FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF MASTER'S THESIS

Title:	Informed DDoS mitigation based on reputation
Student:	Bc. Tomáš Jánský
Supervisor:	Ing. Tomáš Čejka
Study Programme:	Informatics
Study Branch:	Computer Security
Department:	Department of Computer Systems
Validity:	Until the end of summer semester 2018/19

Instructions

Study the principles of Distributed Denial of Service (DDoS) attacks and mitigation techniques against them.
Study the current existing version of Nework Entity Reputation Database (NERD), which gathers information about suspicious network entities and computes so called reputation score.
Design a heuristic algorithm for cleaning the network traffic containing an attack against target address, focus on efficiency of (malicious/evil) packet discarding and number of created rules for discarding. The algorithm should select packets for discarding with priority according to information in NERD.
Create a software prototype of the designed algorithm.
Evaluate the created algorithm with synthetically generated data sets that represent various characteristics of DDoS attacks.
In the cooperation with the supervisor of the thesis, integrate the algorithm into a scrubbing system that is currently implemented in CESNET, a.l.e.

References

[1] http://nerd.cesnet.cz

prof. Ing. Róbert Lórencz, CSc.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague December 13, 2017

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Informed DDoS Mitigation Based on Reputation

Bc. Tomáš Jánský

Department of Computer Systems
Supervisor: Ing. Tomáš Čejka

May 7, 2018

Acknowledgements

I would like to thank Ing. Tomáš Čejka for guidance, counseling, advice, and comments while writing this thesis. I would further like to express my gratitude to *CESNET a.l.e.* for providing technical resources and also to my colleagues at *Liberouter* who did a great job developing the applied security tools. A special thanks belong to Mr. Pavel Šiška from the Brno University of Technology for providing technical support with the deployment of the scrubbing center. A special recognition for their mental support during my studies deserve my family, friends, and especially my girlfriend Gabriela Karfilátová. Last but not least, I would like to thank Mr. Hanz Zimmer whose music themes helped me to overcome desperate times while writing this thesis.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on May 7, 2018 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Jánský, Tomáš. *Informed DDoS Mitigation Based on Reputation.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.

Abstrakt

Mezi nejrozšířenější a zároveň nejnebezpečnější síťové útoky patří bezesporu distribuované útoky odepření služby (DDoS). Tyto útoky jsou neustálou hrozbou všem poskytovatelům internetového připojení a jsou schopny vyřadit z provozu síťovou infrastrukturu i těch největších společností. Cílem těchto útoků je zpravidla zahlcení síťového zařízení, nebo dokonce sítě samotné, pomocí velkého množství provozu. Následkem toho dochází k nepředvídatelnému zahazování síťových paketů. Obrana proti DDoS útokům na základě rozpoznávání škodlivých paketů je obtížnou výzvou, neboť ty se od legitimních paketů mnohdy liší jen minimálně. Tato práce se zabývá návrhem heuristiky, která filtruje síťový provoz během DDoS útoku a využívá k tomuto účelu znalosti tzv. *reputačního skóre* síťových entit. Hlavním přínosem práce je integrace takto navržené heuristiky do zařízení pro čištění síťového provozu vyvíjeného sdružením *CESNET z.s.p.o.*

Klíčová slova amplifikační DDoS, mitigace DDoS, reputační skóre, čištění provozu, víceklíčové řazení

Abstract

Network attacks, especially DoS and DDoS attacks, are a significant threat to all providers of services or infrastructure. The most potent attacks can paralyze even large-scale infrastructures of worldwide companies. The objective of DDoS attacks is usually to flood the target network device or even the network itself with a large number of packets. Such attack results in nondeterministic discarding of network packets. DDoS mitigation strategy based on the recognition of malicious packets is a complex task due to the similarity between legitimate and malicious packets. This thesis proposes a design of a mitigation heuristic which utilizes the knowledge of the so-called *reputation score* of network entities. The primary objective of this thesis is to integrate the proposed heuristic into a scrubbing center developed by *CESNET a.l.e.*

Keywords amplification DDoS, DDoS mitigation, reputation score, traffic filtering, multiple-key sorting

Contents

List of Figures

List of Tables

Introduction

Distributed Denial of Service (DDoS) attacks are among the most dangerous threats on the Internet because an efficient defense against them proves to be complex and expensive. The aim of attackers is to take down a service or even a network so it cannot be reached and used by legitimate users. There are many different types of DDoS attacks hence every mitigation technique addresses only a portion of them. The group of DDoS amplification attacks – notably their mitigation – represents nowadays a significant topic of interest of many researchers and security companies.

The overall traffic during a DDoS amplification attack consists of legitimate packets and malicious packets which are mostly meaningless packets with a sole objective – to flood a targeted device or a network. The malicious packets consume bandwidth and victim's other resources and consequently cause nondeterministic packet discarding. Naturally, many legitimate packets are discarded as well. Mitigation techniques aim to decrease the traffic volume so that it does not saturate the victim's links. However, it is necessary to reliably distinguish between legitimate and malicious traffic to avoid the disruption of legitimate connections. This task proves to be complicated, and a perfect method to distinguish between a legitimate and a malicious packet has not been discovered yet. Therefore, the current approaches to the mitigation of DDoS attacks are based on heuristics.

One of the primary objectives of this thesis is to design a new heuristic algorithm to mitigate DDoS amplification attacks. The proposed algorithm is supposed to discard most of the malicious traffic while trying to minimize the impact on legitimate connections. The heuristic utilizes the knowledge of the previous harmful behavior of network entities to accomplish mentioned goal. The previously observed malicious behavior of a network entity can be summarized into a single real number – formally known as the *reputation score*.

The second objective of this thesis is to integrate the proposed heuristic into a mitigation device developed by *CESNET a.l.e.* and replace the previous

mitigation heuristic, which is insufficient. *CESNET a.l.e.* is an operator of the Czech national research and education network (NREN). The system for mitigating DDoS attacks, called DDoS Mitigation Device (DMD), is deployed in the network `CESNET2` [1], which is a backbone network allowing high traffic rate – up to 100 Gbps.

The thesis is organized as follows: Chapter 1 introduces Distributed Denial of Service attacks, the related state-of-the-art mitigation techniques, and also presents security tools developed by *CESNET a.l.e* which are essential for completing the determined thesis objectives. Chapter 2 proposes design changes of DMD in order to replace the previously used mitigation technique with a new one called *RepTopN*. Chapter 3 focuses on describing specific implementation details of the enhanced mitigation system, especially the communication between DMD and the reputation database. Lastly, chapter 4 pursues the goal of evaluating the efficiency of the whole new mitigation system as well as the mitigation efficiency of the proposed *RepTopN* heuristic.

Chapter 1

State of the Art

This chapter introduces Denial of Service (DoS) attacks (especially the distributed version – DDoS) and briefly covers the history and modern trends of DDoS attacks. The chapter also presents a possible taxonomy of DDoS attacks and further focuses primarily on the characterization of DDoS amplification attacks. Research in the field of the possible defense and mitigation of DDoS amplification attacks is summarized in Sec. 1.1.3. Last but not least, the chapter describes the research and development of two security tools, namely:

- **Network Entity Reputation Database** – a reputation database described in Sec. 1.2.1
- **DDoS Mitigation Device** – a scrubbing center described in Sec. 1.2.2

The both mentioned security tools are developed by *CESNET a.l.e.* and are closely related to the primary objective of this thesis – the implementation of a mitigation system based on the knowledge of reputation score of network entities.

1.1 Distributed Denial of Service Attacks

The initiator of a cyber-attack known as Denial of Service aims to render a service, device, or even a network unavailable for its legitimate users. There are numerous ways for the attacker to achieve such desired state. The most common include:

- depletion of a network resource (e.g. bandwidth) by flooding the network with otherwise pointless packets,
- disruption of critical systems of the network infrastructure, such as Domain Name System (DNS) server which affects all the network users,

- exploitation of a specific feature or implementation bug of a protocol or application installed on the targeted device. Some protocols can be exploited in a way that may lead to depletion of the device resources (e.g., operation memory, CPU time, the maximum number of opened ports) or undefined behavior of the system (e.g., deadlock or even a crash).

If a DoS attack originates from multiple sources, it is called Distributed Denial of Service attack. As described in [2], DDoS attacks have two stages.

Firstly, the attacker must recruit other devices on the Internet by infecting them with malware which serves the attacker as a backdoor to the machine in the future. The malware typically consists of a communication unit for receiving commands from the attacker and a control unit which executes the commands. The infected device is called "*zombie*" and group of *zombies* forms *botnet*. There are multiple ways the malware spreads to other devices to acquire the sufficient number of *zombies*:

- the attacker actively scans the Internet for machines running an application with a known security hole which would allow the attacker to upload the malware,

- the malware spreads via another medium (e.g. USB drives), can be downloaded directly by a careless user in the form of an email attachment or disguise itself under the impression of a useful application,

- many devices connected to the Internet use weak or even default authorization credentials. This flaw is especially the case of the Internet of Things (IoT) devices. Recent data [3] show that the number of IoT devices connected to the Internet was about 8.4 billion in late 2017. The total number of connected IoT devices have increased by 31 % compared to the number in 2016, but experts suggest that the annual growth is most likely to rise even further due to the increasing popularity of IoT devices, despite the related security issues. The estimated number of IoT devices is 20 billion in 2020.

The infected device may also attempt to recruit other machines automatically. Such behavior speeds up the process of acquiring a massive *botnet*.

The second stage is the execution stage. The attacker specifies attack characteristics (e.g., type of the attack, duration, victim) and informs all the infected *zombies*. The communication between the attacker and *zombies* is not usually direct since the discovery of a single compromised machine would disclose the whole *botnet*. *Botnet* devices utilize some legitimate communication services – such as Internet Relay Chat (IRC) – to synchronize the attack. Usage of IRC makes the *botnet* particularly hard to identify since the discovery of a *zombie* may not lead further than the identification of the IRC

server and the channel group name. Other observed means of communication between the attacker and other zombies are Web-based (using Hypertext Transfer Protocol requests) or P2P-based.

1.1.1 Brief Evolution and Modern Trends

Although the first DDoS attacks date back to 1996 according to [4], one of the first notable cases of a DDoS attack which attracted the attention of general public happened in the year 2000. A 15-year-old hacker, who called himself *MafiaBoy*, launched a series of DDoS attacks and successfully managed to bring down multiple prominent websites including *Yahoo*, *Amazon*, or *CNN* for several hours [5]. The FBI identified the hacker after he repeatedly claimed the responsibilities for the attacks in IRC chatrooms and sentenced the hacker to one year in a juvenile detention center.

During the last couple of years, the intensity and the number of DDoS attacks has increased noticeably. *Brian Krebs* also confirms this trend in the article [6]. *Brian Krebs* is a respected journalist, an investigative reporter, and the owner of *Krebs on Security* website covering the activities of cybercriminals. Such status makes his website a frequent target of many DDoS attacks as depicted in 1.1.

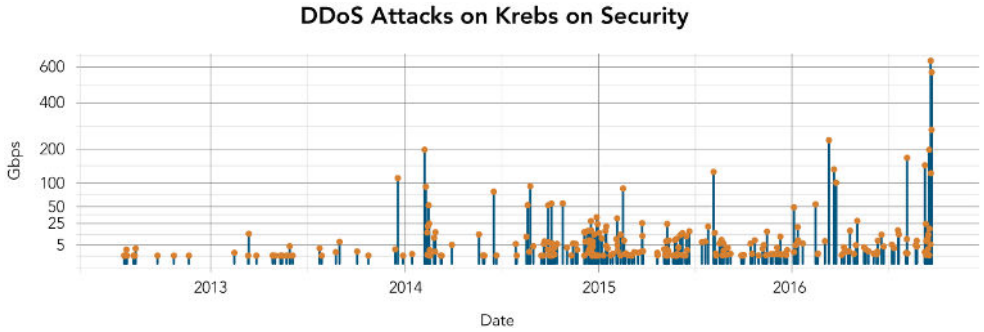

Figure 1.1: Visualization of the size and frequency of DDoS attacks against *KrebsOnSecurity.com* (source: [6])

A prominent example of the integration of IoT devices into a botnet is the infamous *Mirai* botnet which was first identified in August 2016 [7]. The *Mirai* botnet contained more than 400,000 devices and was offered for rent by the creator which lead to multiple high rate DDoS attacks in the following months. In October 2016, the DNS service provider *Dyn* was hit by a large-scale *Mirai* attack resulting in the decommissioning of hundreds of web pages for six hours including, for instance, *Twitter* or *Netflix*.

On February 28*th*, 2018, server *Wired* reported about a DDoS attack exceeding 1 Tbps in the article [8]. The target of the attack was the hosting service *GitHub*. The attack lasted about 9 minutes and peaked at stunning

1.35 Tbps. *GitHub* managed to withstand this example of a DDoS amplification attack – a specific type of DDoS attacks further described in Sec. 1.1.4. Only a week later a 1.7 Tbps DDoS attack was reported in [9]. The attack used the same pattern and was aimed at a US-based service provider. This DDoS attack is the largest one that has ever been reported at the time of writing this thesis.

It is evident that DDoS attacks have rapidly evolved in the last 20 years – not only technically but also in the sense of attackers' incentives. What began as an attempt of individual hackers to prove themselves in the community quickly turned into a large business. Currently, *botnets* are marketed as online services which can be used to execute DDoS attacks by anybody who is willing to pay the price in tens of dollars. According to [10, 11], the most common motives of attackers include:

- *Financial gain:* This category covers the majority of attacks targeted against corporations. The attackers are paid by corporations to attack other competing corporations or may demand a ransom (usually in the form of cryptocurrency, such as *Bitcoin*) from the victim to stop their DDoS attack against the victim.

- *Ideological belief:* Attackers motivated by their ideological belief including "*hacktivist*" groups, such as *Anonymous* which are responsible for several significant DDoS attacks.

- *Intellectual challenge:* A relatively small category usually consisting of hacking enthusiasts who try to improve their skills or prove themselves in a community.

- *Cyberwarfare:* Military or terrorist organizations usually targeting government infrastructure of other countries. Attackers in this category probably posses vast expertise, are well trained, and dispose of the vast amount of resources provided by their organizations or countries.

- *Cover for targeted attacks:* A trend occurring more frequently in last few years. A DDoS attack should divert attention from additional compromise, i.e., viruses, ransomware, and other malware.

1.1.2 Taxonomy

There are many possible classifications of DDoS attacks. Some of them are presented by *Mirkovic et al.* [2], e.g., classification by the exploited weakness, the validity of the attack source, victim type, or communication with devices forming the *botnet*. What *Mirkovic et al.* classify as *brute-force* attacks, *Zargar et al.* [10] further divide into smaller categories to describe individual weaknesses of transport and application layer protocols which can be exploited by

the attackers to perform a DDoS flooding attack. Some of the most commonly practiced DDoS attacks include:

L3 flooding attacks

Attacks which utilize transport layer network protocols, namely Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Internet Control Message Protocol (ICMP).

