Parallel Dissector

Parallel Processing of DDoS Data

M.Sc. Thesis

A. Kazemi Koohbanani

Delft University of Technology

TU Delft

Parallel Dissector

Parallel Processing of DDoS Data

by

A. Kazemi Koohbanani

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on July 11, 2023.

Student Number:	4895320
Project Duration:	November, 2022 - July, 2023
Thesis committee:	Prof. G. Smaragdakis, TU Delft
	Dr. G. C. Moreira Moura, TU Delft and SIDN Labs
	Dr. Q. Wang, TU Delft

Preface

This thesis was created as part of the Cybersecurity specialization of the M.Sc. program in Computer Science at the Delft University of Technology.

The research focuses on understanding how DDoS attacks are executed using the traffic data received by victims of previous attacks. We develop an algorithm to analyze network traces of any size with limited hardware requirements. In addition, we investigate the characteristics of several DDoS attacks by analyzing the attack vectors that compose them and discuss the limitations of our analysis. Lastly, our study on spoofing of packet headers allows us to determine the distribution of operating systems among the sources of the attack. Our insights and tooling could be used to further improve the performance of detection and mitigation systems.

I would like to thank my thesis advisors Georgios Smaragdakis and Giovane Moura for their continued support throughout the entire process. Their help and feedback were critical to making this report a success. Furthermore, I would like to thank my thesis committee Qing Wang.

Lastly, I would also like to acknowledge my family and friends for their support and encouragement during my studies.

A. Kazemi Koohbanani
Delft, July 2023

Abstract

Distributed Denial of Service (DDoS) leverages the power of multiple servers to disrupt the operations of a victim service. Due to the financial risks posed by downtimes on critical online infrastructure, DDoS is among the top threats in the cybersecurity landscape.

In this paper, we analyze the characteristics of previously launched DDoS attacks using collected network data. To extract the characteristics from a network trace file, we expand the DDoS Dissector tool with additional statistics representing the peak traffic strength and the sources of the attack. In addition, we implement an algorithm to parallelize the analysis of large-scale attacks when executed in memory-constrained environments. Our results show that the error difference in the statistics obtained when running the parallelized version and the original one is less than 0.5%.

Furthermore, we investigate several DDoS attacks by analyzing the contained attack vectors and their corresponding characteristics. Our software correctly detects the attack vectors, however, we remark that the output quality is impacted by the percentage of non-attack traffic. In particular, we provide an overview of the current state of the DDoS landscape seen from the point of view of a scrubbing service and study the effect of a Booter takedown on the frequency of DDoS attacks. Lastly, we introduce spoof detection techniques based on the time-to-live value found in the packet headers. From the spoofing analysis, we can deduce the distribution of operating systems that make up the sources of the attack.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
AS	Autonomous System
ASN	Autonomous System Number
BGP	Border Gateway Protocol
Bpp	Bytes per packet
bps	Bits per second
DDoS	Distributed Denial-of-Service
DNS	Domain Name System
HTTP	Hypertext Transfer Protocol
IXP	Internet Exchange Provider
IP	Internet Protocol
NTP	Network Time Protocol
pps	Packets per second
TCP	Transmission Control Protocol
TTL	Time To Live
UDP	User Datagram Protocol
URI	Uniform Resource Identifier

1 Introduction

In this chapter, we describe the motivation behind the work and the corresponding research questions we are going to answer throughout the paper. Next, we list our main contributions and provide an overview on the structure of the following sections.

1.1. Motivation

Denial-of-Service (DoS) is a technique used by malicious actors to disrupt the operations of one or more computer networks or systems. If the attacker is able to overload the victim service, regular users will not be able to access it, and therefore it could result in a significant loss for the service operators. The most prevalent attack vector leverages the power of multiple servers to target a specific service. When several servers are used to launch a DoS attack, it is referred to as Distributed Denial-of-Service (DDoS). The first DoS attack was launched in 1996 [1] towards the internet service provider (ISP) Panix [2] and it consisted of sending as many TCP SYN packets as possible to the victim. The SYN packets represent a request to establish a TCP connection between the attacker and the target. In this case, the target cannot handle the large amount of requests being received and it becomes inaccessible to any user trying to access it.

