challenges and future work. 2.

[127] Arthur Fabre. XDP Based DoS Mitigation. (Figure 3).

[128] Sebastiano Miano, Roberto Doriguzzi-corin, Fulvio Risso, Domenico Siracusa, and Raffaele Sommese. High-Performance Server-based DDoS Mitigation through Programmable Data Planes. (iii):1–7.

[129] Sebastiano Miano, Matteo Bertrone, Mauricio Vásquez Bernal, Yunsong Lu, and Jianwen Pi. Securing Linux with a Faster and Scalable Iptables. 49(3), 2019.

[130] Ahamed Aljuhani, Talal Alharbi, and Hang Liu. XFirewall: A Dynamic and Additional Mitigation Against DDoS Storm. *Proceedings of the International Conference on Compute and Data Analysis*, pages 1–5, 2017. doi: 10.1145/3093241.3093252. URL `http://doi.acm.org/10.1145/3093241.3093252`.

[131] Daisuke Makita. A Study on Observation of DRDoS Attacks for Proactive Countermeasure and Real-time Response. (March), 2017.

[132] Muhammad Raheem, Muhammad Raheem, and Marco Chiesa. Mitigation of inter-domain Policy Violations at Internet eXchange Points Mitigation of inter-domain Policy Violations at Internet eXchange Points. 2018.

[133] Tom Barbette. Architecture for programmable network infrastructure. (May), 2018.

[134] Paul Emmerich, Maximilian Pudelko, and Georg Carle. Demystifying Userspace Packet IO Frameworks. pages 3–8.

[135] George Varghese. *Network Algorithmics,: An Interdisciplinary Approach to Designing Fast Networked Devices (The Morgan Kaufmann Series in Networking)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004. ISBN 0120884771.

[136] James Daly and Eric Torng. TupleMerge: Building online packet classifiers by omitting bits. *2017 26th International Conference on Computer Communications and Networks, ICCCN 2017*, 2017. doi: 10.1109/ICCCN.2017.8038399.

[137] HuubvanWieren/masterthesis, 8 2019. URL `https://github.com/HuubvanWieren/masterthesis`.

[138] IOVvisor BPF Compiler Collection (BCC), 8 2019. URL `https://github.com/iovisor/bcc`.

[139] Nguyen Xuan Vinh. Information Theoretic Measures for Clusterings Comparison : Variants , Properties , Normalization and Correction for Chance. 11: 2837–2854, 2010.

[140] C Pascoal, M R de Oliveira, R Valadas, P Filzmoser, P Salvador, and A Pacheco. Robust feature selection and robust PCA for internet traffic anomaly detection. In *2012 Proceedings IEEE INFOCOM*, pages 1755–1763, 2012. doi: 10.1109/INFCOM.2012.6195548.

[141] Fatemeh Amiri, MohammadMahdi Rezaei Yousefi, Caro Lucas, Azadeh Shakery, and Nasser Yazdani. Mutual Information-based Feature Selection for Intrusion Detection Systems. *J. Netw. Comput. Appl.*, 34(4):1184–1199, 2011. ISSN 1084-8045. doi: 10.1016/j.jnca.2011.01.002. URL `http://dx.doi.org/10.1016/j.jnca.2011.01.002`.

[142] Sample Captures, 2019. URL `http://tcpreplay.appneta.com/wiki/captures.html`.

[143] Tcpreplay - Pcap editing and replaying utilities, 2019. URL `https://tcpreplay.appneta.com`.

[144] tcpdump. Tcpdump/Libpcap public repository, 2019. URL `https://www.tcpdump.org`.

[145] majek. dump, 2 2018. URL `https://github.com/majek/dump/tree/master/how-to-receive-a-million-packets`.