- *Network flooding attacks:* Attacks focused on flooding a target network using meaningless packets, thus exhausting bandwidth of the network. Packets of legitimate users are most likely discarded in the affected network. These attacks are primitive to realize but can be highly effective. UDP flood, which resides in sending an immense number of UDP packets on random ports of the victim, is a typical example of this category. The victim realizes that no application listens on the specified port and replies with an ICMP packet. If many hosts are targeted, the combination of UDP and ICMP packets can consume all bandwidth.

- *Protocol exploitation flooding attacks:* Attackers exploit specific features or implementation bugs of L3 protocols to deplete specific resources of the victim. Some TCP design features make this protocol particularly vulnerable to exploitation. For instance, TCP SYN flood takes advantage of the TCP "*three-way handshake*" by not initiating the TCP session with the victim completely. Such behavior results in binding resources of the server for a certain amount of time and may eventually deplete all available resources of the victim. Therefore, legitimate users are not able to establish a connection with the target server.

- *Reflection-based flooding attacks:* A reflector is an IP host that replies when it receives a packet. Attackers may abuse reflectors by sending them requests (e.g. ICMP echo request) with forged source IP address (this technique is called IP address "*spoofing*"). If the attacker fills the source IP address with the victim's IP address, the reflector replies to the specified forged IP address. Attackers using reflectors are much harder to trace because, among others, a cooperation between the administrator of the reflector and the administrator of the target device is necessary. For instance, *Smurf* attack belongs to this category. The attacker broadcasts a large number of ICMP packets in the network with victim's spoofed source IP address. If the devices in the network are not configured to ignore such packets, they direct their response to the victim, thus exhausting its resources.

- *Amplification-based flooding attacks:* Attacks which fall into this category are a variation of the reflection-based attacks. The key difference is that the response from the reflector is considerably larger than the

packet sent by the attacker, or the reflector sends multiple responses. DDoS amplification attacks are the focus of this thesis; therefore, are further elaborated in Sec. 1.1.4.

L7 flooding attacks

Attacks based on the exploitation of particular features of application layer protocols (e.g., Session Initiation Protocol – SIP, DNS, HTTP). These attacks focus on exhausting the victim's resources (e.g., sockets, CPU, memory, disk/database bandwidth, and I/O bandwidth) rather than consumption of the network bandwidth. L7 flooding attacks are generally harder to detect since they consume less bandwidth and the malicious packets are almost indistinguishable from the legitimate ones. However, the impact of these attacks is similar to those in using transport layer protocols and appear to be popular among the community of attackers.

- *Reflection/amplification-based flooding attacks:* The employed attack technique is similar to the technique of reflection/amplification attacks on the transport layer except for the exploited protocols. The most common attacks such as DNS flood and VoIP flood are thoroughly described in Sec. 1.1.4.

- *HTTP session/request flooding attacks:* Attacks in this category can be characterized with a high rate of HTTP requests oriented towards the target HTTP server either in one or multiple sessions. The attacked server eventually runs out of available resources due to the sheer number of HTTP requests. Therefore, the request of legitimate users cannot be processed or are processed with an unacceptable delay. The examples include "*excessive VERB*" and "*excessive VERB single session*" attacks.

- *HTTP Slow request/response attacks:* The distinct characteristic of these attacks is their slow communication with the HTTP server. All attacks in this category attempt to exhaust the maximum number of opened sockets. Attackers focus on establishing as many sessions with the HTTP server as possible while trying to keep all the sessions in progress, so the sockets never close. This effect can be achieved in many ways, e.g., slowly updating the HTTP request headers (*Slowloris attack*), fragmenting the request packets and sending individual fragments just before a timeout would occur (*HTTP fragmentation attack*), or sending one byte of payload at a time after specifying huge payload in the appropriate HTTP header field (*R-U-Dead-Yet attack*).

1.1.3 Mitigation Techniques

DDoS attacks have been known for more than 20 years. The nowadays frequency of DDoS attacks prove that a reliable defense mechanism against DDoS attacks has not been invented yet, despite the long attempt of researchers to improve the situation. *Mirkovic et al.* in [2] debate the reasons why the defense against DDoS attacks is difficult. The main issue is related to the design of intermediate networks which provide best-effort packet forwarding service but leave the deployment of security features and advanced protocols to the communication endpoints. Every network has its own security policies since the control over the Internet is distributed. Therefore, it is not possible to enforce network managers to accept certain security policy or deploy particular security mechanism. The situation can improve if the systems directly connected to the Internet are up-to-date and follow current security recommendations. However, the reality is opposite which is proven by the number of vulnerabilities from which IoT devices currently suffer. Lastly, DDoS attacks are infamous because of the anonymity of attacker due to the IP address "*spoofing*" technique (forging the IP address of the packet originator).

Specific techniques to identify attackers with spoofed IP address have been summarized in [12]. A commonly used technique is *Hop-by-Hop IP Traceback* which resides in accessing the router closest to the victim and determining the upstream router which is forwarding the malicious packets. It is possible to recursively discover the attack originator if the originator is located inside the administrated network. The network borders are the main limitation of such techniques. If the attack originates beyond the borders of the Internet Service Provider (ISP), cooperation between multiple ISPs is necessary to trace the source of the attack, and that is only possible if all ISPs possess the necessary traceback technology. Even if one or more sources of the attack are discovered, they are most likely just infected devices, and the identity of the master device controlling discovered zombies remains unknown.

DDoS defense mechanisms usually consist of a detection heuristic and a mitigation strategy in case an attack is detected. The options to mitigate a DDoS attack are limited once it occurs. *Mirkovic et al.* in [2] present four basic response strategies to mitigate the ongoing attack.

- *Agent Identification:* Various traceback methods which rely on successful identification of the actual attack source fall into this category – such as *packet marking*, *link testing*, or *Hop-by-Hop IP Traceback* described in the paragraph above. Other approaches may utilize the actual IP address of the attacker to reduce the impact of the attack. The main arguments for using such techniques are to make the actual source of malicious packets accountable for the actions and cleaning the attacking device of any malware afterward. The disadvantages of these mechanisms include, e.g., limited accuracy, computational and network

overhead, and the necessity to have a sufficient number of routers that support traceback before it is effective.

- *Rate-Limiting:* Systems which utilize rate-limiting mitigation strategy place a restriction on the maximum number of allowed packets originating from the sources which have been identified as malicious by a certain detection heuristic. Such restriction is not necessarily imposed on all traffic originating from the misbehaving IP address but rather on packets carrying a specific protocol or aiming at particular ports. This mitigation strategy is usually deployed when the outcome of the used detection heuristic is inaccurate in characterizing the ongoing attack precisely. The downside of this technique is that it also allows some of the malicious traffic to reach the victim.

- *Filtering:* Traffic filtering completely discards unwanted traffic streams in the network. This response to a DDoS attack might be risky because it can accidentally disrupt connections of legitimate users if the detection mechanism produces false positives. Dynamically configured firewalls often utilize this strategy.

- *Reconfiguration:* Reconfiguration is a less popular strategy which resides in the reconfiguration of the network topology once an attack is observed. This strategy aims to either add more resources to the victim or to isolate the attack machines.

Zargar et al. [10] present a classification of various DDoS detection methods based on the deployment location. It is evident that the portion of DDoS traffic is the highest in the attack destination. Therefore, the accuracy of DDoS detection mechanisms increases with the decreasing distance between the deployed DDoS detection mechanism and the victim of the DDoS attack. The authors emphasize the importance of detecting the DDoS attack as close to its source as possible before it depletes resources of DDoS defense mechanisms.

An example of source-based defense mechanism is *Ingress/Egress* filtering. *Ingress* respectively *Egress* filtering is a filtering mechanism of packets with spoofed IP addresses forwarded *in* respectively *out* of the internal network. Such filtering can be achieved thanks to the knowledge of peering networks. However, the mechanisms are inapplicable in transit networks and also worthless if the spoofed IP address resides in the valid internal/external IP address range. Other source-based detection algorithms include, e.g., comparing both inbound and outbound traffic to predefined models, comparing the proportion of inbound and outbound traffic rate, and reverse firewall – a firewall which limits the forwarding rate of packets that do not reply to other packets forwarded in the other direction.

Dozens of DDoS detection mechanisms have been already proposed. Most of them rely either on the detection of statistical abnormalities in observed traffic or the recognition of malicious packets.

Statistical-based approaches

A continuous observation of statistics about various aspects of the network traffic can aid in discovering specific DDoS attacks which are not characterized by the increase in traffic volume. Most of the observed DDoS attacks have a typical pattern describable with certain statistical abnormalities in the observed traffic. Mapping specific statistics about ICMP, UDP, and TCP packets to a predefined model of a specific DDoS attack may lead to identifying the nature of the attack. Having such information during an ongoing DDoS attack greatly helps in choosing the suitable mitigation strategy.

Recognition of malicious packets

These mechanisms usually utilize various characteristics of individual packets placed in a broader context (e.g., the amount of traffic originating from the particular IP address or the IP address behavior history) to decide whether the packet is malicious or not. The main disadvantage of such mechanisms is that they are partially dependent on the scale and strength of the attacks which unfortunately may result in the depletion of resources of the defense mechanism (e.g., CPU time or memory).

- *Hop-count filtering:* Most of the attackers attempt to spoof their IP address to avoid accountability for their actions. However, IP address spoofing can also backfire against the attackers. Such is the case of using *hop-count filtering* technique which resides in storing the information about the source IP addresses and their corresponding hop-count when the device or network is not under attack. Afterward, during the attack, the defense mechanism recognizes spoofed packets thanks to the deviation between the source IP address and hop-count pair present in the traffic and the expected value observed before. However, it is not recommended to rely solely on this solution because attacks without the use of IP address spoofing would carry out. Moreover, this solution may result in disrupting legitimate connections if the topology of the network changes.

- *History-based IP filtering:* These mechanisms filter inbound traffic during a DDoS attack and usually follow one of two rules: either allow only IP addresses which were previously seen communicating with the victim or discard traffic of IP addresses which were previously observed participating in other attacks. Such solutions are especially useful against flooding attacks because it helps hosts in resource management during a

DDoS flooding attack. An example which falls into this category is presented by *Peng et al.* in [13]. Their proposed solution keeps track of IP addresses which frequently communicate with the protected host. When a DDoS attack occurs, the defense mechanism allows only the packets of IP addresses which were already seen during a typical network state. *CESNET a.l.e.* is developing the solution which utilizes history of misbehaving IP addresses which is further described in Sec. 1.2.1.

A popular way to implement destination-based defense mechanisms is to integrate them into a scrubbing center [14]. A scrubbing center is a centralized data cleansing station usually equipped with multiple different algorithms utilizing deep packet inspection to detect known attacks and exploits (e.g., SQL injection, XSS, or DDoS). The scrubbing center is dedicated to handling high volume DDoS floods, but it is also capable of protecting network users from slow DDoS attacks, such as *Slowloris*. The scrubbing center is usually passive, and the traffic to the scrubbing center is redirected only when an attack is detected.

1.1.4 Amplification DDoS Attacks

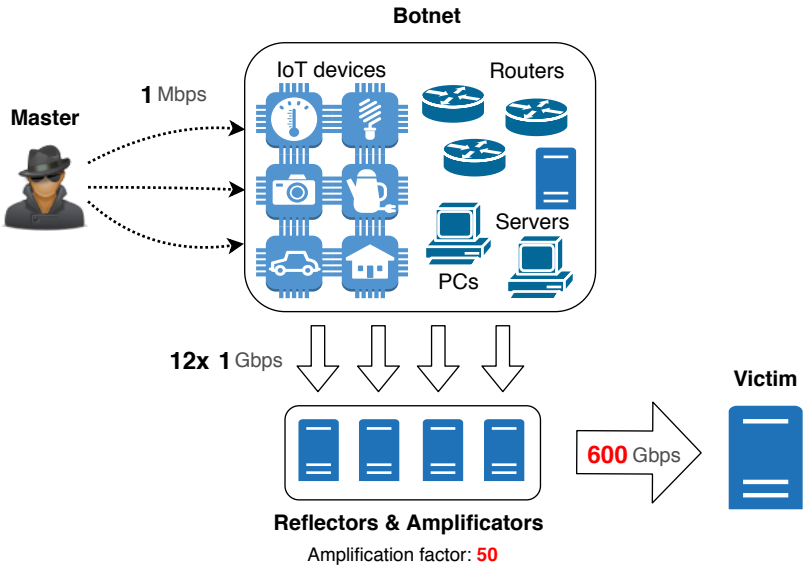

Figure 1.2: 600 Gbps amplification DDoS attack.

Previous section discusses some possibilities to trace a packet stream back to its source even if the IP address is spoofed. Such packet tracing techniques are worthless if the attacker decides to use reflectors to bounce off the attack rather than sending the packet stream directly to the victim. A reflector is an

Table 1.1: Amplification factors of certain abusable protocols. (Source: US-Cert [16])

Protocol	Amplification factor
Domain Name System	25 to 54
Network Time Protocol	556.9
Lightweight Directory Access Protocol	46 to 55
Steam Protocol	5.5
BitTorrent	3.8
Simple Service Discovery Protocol	30.8
Memcached	10,000 to 51,000

IP host that replies when it receives a packet. If the attacker fills the source IP address with the victim's IP address, the reflector replies to the specified forged IP address. The victim does not require any traceback technique to identify the sources of the attack since the source IP addresses present in reflected packets are valid.

Furthermore, the response from the reflector can be considerably larger in terms of bytes or packets than the message sent by the attacker. The attack is therefore amplified, hence the terms *amplificator* and amplification DDoS attacks. Figure 1.2 depicts the usual course of the DDoS amplification attack. The attacker who is in control of the botnet (master) commands devices in the botnet to target previously discovered amplificators. Every zombie spoofs its IP address with the address of the victim. Therefore, the amplificator aims its reply to the victim because it assumes the victim is the host who issued the request. There are number of possible protocols which can serve attackers' purpose to amplify the attack. Each protocol has a different amplification factor; however, the attacker must also find enough of vulnerable reflectors supporting the chosen protocol to fully utilize the attack potential. Some of the abusable protocols including the way of abuse to gain the desired amplification effect are listed in [15]. Table 1.1 provides amplification factors of some selected protocols.