Due to the financial damage it can cause, DDoS is among the leading threats in the cybersecurity landscape [3]. DDoS attacks are executed through amplification methods such as DNS exploits [4], [5], or using multiple devices controlled by a single operator known as Botnets [6]. For example, Mirai infected IoT devices (e.g., cameras) and routers to target well-known websites and game servers; investigations have shown that there were multiple independent operators, as well as how the botnet evolved over time [7]–[9]. Recently, powerful attacks stronger than Mirai have been launched and reported about [10], the largest recorded to date was stopped by Microsoft and peaked at around 3.47 Tbps [11].

Among the various amplification methods, the DNS protocol is commonly exploited to generate large amounts of traffic towards a single server. A DNS server receives a request for the DNS records of a particular domain and will respond to the original sender with the information. The DNS protocol uses UDP on the transport layer, there is no state or connection being established, thus anyone can impersonate a different IP address' request. This means that it is possible to craft a packet with a DNS request for a victim IP, send it to a DNS server, and the response will be sent to the victim. In particular, DNS queries can be as small as 64kb, but their responses are up to 179 times larger [12], for example, when using the DNSSEC features. There are options to avoid abuse that a DNS server can take; however, millions of DNS servers remain misconfigured [13] and can be used to amplify the attack. A list of misconfigured DNS servers can be put together by scanning the entire internet IP address space for the specific vulnerability. The scan is feasible because there are at most 4 billion IPv4 addresses and the vulnerability check takes a single DNS request time. As such, using a list of vulnerable servers, we can start an attack towards a specific IP by sending the crafted query to all DNS servers in the list. It should be noted that for the exploit to be successful, spoofed packets should be sent to the DNS server to impersonate the victim, most internet service providers will block such packets.

In most cases, these attacks are launched using booters that offer such services at a competitive price, generally below $10 [14]. Due to the low-cost, DDoS attacks can be used to extort money from victims that cannot properly defend themselves [15]. In particular, there are several options to defend against a DDoS attack but deploying such defenses is much more expensive. The ISP housing the servers typically deploys

DDoS detection and mitigation systems to protect its customers. However, those do not provide protection against advanced or powerful attacks; hence, multiple anti-DDoS companies have been founded over the years, such as Imperva [16] or Cloudflare [17]. These companies provide anti-DDoS services, the traffic goes through their infrastructure and the attacks are mitigated before they reach the customers' servers. Although solutions to DDoS exist, they become more expensive to deploy when the attacks get stronger and therefore are usually out of reach, except for large companies.

Our intention is to understand how DDoS attacks are executed by analyzing them from the point of view of the victim. In particular, we wish to understand the different attack vectors that compose a single attack and their unique characteristics.

1.2. Research Questions

The research focuses on finding a data-driven approach to analyze DDoS attacks. Using network data of previous attacks, we create signatures of the measured attack vectors summarizing the main characteristics and parameters employed by the attacker. In particular, we investigate the characteristics and categories of the discovered attacks to understand their similarities and differences across various types of attacks originating from several sources. To conduct the research, we will answer the following questions.

How can we scale up and parallelize the analysis of DDoS attacks while preserving the properties of the attack? Using network data, we are going to analyze the characteristics of large-scale attacks, extracting key parameters unique to each attack vector. The aim is to generate a summary that contains the most important features of a previously detected DDoS attack, which we call a signature. The signature contains information that can be extracted from traffic data, such as port number, protocol, or packet size. The extracted parameters can both be used to detect attacks by comparing them to the signature obtained and to analyze how various attacks differ from each other. In particular, large-scale attacks send more bytes in a few minutes than the memory capacity of a single machine, therefore we wish to find a method to systematically analyze these files with the memory restrictions of our infrastructure. We can further generalize the algorithm to fit any memory requirement and show the trade-off in the results quality. For example, we can split the input into smaller chunks which can be individually analyzed and later combined to create the original signature.