The amplification factor ranges in the order of tens or hundreds for most of the protocols – except for *Memcached*. Memcached is a distributed memory object caching system often utilized to speed up dynamic web applications. The misuse of improperly configured Memcached servers was the root of the two largest DDoS attacks which are mentioned in Sec. 1.1.1. Although the title of the most abused protocol held NTP and DNS for many years, recent data published by *Apnic* [17] suggest that attackers have started to incline to other protocols. However, the Internet consists of many obsolete devices which support outdated protocol versions. Hence, the threat of certain attacks is still present even though those means of abuse have already been resolved.

1.2 Related Security Tools

One of the contributions of this thesis is the application of reputation scores from **Network Entity Reputation Database** in the scrubbing center **DDoS Mitigation Device** for the purpose of informed mitigation of DDoS amplification attacks. This section briefly summarizes research about reputation scores and discusses the relevance of the usage of reputation scores in the scrubbing center. Furthermore, the section introduces both systems and describes their main principles.

1.2.1 Network Entity Reputation Database

It is a known fact that some network entities are more likely to attack than others. The most likely attackers are presumably hosts infected with malware. The affected hosts stay compromised for a certain amount of time. Thus, it is probable to observe more security events originating from the affected hosts until the malware is removed. The fact that the compromised hosts evince such behavior is often used for assembling lists of known malicious IP addresses (blacklists), which are shared among the Internet community to block the known attackers (e.g. blocking known sources of spam [18]). Moreover, according to the articles by Shue et al. [19] or Moura [20, 21], the distribution of malicious sources on the Internet is not uniform, and the sources exhibit some spatiotemporal correlations. The work of Moura [21] suggests that not only some hosts are more likely to attack than others, but the same can be stated about larger entities (e.g., network prefixes, ISPs, and even countries). Although his work focuses primarily on discovering the distribution of spam sources, the presented concept of "Internet Bad Neighborhoods" can be further generalized to other types of attacks. The spatiotemporal correlations in other attack types (e.g., scans, ssh brute-force, syn floods) are examined by Bartoš et al. in the study [22]. The authors discovered various correlations even for a local network and proposed that such analysis should be the basis for entity reputation modeling techniques.

The term **reputation score** was first introduced by Bartoš et al. in [23], and is formally defined as follows:

> "*Reputation score of a network entity (e.g. an IP address) represents the probability that the entity will perform a malicious activity in the near future (e.g. next 24h), based on its past behavior and other information.*"

The authors argue that – to achieve a reliable value of the reputation score – the input of the prediction algorithm should be mainly a summary of all detected security incidents in a past time window (e.g. a month) and recommend the alert-sharing systems as suitable input data sources. However, based on the previous findings regarding spatial correlations between attackers, the incident history can be supplemented by various other inputs, such as

blacklists, geographical location, an autonomous system the entity belongs to, or the reputation scores of other entities with the same network prefix. The output of the prediction algorithm should be the probability that the given entity behaves maliciously in the near future. Due to the sheer complexity of the prediction algorithm and the number of possible factors which influences the reputation score value, the authors' approach utilized supervised machine learning methods to infer the algorithm from the data. The results presented in their work show that the created predicator accurately estimates the probability of future attacks of each evaluated entity, thus experimentally proving that the machine learning methods are feasible for such task.

The reputation score can be utilized in various ways including, for instance, alert prioritization or attack prediction, which are both essential methods for coping with the increasing number of adversaries. Another possible use-case of the reputation score is to assemble highly predictive blacklists frequently. The size and restrictiveness of the blacklist can be controlled by the user – either by taking a fixed number of the worst IP addresses or taking all IP addresses with reputation score higher than a fixed threshold. Assuming the probability estimation is accurate, it is guaranteed that such blacklist has the highest hit count possible with the given length of the list. Such use-case of the reputation score poses the topic of interest of this thesis – to utilize the reputation score to identify probable attackers during a DDoS amplification attack.

Different approaches for blacklist construction based on the evaluation of the previous behavior of IP addresses were also proposed by various other researchers, for instance, [24, 25].

Network Entity Reputation Database[1] (NERD) is a system developed by *CESNET a.l.e.* which serves as a proof of concept in the research regarding reputation score modeling. NERD consists of two parts – the entity database and the entity scoring. The entity database keeps information about network entities (such as IP addresses, subnets, hostnames, autonomous systems, BGP prefixes) that were reported as malicious by some of the contributing data sources via a *CESNET's* alert-sharing system *Warden*.[2] However, any source of such data is possible. The database records are further enriched by information from DNS and whois systems, various public blacklists and other relevant databases. It aims to maintain a global database of known malicious sources on the Internet, including all security-relevant information related to them.

The main feature of NERD is evidently the ability to compute the reputation score for every single entity in the database to determine the probability the entity behaves maliciously in the next 24 hours. NERD also computes

[1] https://nerd.cesnet.cz/
[2] https://warden.cesnet.cz/

the reputation score in different contexts, each predicting a different class of attacks. Thus, each IP address gets assigned multiple reputations scores such as RS_{scan} or RS_{ddos}. NERD also aggregates the reputation score of individual IP addresses to a reputation score of various IP address groups, such as BGP prefixes and ASNs.

NERD is used as the source of reputation scores utilized in the proposed mitigation heuristic since it is free and open on how it works, i.e., the list of its data sources and methods for their processing is well documented which may not hold for other reputation databases. However, any other reputation system or database providing similar reputation metric may also be used for the proposed mitigation heuristic.

1.2.2 DDoS Mitigation Device

CESNET a.l.e. has started to develop its own scrubbing center called DDoS Mitigation Device (DMD) that can operate at 100 Gbps. The scrubbing center is presented, for instance, in [26] or at webpage [27]. DMD is a commodity hardware server with a dedicated network interface card equipped with an FPGA chip. The DMD architecture is depicted in Fig. 1.3.

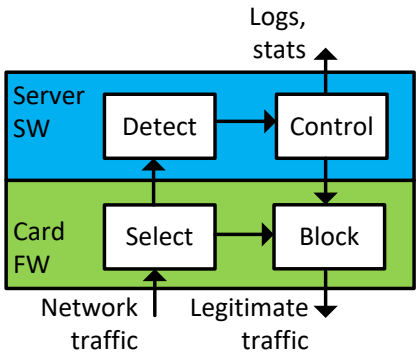

Figure 1.3: High-level architecture of DDoS Mitigation Device.

The FPGA chip implements fast-forwarding and filtering data plane that is controlled by a software controller which continuously evaluates network traffic parameters. As soon as a DDoS attack is detected, a heuristic algorithm identifies offending IP addresses and the controller enables packet filtering in FPGA immediately. The whole process of analyzing network traffic parameters, applying mitigation heuristic and updating filtering rules in FPGA is called **mitigation cycle**, and repeats every second in the default settings.

DMD utilizes several heuristics to select packets that ought to be discarded in order to protect the specified network prefixes during DDoS attacks. This thesis is mainly focused on the mitigation heuristic for the specific family of DDoS attacks called reflective amplification attacks which are described in Sec. 1.1.4. In case of these attacks, the reflected packets hitting a victim contain valid (i.e. not spoofed) source IP addresses of the reflectors. Therefore, blocking the traffic of the particular source IP addresses is a sound strategy. However, the crucial part is to identify only the IP addresses of the reflectors correctly.

The approach of DMD to reflective attacks is to reduce the attack to an acceptable intensity which can be handled by the victim server without its outage. To this end, an operator of DMD defines a set of rules per each protected network prefix (also called **amplification rules**). Such rule consists of three parts: condition, limits and optimal traffic rates. The condition part describes which packet can match the rule (e.g., `dst_IP` must match IP prefix and `dst_port` number of a packet must match the `dst_port` number stated by the condition). The limits define the number of packets/s or bytes/s that must be exceeded to apply filtering of the traffic targeting the protected IP prefix. Optimal traffic rates define the desired number of bytes/s or packets/s the DDoS traffic should be reduced to.

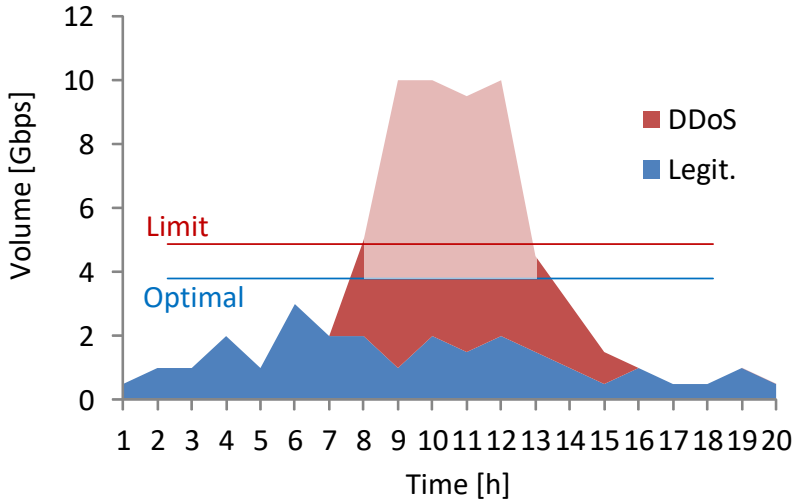

Figure 1.4: Example of mitigation principle by DDoS Mitigation Device.

Upon the detection of a DDoS attack, the DMD mitigation algorithm comes into play. The algorithm blocks traffic arriving from IP addresses producing the most (by packets or by bytes) traffic, i.e., blocks *top-n* IP addresses where n is selected so that the optimal traffic rate is achieved. Figure 1.4 shows

the application of a rule utilizing the plain *top-n* approach mitigation, provided the traffic of all *top-n* addresses belong to the attack, which is not always the case in real-world situations. Therefore, a portion of the traffic of legitimate users might be discarded along with the malicious traffic in particular situations. It is the primary objective of this thesis to design an extension of the presented mitigation technique which would reduce this undesired effect.

Chapter 2

Analysis and Design

This chapter thoroughly summarizes requirements placed on the new mitigation heuristic algorithm, presents several assumed changes in the DMD design, introduces the main idea of the new mitigation heuristic and describes in detail the whole new mitigation process called *augmented mitigation*.

2.1 Requirements Analysis

Before designing a new mitigation heuristic algorithm for DDoS amplification attacks, it is necessary to identify requirements resulting from the purpose of the algorithm as well as from the deployment of the system. The following requirements must be taken into consideration in the heuristic design:

- The algorithm receives data from DMD in the form of a table where each row consists of an IP address and its contribution to the traffic. The heuristic utilizes the reputation scores from NERD for its decision-making process of selecting suitable IP addresses to block. The output data consists of a list of IP addresses which packets should be discarded used by DMD to mitigate the ongoing DDoS attack.

- The algorithm must meet time constraints to fit into a single mitigation cycle of DMD even for DDoS attacks reaching up to 100 Gbps magnitude (which is the current limit of the network interface card with FPGA).

- The algorithm must assume uncommon extreme scenarios such as when the reputation scores are not present during a current mitigation cycle (e.g. communication error with NERD) and always discard traffic deterministically (e.g. utilize the currently implemented *top-n* algorithm as the mitigation heuristic).

- The algorithm must be easily merged with the current implementation of DMD implemented in *C* programming language.

2.2 Augmented Mitigation

The proposed method is designed to work under the following conditions:

1. The source IP addresses in the DDoS attack are not spoofed.
2. The reputation score of an IP address in NERD corresponds to the likelihood that it attacks in nearby future.
3. The IP addresses not present in NERD are assigned zero reputation score and are considered as generally less likely to generate malicious traffic contrary to those present in NERD.

The first condition originates from our focus on reflection attacks (other mitigation methods are more suitable for attacks with randomly spoofed source addresses). The other two conditions may not hold in practice if the data in NERD are of low quality. In such case, the proposed algorithm may give suboptimal results depending on the scale of violation of these conditions. This issue is further elaborated in Sec. 4.3.3.

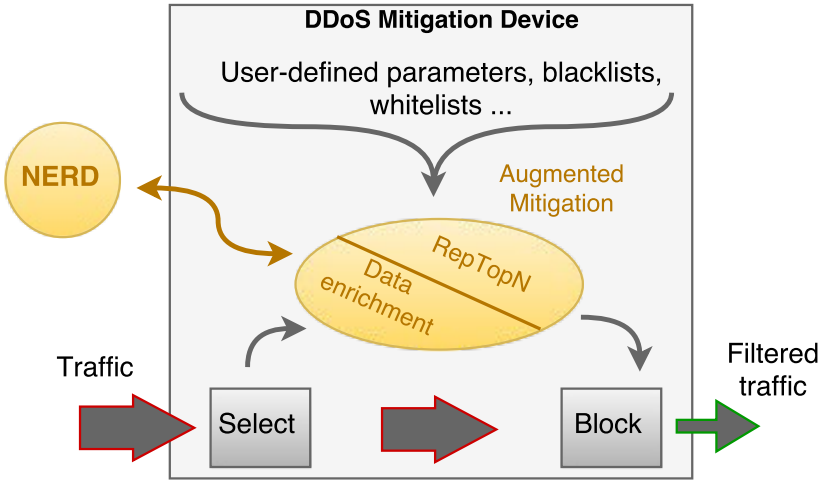

Figure 2.1: DDoS Mitigation Device enhanced with augmented mitigation.

Figure 2.1 depicts the proposed extensions to the existing mitigation system. The main idea of DMD remains intact (described in Sec. 1.2.2). The **mitigation cycle** still consists of the traffic sample selection, the creation of a set of blocking rules based on used heuristic and the rules update in FPGA. Only the process of creation of the set of blocking rules was modified and split into two parts: *data enrichment* and *RepTopN*. The *data enrichment* stage of the mitigation cycle handles the processing of the data sample received from live traffic, communication with NERD and subsequent enrichment of the traffic sample with the corresponding reputation scores. The enriched

data from this phase are passed to the next stage and processed with the *RepTopN* heuristic. The *RepTopN* algorithm replaces the previously used *top-n* algorithm, and unlike the *top-n* algorithm, *RepTopN* utilizes both the traffic volume contribution and the reputation score from NERD of each IP address present in the live traffic sample. The desired effect is to prioritize discarding of packets from the sources with non-zero reputation score and thus possibly preserve legitimate connections, which would otherwise be disrupted.

2.2.1 RepTopN Heuristic

DMD previously blocked the most contributing IP addresses during an attack. Traffic filtering based solely on the traffic volume contribution of each IP address is not efficient in certain cases. If the attack consists of many attackers and the volume contribution of each attacker is not significant, the attackers might not be sufficiently distinguishable from the perspective of the traffic volume contribution. In such cases when the number of attackers is immense, the probability of disrupting legitimate communication rises considerably even to the point in which the most of the discarded traffic consists of legitimate traffic.

The main idea behind the *RepTopN* heuristic is that the traffic sources which were recently observed performing a malicious activity are more likely to attack again than the sources with no attack history. This likelihood is represented in the form of the reputation score. By prioritizing IP addresses with non-zero reputation score in the discarding process, the mitigation system ought to achieve a lower amount of discarded legitimate packets.

However, it is highly improbable that every attacker in the ongoing attack was reported before, thus having a non-zero reputation score. Therefore, it might not be sufficient to discard the traffic of IP addresses with non-zero reputation score to reach the desired traffic rate. Regarding this, the combination of the reputation score and the traffic contribution must be utilized to make the traffic discarding deterministic.

It is worth noting that the reputation score in NERD represents the likelihood of an entity to attack in nearby future and not the probability that the observed packets originating from the entity are malicious. For the purpose of this research it is possible to consider both interpretations equal since it is generally more likely that the higher the reputation score of a source is, the more likely is the traffic of the source malicious during an attack. However, this slight difference may be manifested in the following scenario. A company administrates own DNS server. The traffic originating from the DNS server is valid in the vast majority of cases. Nevertheless, once a day the DNS server is used as a reflector for a DDoS attack. The DNS server would have a high reputation score even though the most of its produced traffic is legitimate throughout the day. Naturally, the packets originating from the mentioned DNS server are present in the network traffic all the time. When a DDoS

attack occurs, blocking of this DNS server on the basis of the high reputation score might result in disrupting legitimate communication even though the DNS server is not misused in the ongoing attack. Due to these extreme cases, *RepTopN* offers a possibility to improve its decision-making process by including blacklists and whitelists provided by the system administrator.

The method of an appropriate prioritization based on multiple attributes is a complicated issue, and multiple approaches are possible to design a solution. The proposed approach is **multiple-key sorting** which has multiple advantages. Since DMD operates on networks with a high traffic rate – up to 100 Gbps, the processing time of the heuristic is emphasized, and the simplicity of the proposed heuristic may prove beneficial. Another advantage resides in its integration with the algorithm already implemented in DMD.

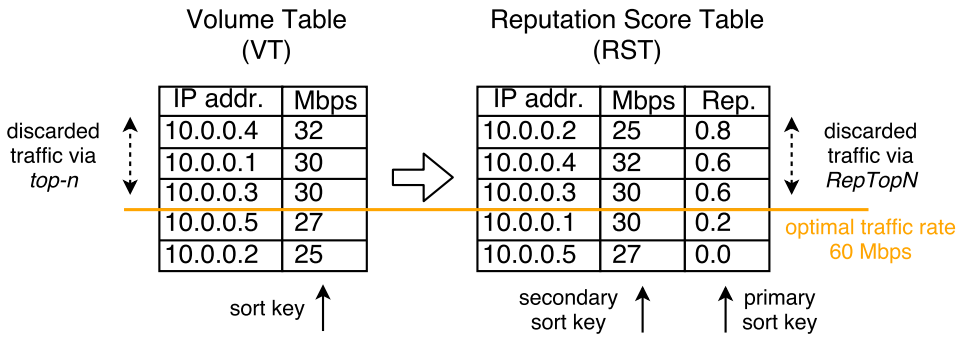

Figure 2.2: An example of a difference in discarded traffic depending on the used heuristic.

Figure 2.2 shows the difference in discarded traffic depending on the used heuristic. The current implementation of DMD constructs a Volume Table (VT) by sorting the traffic sample table via the traffic volume contribution column. *RepTopN* introduces a second stage in which the list of addresses from the VT is queried in NERD and the resulting reputation scores are joined with the information from the VT. The reputation score of every IP address unknown to NERD is automatically set to 0. The new table containing both the traffic volume contributions and the reputation scores is called Reputation Score Table (RST). RST is always sorted according to the reputation score as the primary key and, in case multiple addresses have the same score, according to the traffic volume as the secondary key. After that, the algorithm selects as many IP addresses from the top of the table for blocking as is needed to get the total volume of the remaining addresses below the optimal level.

2.2.2 Design Overview

One of the advantages of *RepTopN* is that it requires minimal changes to the existing mitigation system design. Figure 2.3 describes the augmented

mitigation system. The flowchart is simplified for a single mitigation rule, and its highlighted parts represent the proposed design changes.

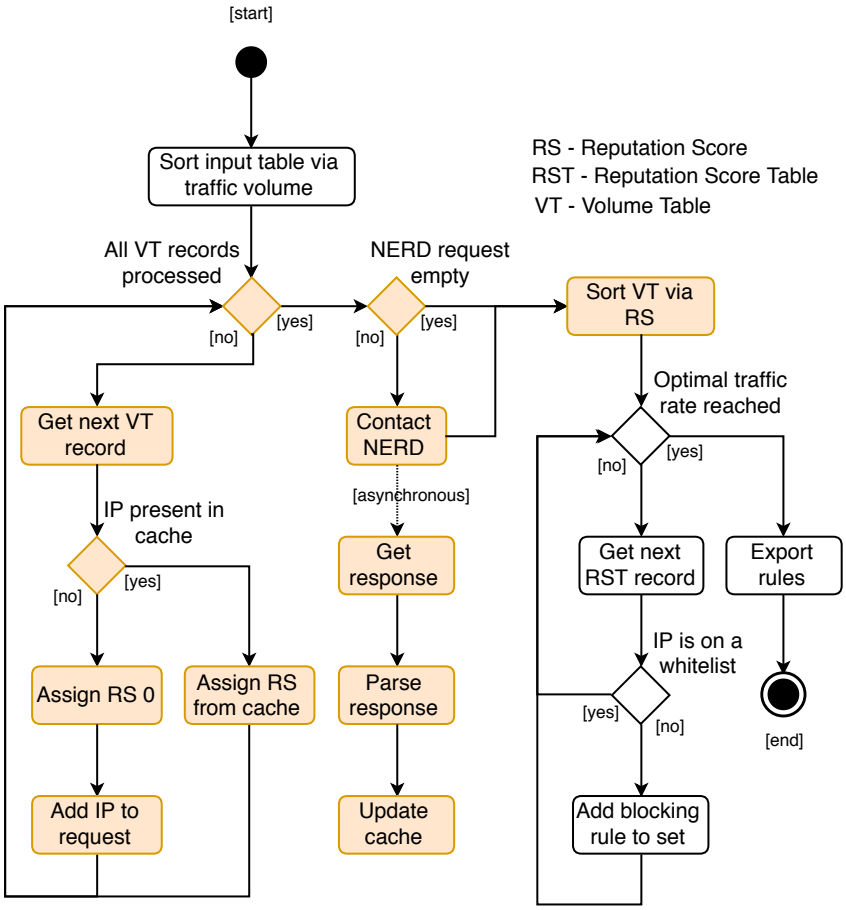

Figure 2.3: Flowchart of the augmented mitigation simplified for a single mitigation rule.

The system begins with sorting of an input table which represents live traffic sample in descending order according to the traffic volume contribution each IP address generates and thus creating a Volume Table (VT). Since the number of IP addresses in VT may be considerable during an attack, fetching reputation scores for all of them may take a significant amount of time. Therefore, the process of acquiring reputation scores from NERD is performed independently of the mitigation cycle, and a local cache is used to acquire a reputation score for every IP address present in VT. If an IP address is not found in the cache (e.g. when a new source contributes to the attack), its reputation score is set to 0 during the current cycle, and the IP address is added to a query which is later sent to NERD. After obtaining the last reputation score, the assembled query is sent to NERD API, and VT is

sorted in descending order according to the reputation score column utilizing a stable sort resulting in the creation of a Reputation Score Table (RST). IP addresses occurring in the RST are subsequently added to the blocking set (unless they are present on a whitelist) until the optimal traffic volume rate is reached.

When the reputation scores are received after an inevitable time delay (usually a few seconds), they are stored in the cache and used to assemble RST in the next cycle. Naturally, the cache may be empty at the very beginning of an attack. In such case, all reputation scores are set to 0 which leads to the fact that only the traffic volume contribution is used to sort the table and select addresses for blocking as in the standard *top-n* method. The advantage of *RepTopN* begins to apply in the mitigation cycle following the reputation cache update.

Based on the design of *RepTopN* it is apparent that the efficiency of the *RepTopN* algorithm is directly proportional to the quality of the reputation database and the number of IP addresses found in the reputation database. If either of the following scenarios occurs:

- query result from NERD has not been processed yet,
- none of the queried IP addresses was found in NERD,
- a communication issue between DMD and NERD occurs,

the *RepTopN* heuristic acts like a simple *top-n*, thereby satisfying one of the design requirements – to always discard traffic deterministically. Otherwise, in the usual state *RepTopN* creates the set of blocking rules considering the information about entities reputation.

2.3 Data Model Overview

Figure 2.4 summarizes an updated logical data model of the amplification module in DMD. The parts highlighted with orange color represent added data structures. Similarly, the red parts represent deprecated structures. Some aspects of the data model essential for the augmented mitigation system design are elaborated below.

- `Amplification Module`:
  - a primary data structure which holds all the configuration and other data structures
- `Amplification Rule`:
  - represents one of the amplification rules for attack mitigation described in Sec. 1.2.2.

- `Augmented Mitigation`:
  - a data structure for storing all of the useful information for performing the augmented mitigation (e.g. reputation score cache)
  - contains necessary data for communication with NERD
- `Reputation Score Table`:
  - a data structure representing a table with fixed maximum size
  - contains records used for the *RepTopN* algorithm
  - replaces `Volume Table` data structure
- `RST Record`:
  - represents a record in Reputation Score Table
  - contains the IP address (16 bytes to support IPv6), the traffic volume contribution in bytes or packets, and the reputation score
  - replaces `Volume Table Record` data structure

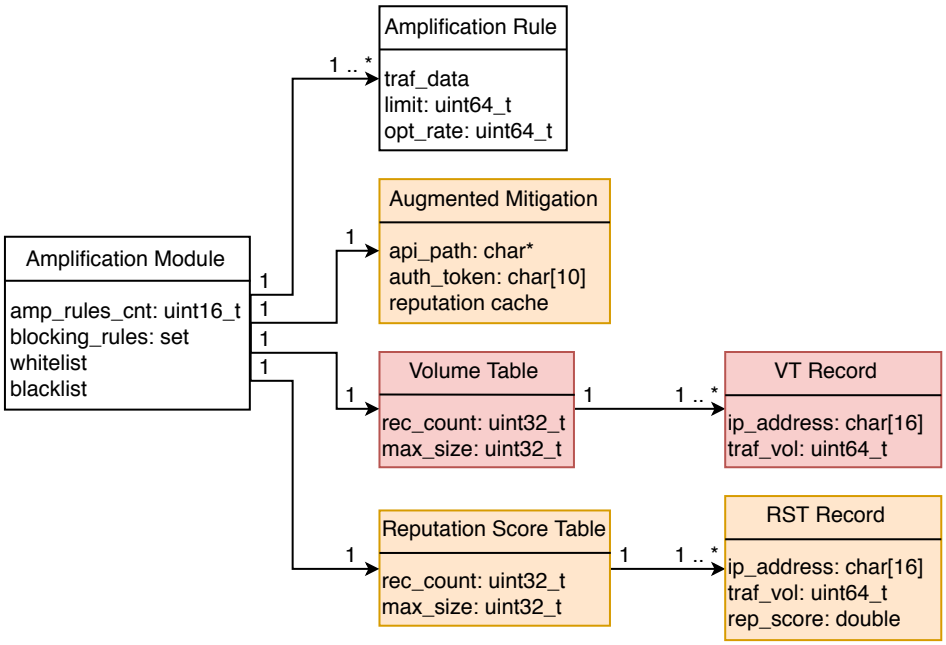

Figure 2.4: Simplified data model of the augmented mitigation system.

Chapter 3

Implementation

The contents of this chapter discusses specific implementation details of particular critical system components, namely the process of acquiring the reputation scores from NERD, the implementation of the reputation cache and the description of applied sorting algorithms. Each of the components mentioned above is elaborated and placed in the context of the rest of the DMD implementation. Finally, the processing time complexity of each component is discussed in this chapter since it is a crucial factor influencing most of the implementation decisions.

3.1 Acquisition of Reputation Scores

The NERD system provides a RESTful API to communicate with other systems. The RESTful API is based on representational state transfer (REST) technology. REST is an architectural style and approach to communications often used in web services development. REST-compliant web services allow the requesting systems to access and manipulate resources of the web service by using a predefined set of stateless operations. The most common communication protocol related to RESTful APIs is `HTTP` or its encrypted variant – `HTTPS`. By using predefined `HTTP` methods (e.g., `GET`, `POST`, `PUT`, `DELETE`), systems are able to issue requests to the URI of the resource, execute the desired operation and the REST-compliant web service sends a response in a predefined format (e.g., `XML`, `JSON`).

NERD defines few API endpoints used primarily for requesting data about the network entities in the database rather than for the data insertion or erasure. Unless stated otherwise in the API endpoint definition, every response is in the `JSON` format.

To ensure confidentiality of the data in the database NERD implements a token authorization mechanism and an access control list (ACL). Every API request must include an `HTTP` header with the authorization token. The token is a ten characters long string consisting of random alphanumeric symbols

(62 characters: $A-Z, a-z, 0-9$), and therefore the total of $62^{10} = 8.39 \times 10^{17}$ different permutations exist. Assuming an online brute-force attack with the rate of $1,000$ guesses per second, the search space would be exhausted in 2.66 hundred thousand centuries. Moreover, the server can utilize a countermeasure techniques such as *fail2ban* to prevent massive guessing attacks. NERD receives the authorization token in the request, authenticates the user and verifies his privilege to access the resource using ACL.

It is vital to ensure a proper manipulation and storage of the NERD authorization token in DMD. Therefore, the authorization token is kept (along with other NERD API information) in a configuration file separately from the program itself. The configuration file with the authorization token has the least possible access permissions, and the path to the configuration file is passed as a parameter during the DMD startup.

The communication with NERD via `HTTPS` with the aim to obtain a large number of reputation scores is in DMD realized by using the *libcurl* library[3]. The semantics and the details of the bulk API queries are further elaborated in the following subsection.

3.1.1 Bulk Queries API

The design of the augmented mitigation system described in Sec. 2.2.2 anticipates the possibility of sending bulk queries to the NERD API to acquire the reputation scores of an immense number of IP addresses. Sending a request with multiple IP addresses is undoubtedly more efficient than requesting reputation scores one by one. Unfortunately, NERD API currently does not support bulk queries, and therefore a new endpoint needs to be implemented.

An API endpoint `/nerd/api/v1/ip/bulk/` is designated for the bulk reputation score querying. The IP addresses in DMD are stored in 16 bytes long arrays, however, they are represented as strings in NERD. Which means that the necessary conversion from bytes to string needs to be performed in one of the systems. To broaden the possible usage of the bulk query endpoint (not limited only for the connection with DMD) and to save the precious computational time in DMD, the API endpoint for bulk queries supports two possible input and output data formats – **plain text** and **binary**.