How can we efficiently identify single and multi-vector attacks? In this case, we are looking to extract meaningful information from a dataset of previously launched DDoS attacks using our generated signatures. In particular, we would like to find what the main characteristics are that help us distinguish between different kinds of attacks. For example, we can categorize each attack according to the detected attack vectors such as DNS amplification or TCP flood. Furthermore, we can distinguish attacks using a single exploit from the ones that contain several attack vectors. Next, we can further define the unique features of the attack by looking into specific parameters of each vector. In particular, from the signature we can extract the attack's strength, whether they are sending spoofed packets, or the device characteristics of the sources. We can use the information to gain a deeper understanding on the differences that these attack vectors display. Moreover, the insights obtained could be used to help advance DDoS detection systems or analyze well-known large-scale attacks. Lastly, we want to address the limitations of our analysis with respect to the quality of results obtained from different dataset sources. Given the limited availability of data, we can show the main constraints of our methodology that could also be applicable to production environments.

How can we infer and attribute an attack to a single or multiple attackers? Large-scale attacks leverage the combined power of multiple servers to disrupt a single service. In the case of botnets, these servers are controlled by an operator who is sending commands to each host, instructing them to start the attack. However, the attack could also be launched by a single machine and the victim could still be attacked by several sources, in case the packets have been spoofed. We would like to distinguish between these two types of attacks. We are going to develop techniques to determine whether a DDoS trace is likely to contain spoofed packets. In particular, we can corroborate the analyzed attack's characteristics with the likelihood that the packets were spoofed and conclude whether the obtained insights are plausible.

1.3. Contribution

In this work, we make the following contributions. We build on previous work focused on extracting fingerprints from DDoS attack traces. In particular, we include additional features to the final fingerprint to provide a deeper insight into the sources of the attack. Moreover, we address the scalability challenges that arise when analyzing attacks that are larger than the available memory. We propose a method to split the attack trace into smaller chunks that can be independently processed, and then show how the resulting fingerprints can

be merged to a single one with negligible difference in the results quality. Further, we develop heuristics to detect whether a network trace file is likely to contain a significant portion of packets that have been spoofed. Lastly, we use the fingerprint extraction and spoof detection to analyze several datasets containing DDoS attacks. From the signature analysis, we obtain the main attack vectors used and their peculiar characteristics. This allows us to analyze a recent dataset of attacks launched towards a scrubbing service. In particular, we provide an overview of the current state of the DDoS landscape and study the effect of a Booter takedown on the frequency of DDoS attacks. In addition, we highlight the limitations of our methodology especially when the attack trace contains a significant percentage of packets that do not belong to the attack. Furthermore, the spoofing analysis allows us to determine the distribution of the operating systems running on the attack sources.

1.4. Report Structure

The report is structured as follows. Chapter 2 summarizes the most common techniques used when performing DDoS attacks, as well as defending and mitigating them. Then, an overview of the current literature on well-known DDoS attacks, detection, and mitigation techniques is presented in Chapter 3. Chapter 4 details the architecture and algorithms used to analyze the attacks, including the solutions to the scalability challenges encountered during the analysis of larger attacks. The characteristics of several DDoS attacks are discussed in Chapter 6 followed by a reflection on the data analysis and ethical considerations in Chapter 7. Lastly, Chapter 8 draws the conclusions about the data-driven analysis of DDoS attacks based on the insights discovered and proposes future research directions.

2 Background

This chapter provides the knowledge of DDoS attacks required to follow the rest of the paper. First, we explain the differences between DoS and DDoS as well as how to execute some of the most common attack vectors. Next, we illustrate several detection and mitigation strategies that are generally found in commercial applications.

2.1. Denial-of-Service Attacks

Denial-of-Service (DoS) attacks are carried out by overloading a server or network, sending a large amount of packets and ultimately causing it to be unreachable. For example, in an ICMP flood attack the adversary sends sequential PING requests to the victim such that the receiver bandwidth cannot handle all the requests. This means that an attacker with enough bandwidth can overload a victim service with lower network capacity. In practice, the bandwidth of a single machine is limited and can only be used to take down servers with less capacity. Therefore, to execute large-scale attacks, it is commonly required to take advantage of multiple servers attacking at the same time, multiplying the power of a single machine by the size of the network. In this case, the attack is called a Distributed Denial-of-Service (DDoS) and is executed using a compromised computer network, mainly through amplification exploits or controlled hosts, called Botnets [5]. Figure 2.1 displays the difference between the two approaches, the same type of attack can be replicated on multiple machines in the distributed case.