Figure 3.1 depicts an example of potential usage of NERD API for bulk queries. The figure shows the basic differences between the two possible use-case scenarios depending on the chosen data format. The API endpoint is implemented as follows:

1. Receive an incoming `HTTP` request.

2. Perform authentication and authorization of the request based on the authorization token present in the `HTTP` header.

[3] https://curl.haxx.se/libcurl/

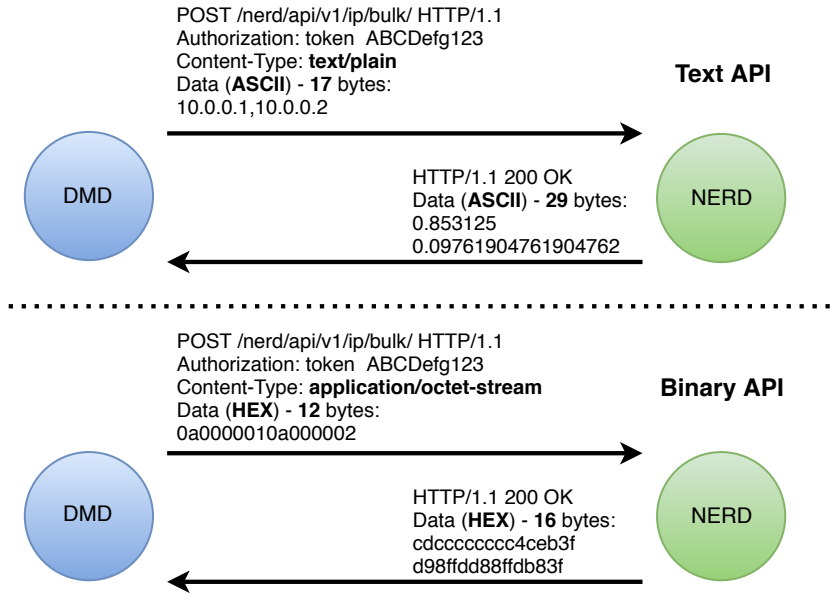

Figure 3.1: Example of NERD bulk queries API usage.

3. The format of the input data is deducted from the `HTTP` header field `Content-Type`. Expected values are `application/octet-stream` or `text/plain`.

4. Individual IP addresses are decoded from the raw payload data. If the format is `text/plain`, the raw data are decoded using `ASCII` encoding, and a comma separates individual IP addresses. The second option is the `application/octet-stream` format. In that case, NERD assumes that every 4 bytes (using network byte order) represent an IPv4 address (NERD does not currently support IPv6 entities). Therefore, no entity separator is necessary, and NERD performs conversion of each IP address from the binary format to the textual format.

5. NERD creates a dictionary of reputation scores using the IP addresses in the request as keys. Every IP address is assigned with the reputation score 0.0 by default in case the requested IP address is not present in the database.

6. A single bulk *Mongo database* query is formed and executed.

7. The query result is processed, and the reputation scores in the dictionary are updated accordingly.

8. A response is created in the same data format as the request. The response contains only a list of reputations scores for each IP address

queried in the same order as the IP addresses were passed to API. If the data format is `text/plain`, the reputation scores (in the form of a string) are separated by a newline. If the `application/octet-stream` data format is used, every 8 bytes (using network byte order) represent a *C* double precision data type, and no separator is necessary.

Using the textual data format is slightly more time efficient because it is not necessary to convert the requested IP addresses to their textual representations. However, the size of the request is significantly larger when using the textual variant of the API endpoint. The textual representation of an IPv4 address might consist of up to 16 bytes (including the separator) while the binary representation will always be only 4 bytes long. The maximum number of IP addresses in a single bulk query from the perspective of DMD is capped to 2^{17} entries (the maximum number of records in RST). In such case, the size of the request in the textual format would be around 2 MB while the size of the request in the binary format would be four times lower – about 512 KB.

3.1.2 Communication with NERD

The communication channel for requesting and obtaining reputation scores from NERD via `HTTPS` is implemented using the *libcurl* library. It is a free client-side URL transfer library supporting many different internet protocols (e.g., `HTTP`, `FTP`, `IMAP`). The main benefits of using the *libcurl* library are that it provides a simple interface and is thread-safe. The usage of the *libcurl* library in the augmented mitigation system is straightforward.

Since the *libcurl* library is thread-safe, multiple requests to NERD can be issued at the same time. With respect to the design described in Sec. 2.2.2, it is convenient to establish and maintain a session for every protected network (amplification rule) specified in DMD. To ensure that multiple connections with NERD can be maintained simultaneously, each of the mitigation rules must keep its session handle and data structures (such as buffers for sending and receiving data). The *curl* session handles are created using `curl_easy_init()` function during an initial phase of the amplification module along with other necessary data structures. Most of the `HTTP` request is formed in the initiation phase as well: specifying target URL, appending header field with the authorization token and defining the data format in the request using the `Content-Type` header field. The *libcurl* library also requires the programmer to specify `CURLOPT_WRITEFUNCTION` and `CURLOPT_WRITEDATA` variables. Otherwise, the response received from NERD would be written to the standard output by the *libcurl* library. A pointer to a write callback function is passed to the *libcurl* library via `CURLOPT_WRITEFUNCTION`. The `CURLOPT_WRITEDATA` option is closely related to the callback function since its purpose is to pass a pointer to custom data to one of the callback function arguments. When processing the live data sample of the currently served am-

plification rule, the NERD bulk query is formed containing all IP addresses which are present in the Volume Table but not in the reputation cache. If the query is not empty, a separate thread is deployed to merge the query with the rest of the prepared `HTTP` request and sends it to the designated NERD API endpoint using `curl_easy_perform()` function. The thread detached from the mitigation cycle is in a blocking state until the first data from NERD are received. The specified write callback function is repeatedly called, continuously storing parts of the response of various size to the prepared buffer. When the whole response from NERD is processed, the `HTTP` response code is checked for the value of `200 OK`. Subsequently, both the request buffer and the response buffer are parsed simultaneously, individual IP addresses are matched to the relevant reputation scores, and the pairs are stored to the reputation cache. The detached thread terminates at this point without a return value. The whole process is summarized in Fig. 3.2.

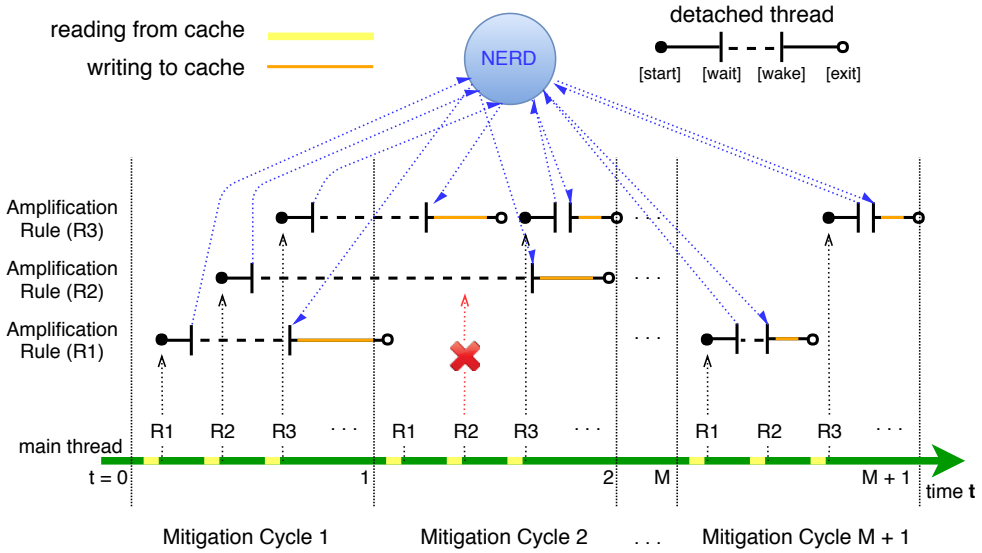

Figure 3.2: Thread-safe augmented mitigation system for 3 different amplification rules.

As proposed, multiple detached threads can simultaneously handle the `HTTP` session as long as each thread is associated with a different amplification rule (a different DDoS attack). A problem arises when multiple threads attempt to use the same resources, thus resulting in a possible data race. The problem can be illustrated with the following example.

Let us consider a large-scale DDoS attack. The reputation cache is empty in the first mitigation cycle; therefore, a large bulk query is formed. A dedicated thread is launched to acquire the reputation scores using the resources (buffers and the *curl* session handle) of the currently serviced amplification rule. Since the number of attackers contributing to the DDoS attack is im-

mense, the response from NERD might not be acquired before the next mitigation cycle. This delay results in the creation of a similar (attack characteristics might vary slightly – e.g. new attackers) bulk query because the reputation cache still does not contain any information about IP addresses present in VT. At this point a second thread would be launched using the same resources as the previously launched thread, thus resulting in undefined behavior. The implemented solution to this problem is to set a flag for every amplification rule indicating whether a detached thread is already attempting to acquire reputation scores from NERD. The main thread drops the flag right before the dedicated thread is launched and the detached thread raises the flag again once it updates the reputation score. In the next mitigation cycle, the main thread does not launch another dedicated thread if the flag is still down.

Some measurements were performed during the implementation stage regarding the possible delay between sending a request to NERD and receiving the reputations scores. Though it would seem that the delay scales in correspondence to the number of IP addresses in the bulk query, the main correlation resides in the quantity of queried IP addresses present in NERD. Such behavior is caused by the fact that in the NERD's *Mongo database* the IP addresses are indexed. Therefore, searching whether an IP address is stored in the database is quite fast. However, if the IP address is present in the database, reading its reputation score is noticeably slower. If DMD sends a query consisting of 2^{17} IP addresses (the default size of VT) which are not present in NERD, the response arrives in less than a second. On the other hand, if all the 2^{17} IP addresses are present in NERD, it can take up to 12 seconds to receive a reply.

It is worth mentioning that such case is theoretical and most of the queries would be processed in few seconds in practice. Further tests with DMD used in a real environment are required to measure the average time delay between the request and the response. If it proves to be problematic, one of the possible solutions to this problem is to limit the number of IP address in the bulk query and the reputation scores would be acquired using multiple requests. The thesis does not further elaborate this problem since it is more related to the reputation database itself than to the augmented mitigation and recommends this issue as a topic of future research and development.

3.2 Reputation Score Cache

The reputation score cache proves to be an essential component of the whole augmented mitigation system design. Every thread handling the acquisition of the reputation scores from NERD accesses the cache to insert the newly acquired reputation scores using the 128-bit key representing the corresponding IP address. The main thread can perform an immense number of cache lookups in each mitigation cycle. Based on these observations, the data struc-

ture implementing the reputation score cache must be thread-safe, and both `insert` and `get` operations must have a constant asymptotic time complexity – $\mathcal{O}(1)$.

An implementation of a hash table with a stash called *Fast Hash Table*[4] applied in many CESNET projects meets both requirements. The *Fast Hash Table* utilizes *MurmurHash3*[5] as a hash function for fast table lookups.

3.2.1 Fast Hash Table

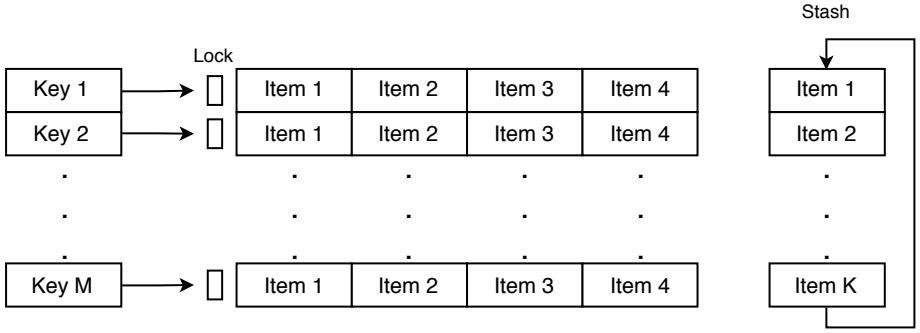

Figure 3.3: Scheme of the *Fast Hash Table* implementation.

Figure 3.3 depicts the implementation scheme of the *Fast Hash Table*. The number of rows of the table and the size of the stash are set during the initialization. Each indexable row can contain up to 4 different items with the same hash value of their respective key. Each row of the table has a hidden vector of flags indicating which item in the row is the least recently used. The vector of flags is updated every time one of the items in the row is read or written. If all columns in the row are occupied, a newly inserted item replaces the least recently used item which is subsequently placed in the stash. The stash is implemented as a circular buffer, therefore, if the stash is already full, the replaced item is written to the first place of the stash. *Fast Hash Table* also provides functions which can bypass the stash entirely; therefore, any data replaced in the table with these functions would be definitely lost.

Thread safety of the *Fast Hash Table* resides in locking the individual rows. Reading or writing an item in the row ultimately locks the whole row for the duration of the operation. However, other threads may freely update other rows. This approach is significantly better than locking the whole table each time a thread is about to use it, which would have a dire effect on the performance time of the main thread.

MurmurHash3 is the latest version in the series of *MurmurHash* functions published on Github by the author *aappleby*. It is a general-purpose hash

[4]https://github.com/CESNET/Nemea-Framework/tree/master/common/
[5]https://github.com/aappleby/smhasher/wiki/MurmurHash3

function for producing 32-bit and 128-bit hash values. It is highly efficient in comparison with some other commonly used hash functions and optimized for keys which size is divisible by 8 (IP address data structure in DMD has a size of 16 bytes).

3.3 Sorting Algorithms

The need for the use of one or more sorting algorithms is evident since the whole *RepTopN* heuristic is based on **multiple-key sorting**. A sorting algorithm is initially applied to the live traffic data sample to create a Volume Table – a table sorted by the traffic volume contribution of each IP address. The traffic volume contributions data have no unique characteristics which would allow to utilize a sorting algorithm which is not based on the compare and exchange principle. Therefore, the standard *Quicksort* implementation provided by the *glibc* library is used for the VT creation. The *Quicksort* algorithm takes on average $\mathcal{O}(n \log n)$ comparisons to order n items; however, it might require $\mathcal{O}(n^2)$ comparisons in rare cases.

The second sorting is performed once the data in VT are extended with the related reputation scores. The value of a reputation score is a real number in a range $\langle 0; 1 \rangle$. That theoretically means there are infinite possible values of the reputation score. However, if we round the reputation score to a fixed number decimal places (let us assume two), then each reputation score can obtain only one of 101 possible values. The slight reduction of the reputation score precision has barely any noticeable effect on the *RepTopN* heuristic. The advantage of this adjustment is the possibility of utilizing *Counting Sort* for the creation of Reputation Score Table.

Counting Sort is a stable (maintains the relative order of items with equal values) out-place (requires additional memory) sorting algorithm introduced by *Harold Seward* in 1954 in [28]. It does not rely on the compare and exchange principle but employs the knowledge of the minimum and the maximum values of sorted items. The algorithm is demonstrated in Fig. 3.4 and can be divided into the following steps (assuming n items and k possible item values):

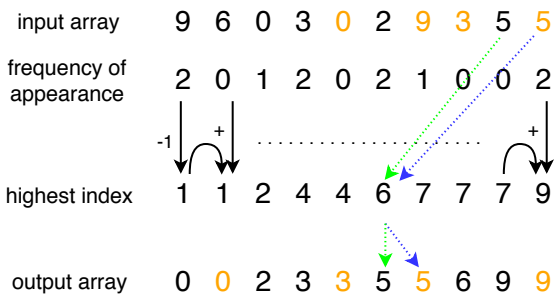

Figure 3.4: Demonstration of an array sorting using *Counting Sort*.

1. A histogram of the number of times each item value occurs within the input array is assembled. – $\mathcal{O}(n)$
2. The prefix sum computation is performed over the histogram to determine, for each item value, the highest position index in the output array of the items having that value. – $\mathcal{O}(k)$
3. The input array is iterated backward to achieve stability of the sorting algorithm. Every item is used as an index in the count array, thus obtaining index in the output array where the item should be copied. It is important to decrement the value in the count array so that the next item would not overwrite the previously inserted item with the same value. – $\mathcal{O}(n)$

Counting Sort is suitable for sorting arrays where items can obtain a known finite range of values. Furthermore, it maintains the relative order of items with equal values (is so-called stable) which is a necessity for performing the desired **multiple-key sorting**. The most apparent advantage of *Counting Sort* is the time complexity. Assuming n items and k possible item values, the overall time complexity of *Counting Sort* is $\mathcal{O}(n + k)$. That is superior to every sorting algorithm based on the compare and exchange principle, which asymptotic function cannot be better than $\mathcal{O}(n \log n)$. The disadvantages include an additional $\mathcal{O}(n + k)$ memory space, the inapplicability to data with unknown minimum and maximum possible values, and the necessity of having discrete data values (meaning it is, for instance, not applicable to real numbers). However, neither the additional memory space requirement nor the discrete value requirement proves to be a problem in DMD (since a limit was placed on the range of possible values of the reputation score). Therefore, its variation for sorting in descending order is applied to create RST in every mitigation cycle.

In general, if two algorithms have different asymptotic functions, it means that the time difference in their execution time scales with the number of items sorted. However, the highly optimized *Quicksort* implementation in *glibc* has probably a better performance for a smaller number of items. So the question arises whether the number of records in VT is high enough to notice the difference when using the proposed *Counting Sort* implementation. Several measurements have been executed to answer this question. The speedup becomes noticeable for $4,096$ (2^{12}) items. For the maximum possible VT size, which is implicitly $131,072$ (2^{17}) items, the *Counting Sort* implementation is already several times faster than the *Quicksort* implementation provided by the *glibc* library.

Chapter 4

Testing and Evaluation

The final chapter of this thesis presents a variety of different test and measurement results based on which the following subjects are evaluated:

- the impact of the number of IP address with a known reputation score on the *RepTopN* heuristic quality in Sec. 4.3.1,
- the impact of the amount of malicious traffic during an attack on the *RepTopN* heuristic quality in Sec. 4.3.2,
- the impact of the reputation database data reliability on the *RepTopN* heuristic quality in Sec. 4.3.3,
- the processing time of the proposed augmented mitigation system in Sec. 4.4,
- the memory requirements of DDoS Mitigation Device related to the changes in the DMD design in Sec. 4.4,
- the efficiency of the presented *RepTopN* heuristic implementation in comparison to the expected theoretical mitigation efficiency in Sec. 4.4.

Most of the subjects of evaluation mentioned above are assessed in comparison either to the previous implementation of DDoS Mitigation Device or the *top-n* algorithm. Such comparison is made to emphasize the benefits of the proposed mitigation system and heuristic enhancements. The performed tests and measurements cover all aspects of the *RepTopN* heuristic and the augmented mitigation system itself which are considered to be relevant by the author of the thesis. The comprehensive test results provide valuable feedback about the assets and the shortcomings of the proposed heuristic and the system.

4.1 Evaluation Metric

It is necessary to choose one or more suitable evaluation metrics to evaluate the performance of the *RepTopN* heuristic. Before introducing one such evaluation metric, the composition of data used in the evaluation tests needs to be described.

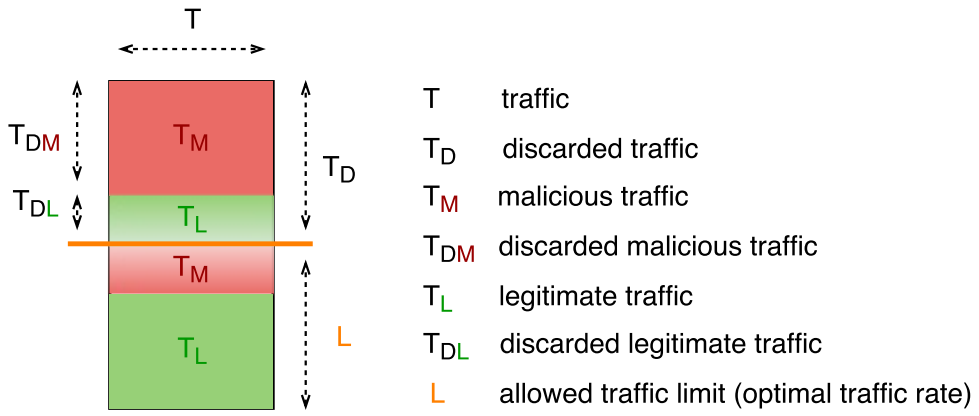

Figure 4.1: Traffic composition during a DDoS attack from the perspective of DDoS Mitigation Device.

Figure 4.1 shows the potential components of the traffic that ought to be cleaned by DDoS Mitigation Device during a DDoS amplification attack. Naturally, the overall traffic (T) consists of the legitimate (green) traffic (T_L) and the malicious (red) traffic (T_M). Since the heuristic algorithms have no prior knowledge of the sources of the malicious traffic, there is a probability that some legitimate packets are discarded (T_{DL}). On the other hand, it is also possible that some malicious packets are forwarded instead of discarded. Such scenario could happen either because the optimal traffic volume rate (L) is reached by discarding enough of other traffic or because the forwarded malicious packets are a residue of the mitigated attack. The amount of discarded traffic (T_D) tends to be slightly higher than it is necessarily required by the specified optimal traffic volume rate. The difference is a direct consequence of the usage of the traffic discarding based on IP addresses because more traffic can be discarded than necessary by discarding the overall traffic of n-th IP address. The following equations summarize the vital parts of the traffic composition:

- $T = T_M + T_L$
- $T_D = T_{DL} + T_{DM}$
- $T_D \geq T - L$

Since the primary objective of *RepTopN* is to mitigate DDoS amplification attacks with the minimal possible disruption of the legitimate communications, it is convenient to define the evaluation metric as the ratio of discarded legitimate traffic to overall discarded traffic – $\frac{T_{DL}}{T_D}$. However, there might be a case in which the data do not contain enough malicious traffic to satisfy the need to reach the optimal traffic volume rate; therefore, some legitimate traffic must also be filtered. The metric naturally takes this case into account, but it should be rare in practice due to a reasonable configuration by an expert who does not set up the optimal traffic volume rate below the legitimate traffic volume rate.

The following function includes the $T_D > T_M$ option:

$$f_1(T_M, T_{DL}, T_D) = \begin{cases} \frac{T_{DL}}{T_D} & T_D \leq T_M \\ \frac{T_{DL}-(T_D-T_M)}{T_D} & T_D > T_M \end{cases} \tag{4.1}$$

By combining both cases of the function f_1, the resulting function f as defined in (4.2), is the evaluation metric used for the evaluation of *RepTopN* and *top-n* in this chapter.

$$f(T_M, T_{DL}, T_D) = \frac{T_{DL} - max(T_D - T_M, 0)}{T_D} \tag{4.2}$$

It is evident from the definition of the function f that the range of the function f is $\langle 0, 1 \rangle$. The lower the value of f is, the lower the amount of the unnecessarily dropped legitimate traffic.

4.2 Test Data Presumptions

A complex data set with many different characteristics would be required to measure the influence of every possible traffic aspect. Therefore, several presumptions about the test data must be set out so that the individual traffic aspects reliably manifest during evaluation tests. As we are interested in discovering the performance of *RepTopN* compared to *top-n*, there are three potentially interesting scenarios:

Scenario 1: If each of the attacking IP addresses generates significantly more traffic than any of the legitimate IP addresses, *RepTopN* produces as good results as *top-n* regardless of the number of IP addresses with known reputation score (neglecting false positive reputation scores).

Scenario 2: If all the attacking IP addresses generate significantly less traffic than the legitimate sources, *top-n* fails to discard malicious packets. On the contrary, *RepTopN* performs well in this scenario in case the malicious sources have a non-zero reputation score.

Scenario 3: If all IP addresses generate approximately the same volume of traffic, *top-n* degenerates to a heuristic which discards random packets. An informed packet filtering (*RepTopN*) can help in this scenario.

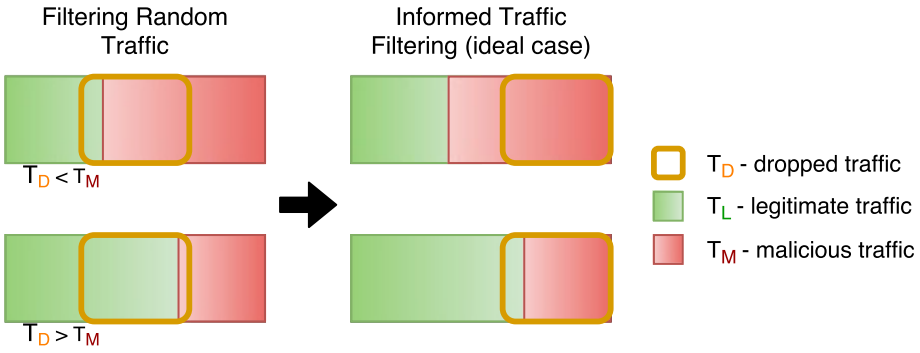

Figure 4.2: Expected outcome of the uninformed and fully informed *RepTopN* heuristic.

The scenarios 2 and 3 were the fundamental motivation of *RepTopN*. The non-trivial scenario 3 is depicted in Fig. 4.2. The figure describes the worst and the best expected possible outcomes of *RepTopN* depending on the ratio of malicious IP addresses with known reputation score to all malicious IP addresses. This scenario is ideal for measuring the efficiency of *RepTopN* compared to *top-n*.

The test data simulating the traffic during a DDoS amplification attack used in the experiments described throughout this chapter meet the following criteria:

- The traffic volume of each IP address is randomly chosen according to Gaussian statistical distribution to simulate the scenario 3.
- IP addresses with a non-zero reputation score generate only malicious traffic.
- IP addresses that generate legitimate traffic always have the reputation score equal to zero.
- IP addresses with zero reputation score may generate either only legitimate or only malicious traffic.

Note that the assumptions regarding the reputation scores are not applied when measuring the impact of the reputation database data reliability on the *RepTopN* heuristic quality in Sec. 4.3.3.

4.3 RepTopN Evaluation

This section presents the *RepTopN* evaluation results in comparison to the *top-n* results using the evaluation function f defined in Sec. 4.1 and the test data described in Sec. 4.2.

The test data simulate a DDoS amplification attack with the rate of approximately 10 Gbps produced by 1,000 individual IP addresses unless stated otherwise. The overall traffic volume rate is not a fixed constant since it depends on the parameters of the Gaussian distribution used to determine the traffic volume of each IP address, and therefore is the cumulative sum of the contribution of all (1,000) IP addresses. The traffic volume contribution of each IP address is randomly chosen according to Gaussian statistical distribution with the mean $\mu = 10$ Mbps and the standard deviation $\sigma = 1.5$ Mbps. The resulting traffic volume contribution of each IP address may vary from around 5 Mbps to 15 Mbps. The exact values of the traffic volume each IP address generates are different every time a test instance is created.

Each of the following subsections focuses on a different aspect of the simulated traffic to assess the impact of the aspect on the *RepTopN* heuristic.

4.3.1 Number of Identified Attackers

This section assesses the impact of the number of IP addresses with a known reputation score on the heuristic quality. One of the following test data parameters is modified for each test instance to achieve the stated goal:

- the number of IP addresses generating malicious traffic
- the portion of malicious IP addresses with a known (non-zero) reputation score

Each tested data configuration is launched multiple times, and the average result is used to reduce the measurement deviation.

Figure 4.3 summarizes the obtained results. As expected, the *RepTopN* algorithm performs better with a higher number of identified malicious IP addresses. Test instances in which no reputation score is known to *RepTopN* (the most rear curve) correspond to the performance of the *top-n* algorithm. The results confirm the anticipation that *top-n* behaves as a random IP address filter if the attacking IP addresses are not distinguishable at first sight based solely on the traffic rate analysis.