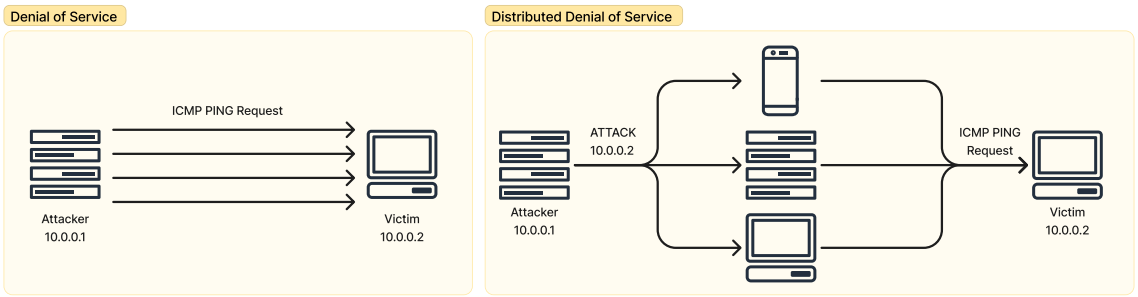

Figure 2.1: Difference between DoS and DDoS.

In the case of Botnets, the adversary needs to infect multiple devices and gain control of their system. For example, IoT devices are commonly connected to the Internet via simple interfaces which can be vulnerable to several exploits. Since the number of these devices is significantly large and growing, successfully exploiting them results in attacks that can be quite powerful, for example, Mirai [8]. On the one hand, Botnets can be effective at launching large-scale attacks, even though they require access to the system and a communication channel with a controller. On the other hand, amplification attacks require the system to be vulnerable but not to the same extent. In particular, a successful amplification attack requires the running application or protocol to include a command that sends a larger response than the request that originated it. Moreover, the protocol should not verify the sender such that an attacker can impersonate a victim request, resulting in the response

Protocol	Source Address	Destination Address	DNS Query Request	Bits
DNS	10.0.0.2:80	1.1.1.1:53	NS root-servers.net	792
DNS	10.0.0.2:80	8.8.8.8:53	NS root-servers.net	792
DNS	10.0.0.2:80	8.8.8.4:53	NS root-servers.net	792
DNS	10.0.0.2:80	9.9.9.9:53	NS root-servers.net	792

Table 2.1: Example request DNS queries for an amplification attack.

to be sent to the victim instead. Impersonation is commonly executed by sending a spoofed packet, i.e., the source Internet Protocol (IP) address contains the victim IP address instead of the attacker's one.

2.1.1. DNS Amplification

Several vulnerabilities have been found in different Internet protocols such as the domain name system (DNS), the network time protocol (NTP), and the character generator protocol (CHARGEN). In particular, the most successful vulnerabilities have a large amplification factor, that is, the ratio between the size of the response and the size of the request. In the case of the DNS, a single request can generate a response that is up to 179 times larger [12]. For example, to launch a DNS amplification attack the attacker sends a DNS query, such as ANY, where the packet source IP address is the one of the victim. The DNS server will send the reply to the victim's IP address since the attacker spoofed it. For the attack to work, the adversary needs a list of vulnerable DNS servers that will respond correctly to a request, since the DNS provider could try to prevent the amplification exploit. However, checking for vulnerable servers simply requires sending a single packet and verifying whether the response is received or not. Therefore, a list of vulnerable servers can be created by scanning the entire IPv4 address space on the Internet, from 0.0.0.0 to 255.255.255.255. Since there are around 4 billion IPv4 addresses, the scan can be done in a short period of time. An example of DNS amplification attack is shown in Figure 2.2, the attacker sends the same DNS query request to a list of DNS servers. Note that the source IP address is the one of the victim, thus the DNS servers will send a large reply to that IP address.