The mitigation efficiency of the presented *RepTopN* heuristic is superior to the *top-n* algorithm in 99 % of cases. In the worst case scenario, the *RepTopN* heuristic performs as bad as the *top-n* algorithm. The 3D graph peaks for the instance where 700 malicious IP addresses are present in the traffic while none of them is in the reputation database. Such result is closely related to the optimal traffic volume rate $L = 3$ Gbps because in this particular instance the

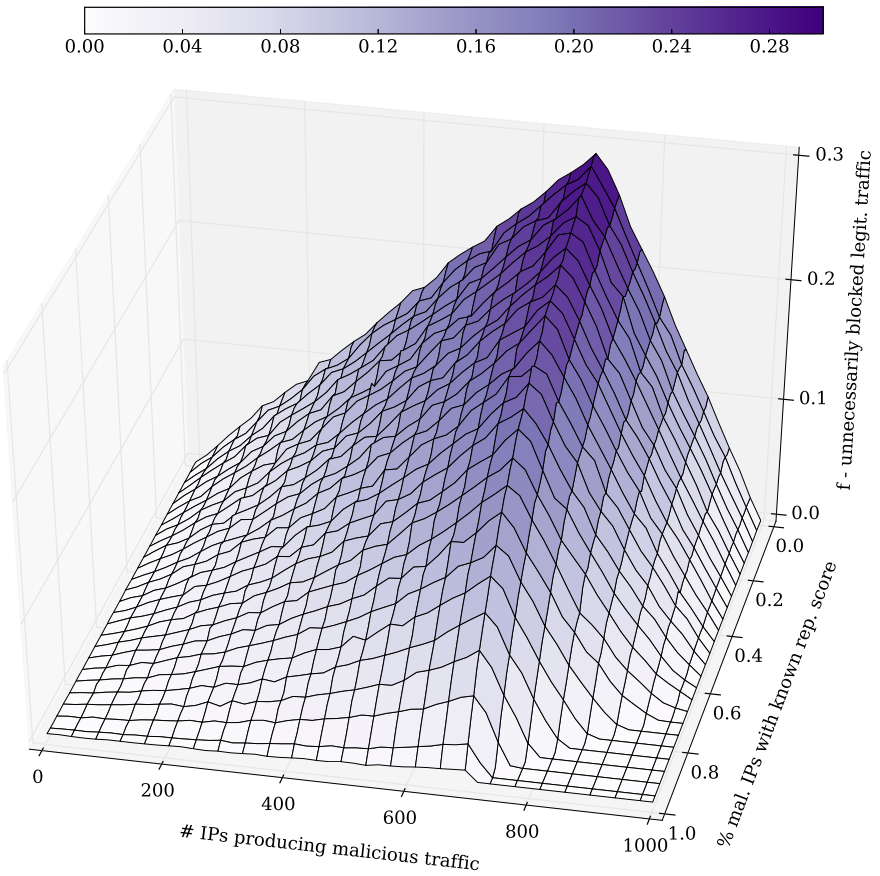

Figure 4.3: 3D graph representing the dependence of the *RepTopN* efficiency on the number of identified attackers for a fixed value of the optimal traffic volume rate $L = 3\,\text{Gbps}$.

amount of the legitimate traffic volume is similar to the optimal traffic volume rate ($T_L \sim L$). Instances where $T_L \sim L$ represent scenarios in which all the malicious traffic can be discarded without any residual malicious traffic. When using the presented evaluation function f, all of the malicious packets would have to be identified so that no discarded legitimate traffic could be replaced by previously not discarded malicious traffic for the heuristic to reach the best possible score. Both *RepTopN* and *top-n* perform the worst in such scenarios.

It is apparent that by using the *top-n* algorithm in the scenario depicted in Fig. 4.3 where $L = 3\,\text{Gbps}$ (which is 30 % of the overall traffic), up to 30 % of the discarded traffic may consist of the legitimate traffic. However, based on the previously mentioned observations, there might be some scenarios in which most of the traffic discarded using the *top-n* algorithm consists of the legitimate traffic when, for instance, the algorithm is required to mitigate 80 % of the overall traffic. In such scenarios, the *RepTopN* heuristic can significantly

reduce the amount of the unnecessarily discarded legitimate traffic with the efficiency related to the number of identified attackers thanks to the reputation score.

4.3.2 Ratio of Legitimate and Malicious Traffic

The previous tests results show that *RepTopN* is the most valuable in situations where the legitimate traffic volume rate is similar to the optimal traffic volume rate ($T_L \sim L$). Such situations are highly likely to appear in practice.

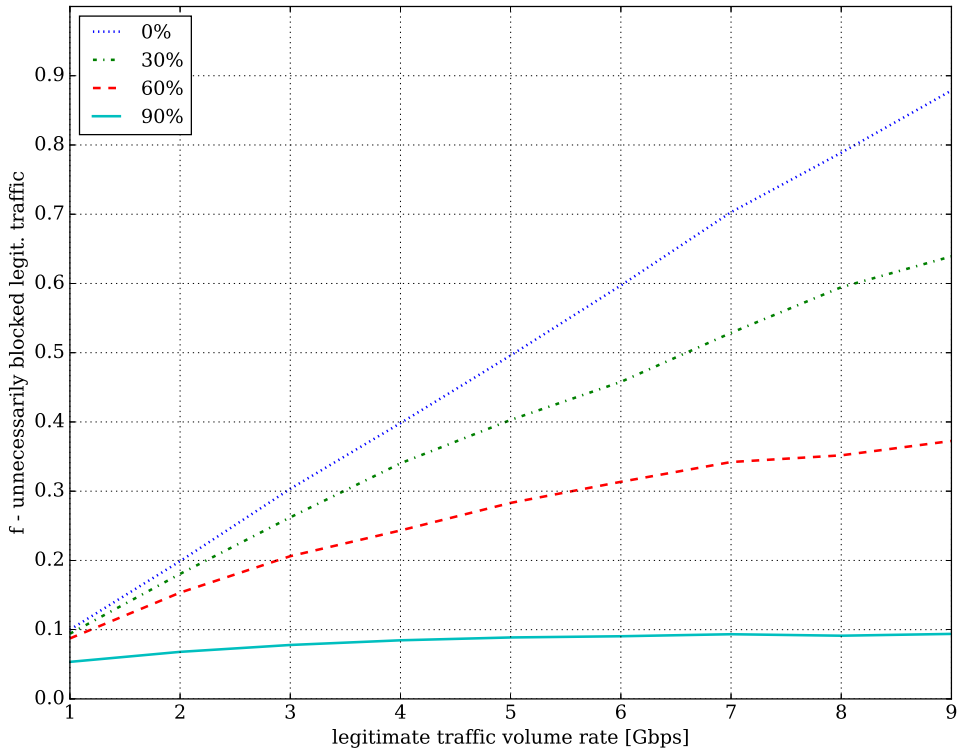

Figure 4.4: The dependence of the *RepTopN* efficiency on the amount of legitimate traffic and the sampled portion of identified attackers. Assuming $L \sim T_L$, $T \sim 10\,\text{Gbps}$.

Figure 4.4 further examines the *RepTopN* benefits depending on the ratio of legitimate and malicious traffic. Each line in the graph represents how well the *RepTopN* heuristic is informed – the portion of attackers *RepTopN* managed to identify. The line labeled as "0 %" represents the performance of the *top-n* algorithm. As we can see, the *RepTopN* heuristic does not perform significantly better than *top-n* for larger attacks since *top-n* would also block most of the identified malicious IP addresses. However, the graph shows that the *top-n* algorithm is the less effective the less amount of traffic is needed to

Table 4.1: Sampled results of tests focused on the ratio of legitimate and malicious traffic. Assuming $L \sim T_L$, $T \sim 10\,\text{Gbps}$.

$T_L : T_M$	Ident. attackers	f	T_{DL} [Gbps]	T_{DM} [Gbps]
2 : 8	0 % (0 IPs)	0.2	1.6 (80 % T_L)	6.4 (80 % T_M)
2 : 8	30 % (240 IPs)	0.18	1.44 (72 % T_L)	6.56 (82 % T_M)
2 : 8	60 % (480 IPs)	0.16	1.28 (64 % T_L)	6.72 (84 % T_M)
2 : 8	90 % (720 IPs)	0.07	0.56 (28 % T_L)	7.44 (93 % T_M)
5 : 5	0 % (0 IPs)	0.5	2.5 (50 % T_L)	2.5 (50 % T_M)
5 : 5	30 % (150 IPs)	0.41	2.05 (41 % T_L)	2.95 (59 % T_M)
5 : 5	60 % (300 IPs)	0.28	1.4 (28 % T_L)	3.6 (72 % T_M)
5 : 5	90 % (450 IPs)	0.09	0.45 (9 % T_L)	4.55 (91 % T_M)
8 : 2	0 % (0 IPs)	0.8	1.6 (20 % T_L)	0.4 (20 % T_M)
8 : 2	30 % (60 IPs)	0.6	1.2 (15 % T_L)	0.8 (40 % T_M)
8 : 2	60 % (120 IPs)	0.35	0.7 (8.75 % T_L)	1.3 (65 % T_M)
8 : 2	90 % (180 IPs)	0.1	0.2 (2.5 % T_L)	1.8 (90 % T_M)

discard. Therefore, the advantages of *RepTopN* become noticeable for weaker DDoS attacks (e.g., 1:1 ratio between legitimate and malicious traffic).

Table 4.1 summarizes some findings from Fig. 4.4 and puts them into a broader context. The table is focused on three different scenarios (based on the ratio of legitimate and malicious traffic) which are examined in more detail. The second column represents the number of attackers the heuristic would have to identify to achieve the presented results. The third column contains *RepTopN* evaluation score which can be simplified to $\frac{T_{DL}}{T_D}$ in these scenarios since there is never the need to discard legitimate traffic. The last two columns of Tab. 4.1 show the absolute value of discarded legitimate respectively malicious traffic and also their relative value with respect to the total legitimate respectively the total malicious traffic.

A disturbing observation can be made by looking at the first row of the table. Results in the first row are expected during large-scale DDoS amplification attacks with a massive number of reflectors. The data suggest that even though the *top-n* algorithm manages to mitigate most of the attack, it discards most of the legitimate traffic as well. Using the *top-n* algorithm in such case may have fatal consequences, and the ongoing DDoS attack can be considered successful. *RepTopN* achieves slightly better score depending on the number of known attackers. However *RepTopN* would require identifying around 75 % of the attackers to preserve at least 50 % of the legitimate connections, which may prove difficult for the DDoS amplification attacks with a massive number of reflectors. For milder DDoS attacks (assuming $T_M \leq T_L$) the *RepTopN* heuristic manages to preserve a reasonable amount of legitimate traffic for the necessary number of attackers it needs to identify.

It is worth noting that the rows with 90 % of identified attackers work in theory. However, such situations are hardly ever achievable in practice especially for large-scale DDoS amplification attacks (regarding the number of reflectors). It would require a thorough and extensive reputation database with highly accurate reputation scores, or the attacker would have to use a similar set of reflectors across the attacks.

An additional observation related to the presented results can be stated. The efficiency of the *RepTopN* heuristic does not scale linearly but polynomially with the number of successfully utilized reputation scores. Therefore, *RepTopN* can become a potent tool to mitigate DDoS amplification attacks if high-quality data are guaranteed in the reputation database.

4.3.3 Reputation Score Reliability

The quality of the *RepTopN* heuristic highly depends on the number of attackers identified thanks to the reputation score. However, further experiments are necessary to discover the relation between *RepTopN* efficiency and the quality of the reputation scores. As previously stated – for the purpose of the augmented mitigation system – it is possible to perceive a reputation score from NERD as a probability that the observed traffic of an IP address is malicious. However, the actual probability that a particular packet stream is malicious could vary if the data are incomplete in the reputation database.

A sophisticated test dedicated solely to this purpose has been launched and the results are shown in Fig. 4.5. The graph contains multiple lines. Each line represents the dependence of the *RepTopN* efficiency on the possible deviation of reputation score from the actual probability the IP address generates malicious traffic for a certain amount of IP addresses acquired from the reputation database (total of $1,000$ IP addresses in every test instance). Every tick on the x-axis represents a possible deviation of the reputation score from the actual probability which means, for instance, that if an IP address reputation score states 0.2 and deviation is 0.1, the actual probability that the traffic is malicious lies within the range between 0.1 and 0.3. The actual probability is used to determine whether the IP address belongs among the attackers or not. A fixed value of 10 % is used to decide whether an IP address with an unknown reputation score produces malicious traffic. Defining attackers in this manner causes that the amount of malicious traffic differs in every test instance. Therefore, even the optimal traffic volume rate had to be adjusted in every instance to ensure that $L \sim T_L$. It is apparent from the above test description that much uncertainty is present in the test results in an attempt to simulate a behavior which is likely to appear in practice. Each test instance is launched $1,000$ times to reduce the data fluctuation, and the average results are used in Fig. 4.5.

Apparently, the *RepTopN* heuristic performance results are relatively similar to the ideal scenario for the deviation value less or equal to 0.2. Never-

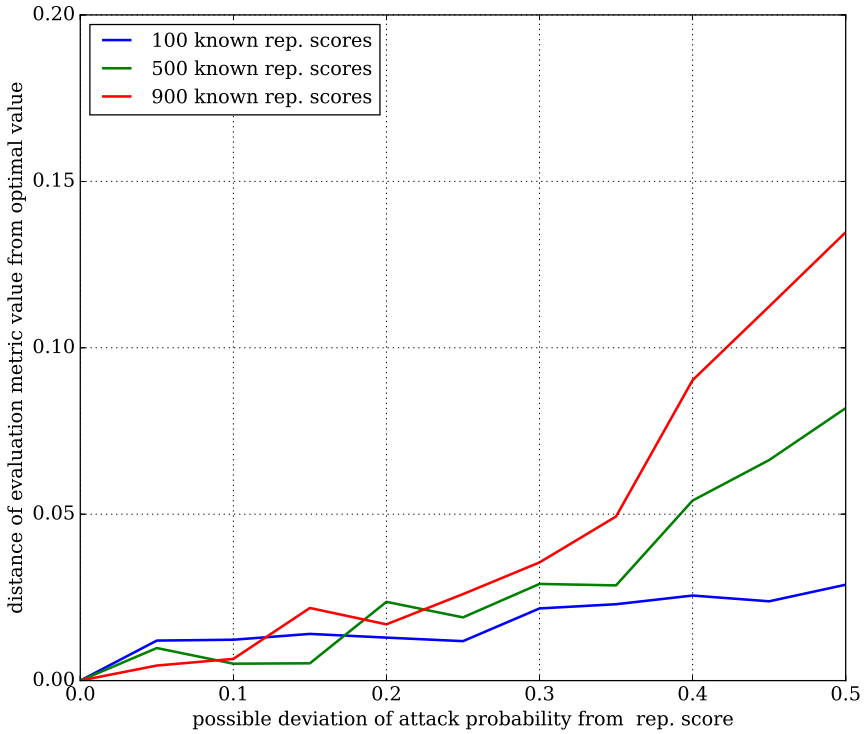

Figure 4.5: The dependence of *RepTopN* efficiency on reputation score reliability. Assuming $L \sim T_L$, $T \sim 10\,\text{Gbps}$.

theless, a degradation of the heuristic can be observed for deviation values higher than 0.25, and the higher number of reputation scores are acquired in the test, the more affected the *RepTopN* becomes.

To summarize the observations *RepTopN* is relatively resistant to unreliable reputation scores if they differ up to 20 % which should be a reasonable limit for every reputation database.

4.4 Implementation Performance

All performed experiments in this section utilize Spirent TestCenter[6] as a source of traffic which is sent directly to DMD. System information about the server operating DMD and used compiler are listed in Tab. 4.2.

According to the numerous tests, the implementation of the *RepTopN* heuristic reaches the expected theoretical results presented in Sec. 4.3.1 once the reputation cache is filled with acquired reputation scores. Slightly suboptimal results are observed for tests with a large number of attackers when more than four IP addresses have the same hash value. This situation results

[6]https://www.spirent.com/Products/TestCenter

Table 4.2: System information about server with deployed DMD.

Operating System	Scientific Linux 7.4
Linux Version	3.10.0-514.26.2.el7.x86_64
Procesor(s)	2× Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00GHz
CPU(s)	24
Operation Memory	RAM 64 GiB
Compiler	gcc v. 4.8.5
Optimizations	-O3

in rewriting reputation scores of certain IP addresses in the reputation cache and subsequently suboptimal mitigation results.

The processing time of *RepTopN* and also of the whole augmented mitigation system is strongly emphasized when dealing with DDoS attacks of magnitudes up to 100 Gbps. Assuming n is the number of IP addresses composing the overall traffic during a DDoS attack, the asymptotic time complexity of the previous *top-n* heuristic can be divided into the following individual components:

- creation of Volume Table using *QuickSort* – $\mathcal{O}(n \log n)$ in an average case, rarely $\mathcal{O}(n^2)$
- selection of m ($m \leq n$) IP addresses which traffic is to be blocked by inserting them into a list – $m \times \mathcal{O}(1) = \mathcal{O}(m)$

Since $m \leq n$, then $\mathcal{O}(m) \leq \mathcal{O}(n)$ and therefore the overall asymptotic time complexity of the *top-n* heuristic is the asymptotic time complexity of the *QuickSort* algorithm. A similar break down can be done for *RepTopN*:

- creation of Volume Table using *QuickSort* – $\mathcal{O}(n \log n)$ in an average case, rarely $\mathcal{O}(n^2)$
- lookup to the reputation cache for every IP address – $n \times \mathcal{O}(1)$
- creation of a request to NERD regarding l ($l \leq n$) IP addresses which reputation score is not present in the reputation cache – $\mathcal{O}(l)$
- creation of Reputation Score Table using *Counting Sort* – $\mathcal{O}(n + k)$ (already described in Sec. 3.3), assuming that k is the number of possible values of the reputation score
- selection of m ($m \leq n$) IP addresses which traffic is to be blocked by inserting them into a list – $m \times \mathcal{O}(1) = \mathcal{O}(m)$

Since $l \leq n$, $m \leq n$ and $k = 101$ in the implementation which is significantly lesser than the expected number of IP addresses present in the traffic n, the

asymptotic time complexity of the *RepTopN* heuristic is

$$\mathcal{O}(RepTopN) = \mathcal{O}(n^2) + 4 \times \mathcal{O}(n) = \mathcal{O}(n^2)$$

in the rare worst case and $\mathcal{O}(n \log n)$ in an average case.

It is evident that *top-n* and *RepTopN* have the same asymptotic time complexity which is caused by the *QuickSort* algorithm in both cases. Nevertheless, the previous system with *top-n* is expected to be slightly faster due to the negligences made during the computation of the asymptotic time. A stress test of both versions of DMD composed of 130,000 attackers with the cumulative traffic rate 100 Gbps proved this expectation. In this extreme scenario, the previous version of DMD is on average faster by 30 – 50 ms. However, some mitigation cycles were slightly slower – around 90 ms. This behavior is probably caused by the frequent usage of the reputation cache by both threads due to the sheer number of IP addresses present in the test. Such deviations from the average values were not observed in a smaller test with 30,000 IP addresses.

Memory consumption of the augmented mitigation system does not increase with the number of IP addresses present in live traffic. That is because all necessary data structures are created during the initialization phase with a fixed size which is either implicitly defined or specified in a configuration file. This solution is reasonable since no additional dynamic memory allocation, which costs precious computational time, is needed for the rest of the DMD runtime. Sizes of most of the data structures are derived from the maximum possible number of blocking rules which can be uploaded into FPGA and from the number of amplification rules. Assuming n is the number of IP addresses in the live traffic sample and k is the number of amplification rules, the memory analysis of the augmented mitigation system can be described as follows:

- size of RST = $n\times$ RST row: IP address (16 B), traffic volume contribution (8 B), reputation score (1 B) – $32n$ bytes if we consider 7 bytes of alignment; which is twice the size of VT
- data structures necessary for communication with NERD is $k\times$ request buffer with IPv4 addresses ($4n$ B), response buffer with reputation scores ($8n$ B), and other items which are negligible in comparison with the ones already mentioned – the total of $\sim 12nk$ B
- size of the reputation score cache: $n \times k$ rows of *Fast Hash Table* without using a stash, which can be further estimated with the knowledge that the table has 4 columns, size of the key is 16 B and size of the reputation score is 1 B – approximately $86kn$ B

In summary, the augmented mitigation system requires around $16n+98nk$ more bytes of operation memory than the previous version of DMD. For the

default values of $n = 2^{17}$ and $k = 1$, the estimated additional memory requirement is around 15 MB. Note that most of the required memory is due to the size of the reputation cache to limit the number of cache misses. Therefore, the mitigation heuristic is able to achieve better results and the reputation database does not get overloaded with large requests.

Conclusion

The theoretical part of this thesis introduces certain DDoS attacks (especially DDoS amplification attacks) and summarizes the research on the mitigation of DDoS attacks. It also discusses the research behind reputation scores and suggests a method of utilizing the reputation scores during the mitigation process of DDoS amplification attacks. The idea behind the inclusion of the reputation scores into the mitigation process is to eliminate the undesirable disruption of the portion of legitimate connections, which is the shortcoming of the majority of mitigation heuristics. The last section of the theoretical part is dedicated to the description of the reputation database (Network entity Reputation Database – NERD) and the scrubbing center (DDoS Mitigation Device – DMD). Both mentioned security tools, which are developed by *CESNET a.l.e.*, are used for the implementation and the evaluation of the proposed heuristic.

The new mitigation heuristic, called *RepTopN*, was designed based on the requirements analysis. The *RepTopN* heuristic combines the traffic volume contribution of the observed IP addresses with the knowledge of their reputation score to effectively mitigate the ongoing attack while preserving most of the legitimate connections. The contribution of this thesis also resides in the implementation of NERD API endpoint for bulk queries which was necessary in order to achieve a reasonable way of communication between DMD and NERD. The severity of the processing time was emphasized in the design and implementation of the whole augmented mitigation system. Therefore, several steps (e.g., reputation score cash, additional detached threads) were employed to reduce the processing time of the system since the CPU time is essential for the mitigation of large-scale DDoS attacks.

The *RepTopN* heuristic respectively the augmented mitigation system were practically evaluated and compared with the previous version of the mitigation heuristic respectively DMD. The experiments confirmed the expectations that the performance of the *RepTopN* algorithm is directly proportional to the number of IP addresses found in the reputation database. The proposed

heuristic is not very sensitive to the inaccuracy of reputation scores unless the deviation of a specific reputation score exceeds 0.2. Overall, the mitigation efficiency of the augmented mitigation system with the proposed *RepTopN* heuristic is considerably higher than the efficiency of the previously used heuristic in most of the tested cases while the rise of the CPU time is negligible. Therefore, the proposed system offers a solution to DDoS amplification attacks since it may reduce the amount of discarded legitimate traffic several times.

There are several topics which should be recommended for future research and development. It is necessary to assess certain aspects of the augmented mitigation system (e.g. size of the requests sent to NERD) during real DDoS attacks and adjust these parameters accordingly. Based on the research behind reputation scores, it would be worth exploring the possibility to integrate reputation scores of other entities, such as subnets, ASNs, or BGP prefixes into the *RepTopN* heuristic.

Bibliography

[1] CESNET a.l.e. CESNET / CESNET2 Network. [cit. 2018-04-24]. Available from: https://www.cesnet.cz/services/ip-connectivity-ip/cesnet2-network/?lang=en

[2] Mirkovic, J.; Reiher, P. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *SIGCOMM Comput. Commun. Rev.*, volume 34, no. 2, Apr. 2004: pp. 39–53, ISSN 0146-4833, doi:10.1145/997150.997156, [cit. 2018-04-24]. Available from: http://doi.acm.org/10.1145/997150.997156

[3] Gantner. 8.4 Billion Connected "Things" Will Be in Use in 2017. [cit. 2018-04-27]. Available from: https://www.gartner.com/newsroom/id/3598917

[4] Arbor Networks. The History of DDoS. [cit. 2018-04-27]. Available from: https://www.arbornetworks.com/the-history-of-ddos

[5] Rosencrance, L. Teen hacker 'Mafiaboy' sentenced. [cit. 2018-04-28]. Available from: https://www.computerworld.com/article/2583318/security0/teen-hacker--mafiaboy--sentenced.html

[6] Krebs, B. Akamai on the Record KrebsOnSecurity Attack. [cit. 2018-04-28]. Available from: https://krebsonsecurity.com/2016/11/akamai-on-the-record-krebsonsecurity-attack/

[7] Kolias, C.; Kambourakis, G.; et al. DDoS in the IoT: Mirai and Other Botnets. *Computer*, volume 50, no. 7, 2017: pp. 80–84, ISSN 0018-9162, doi:10.1109/MC.2017.201, [cit. 2018-04-26].

[8] Newman, L. H. GitHub Survived the Biggest Attack Ever Recorded. [cit. 2018-04-26]. Available from: https://www.wired.com/story/github-ddos-memcached/

[9] Ferguson, S. Arbor Networks: 1.7Tbit/s DDoS Attack Sets Record. [cit. 2018-04-28]. Available from: https://www.securitynow.com/author.asp?section_id=613&doc_id=741202

[10] Zargar, S. T.; Joshi, J.; et al. A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Communications Surveys Tutorials*, volume 15, no. 4, Fourth 2013: pp. 2046–2069, ISSN 1553-877X, doi:10.1109/SURV.2013.031413.00127.

[11] Trusted Knight. DDoS Atacks: 3 Common Motivations. [cit. 2018-04-28]. Available from: https://www.trustedknight.com/blog/ddos-attacks-3-common-motivations/

[12] Aly, A. A.; Barka, E. Tracking and Tracing Spoofed IP Packets to Their Sources. *College of IT, aly@ uaeu. ac. ae*, 2005.

[13] Peng, T.; Leckie, C.; et al. Protection from distributed denial of service attacks using history-based IP filtering. In *Communications, 2003. ICC '03. IEEE International Conference on*, volume 1, May 2003, pp. 482–486 vol.1, doi:10.1109/ICC.2003.1204223.

[14] Stratusly. What is a DDoS Scrubbing Center? [cit. 2018-05-3]. Available from: https://stratusly.com/what-is-a-ddos-scrubbing-center/

[15] Rossow, C. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *NDSS*, 2014.

[16] US-Cert. TA14-017A: UDP-Based Amplification Attacks. [cit. 2018-05-4]. Available from: https://www.us-cert.gov/ncas/alerts/TA14-017A

[17] Gavrichenkov, A. Understanding the facts of memcached amplification attacks. [cit. 2018-05-4]. Available from: https://blog.apnic.net/2018/03/26/understanding-the-facts-of-memcached-amplification-attacks/

[18] Moura, G. C. M.; Sadre, R.; et al. Internet Bad Neighborhoods: The Spam Case. In *Proceedings of the 7th International Conference on Network and Services Management*, CNSM '11, Laxenburg, Austria, Austria: International Federation for Information Processing, 2011, ISBN 978-3-901882-44-9, pp. 56–63, [cit. 2018-04-24]. Available from: http://dl.acm.org/citation.cfm?id=2147671.2147681

[19] Shue, C. A.; Kalafut, A. J.; et al. Abnormally malicious autonomous systems and their internet connectivity. *IEEE/ACM Transactions on Networking (TON)*, volume 20, no. 1, 2012: pp. 220–230.

[20] Moura, G. C. M.; Sadre, R.; et al. Internet bad neighborhoods aggregation. In *2012 IEEE Network Operations and Management Symposium*, April 2012, ISSN 1542-1201, pp. 343–350, doi:10.1109/NOMS.2012.6211917.

[21] Moura, G. C. *Internet bad neighborhoods*. Dissertation Thesis, University of Twente, 2013.

[22] Bartos, V.; Zadnik, M. An analysis of correlations of intrusion alerts in an NREN. In *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2014 IEEE 19th International Workshop on*, IEEE, 2014, pp. 305–309.

[23] Bartoš, V.; Kořenek, J. *Evaluating Reputation of Internet Entities*. Cham: Springer International Publishing, 2016, ISBN 978-3-319-39814-3, pp. 132–136, doi:10.1007/978-3-319-39814-3_13, [cit. 2018-04-24]. Available from: `https://doi.org/10.1007/978-3-319-39814-3_13`

[24] Zhang, J.; Porras, P. A.; et al. Highly Predictive Blacklisting. In *USENIX Security Symposium*, 2008, pp. 107–122.

[25] Soldo, F.; Le, A.; et al. Blacklisting recommendation system: using spatio-temporal patterns to predict future attacks. *IEEE Journal on Selected Areas in Communications*, volume 29, no. 7, 2011: pp. 1423–1437.

[26] Puš, V.; Kučera, J.; et al. Protector: DDoS mitigation at 100G. 2017.

[27] CESNET, a.l.e. DDoS Protector. [cit. 2018-04-24]. Available from: `https://www.liberouter.org/technologies/ddos-protector/`

[28] Seward, H. H. *Information sorting in the application of electronic digital computers to business operations*. Dissertation thesis, Massachusetts Institute of Technology. Department of Electrical Engineering, 1954.

Appendix A

Acronyms

ACL Access Control List

API Application Programming Interface

ASCII American Standard Code for Information Interchange

ASN Autonomous System Number

BGP Border Gateway Protocol

CPU Central Processing Unit

DDoS Distributed Denial of Service

DoS Denial of Service

DMD DDoS Mitigation Device

DNS Domain Name System

FBI Federal Bureau of Investigation

FPGA Field-programmable Gate Array

FTP File Transfer Protocol

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

ICMP Internet Control Message Protocol

IMAP Internet Message Access Protocol

IoT Internet of Things

IP Internet Protocol

IRC Internet Relay Chat

ISP Internet Service Provider

JSON JavaScript Object Notation

NERD Network Entity Reputation Database

NREN National Research and Education Network

NTP Network Time Protocol

P2P Peer-to-Peer

REST Representational State Transfer

RS Reputation Score

RST Reputation Score Table

SIP Session Initiation Protocol

SQL Structured Query Language

TCP Transmission Control Protocol

UDP User Datagram Protocol

URI Uniform Resource Identifier

URL Uniform Resource Locator

USB Universal Serial Bus

VT Volume Table

XML Extensible Markup Language

XSS Cross-site Scripting

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `src`
  - `impl` ..... the directory of implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text`
  - `DP_Jánský_Tomáš_2018.pdf` ..... the thesis text in PDF format
  - `thesis_assignment.pdf` ..... the thesis assignment in PDF format