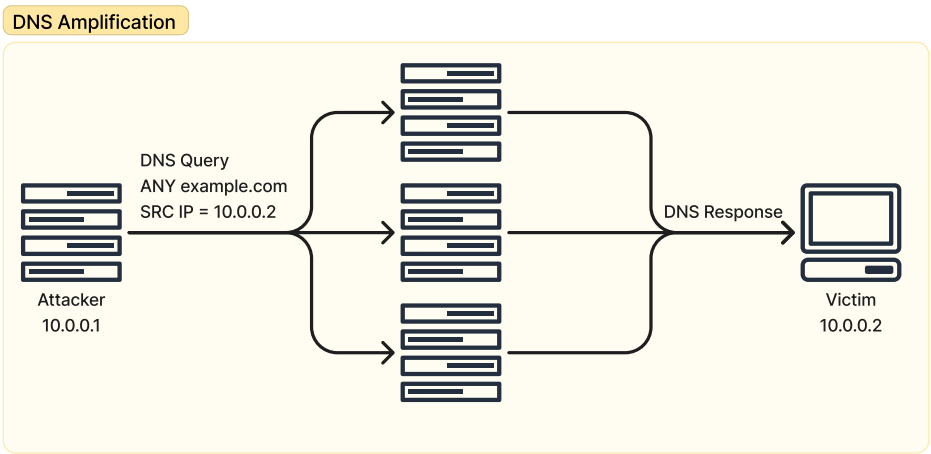

Figure 2.2: Example of a DNS amplification attack.

An example of packet sequences for the requests sent by the attacker and responses received by the victim is shown, respectively, in Table 2.1 and Table 2.2. In this case, the victim is hosting an HTTP server at the IP address 10.0.0.2, therefore, the same address is added as the spoofed source IP address for all requests made by the attacker. Then, the packets are sent to multiple DNS servers that are known to reply, the query asks for the nameservers (NS) of the domain name ROOT-SERVERS.NET. Although the size of the query request is 792 bits, the response can be up to 6936 bits. Note that the response size may differ depending on the server implementation. However, all responses are larger than the request, which means that we can carry out an amplification attack with enough vulnerable servers. Since the average amplification factor is about 6 times, using a single 1 Gbps server, we could send a 6 Gbps attack. In practice, an adversary would craft a request with the highest amplification factor possible, for example by adding large TXT records to a domain they control.

Protocol	Source Address	Destination Address	DNS Query Response	Bits
DNS	1.1.1.1:53	10.0.0.2:80	NS root-servers.net	6936
DNS	8.8.8.8:53	10.0.0.2:80	NS root-servers.net	2360
DNS	8.8.8.4:53	10.0.0.2:80	NS root-servers.net	2360
DNS	9.9.9.9:53	10.0.0.2:80	NS root-servers.net	6936

Table 2.2: Example response DNS queries for an amplification attack.

2.1.2. NTP Amplification

Similarly, the NTP can be exploited to carry amplification attacks. In particular, when sending the monlist command, the NTP server responds with the latest 600 IP addresses that sent a synchronization request. The returned list size can be 200 times larger than the request [18]. Therefore, an NTP amplification attack can reach peaks of 200 Gbps when launched from a server with 1 Gbps capacity. Accordingly, the exploited NTP servers need to have enough capacity to reach the maximum amplification factor. Similar to the DNS amplification case, a list of vulnerable servers is computed by scanning the Internet for servers that correctly respond to the monlist command. Therefore, the maximum strength of an attack is determined by the current number of vulnerable servers worldwide. In fact, the monlist command can be disabled, and it is mostly found to be enabled in misconfigured or older servers [5]. In this case, the port exploited is 123 because that is where the NTP server runs. The transport layer needs to be UDP, thus launching an NTP amplification attack can be done from a single server spoofing the packets such that the source IP address is the victim's address. This way the NTP synchronization requests is sent to the victim.

2.2. DDoS Detection

Since DDoS attacks aim to render certain services inaccessible, detection could be done by checking whether the server is receiving more traffic than what its available bandwidth allows. We could build a detection system based on a threshold that, if exceeded, will trigger a response mechanism to mitigate the attack [19]. For example, the threshold can be set at 1 Gbps, as that is also the port capacity available to the majority of servers. Although a threshold method will not be able to detect sophisticated attacks targeting a vulnerable application with low traffic, volumetric attacks could be identified with this simple rule. To improve this approach, the received traffic can be compared to known attack traffic using association rules. For example, DNS amplification attacks contain traffic originating from DNS servers hosted on port 53, and they carry DNS responses. In this case, the threshold could be applied only to DNS traffic, especially if it comes from several sources. To create association rules, one could record the traffic received during an attack and summarize a subset of parameters that can be used to identify it, such as the source port number or the protocols used. Additionally, rules can be simplified whenever they represent the same attack family, generalizing their characteristics, and improving the chances of detecting more attacks of the same type.

Alternatively, machine learning models have been developed to detect ongoing DDoS attacks [20]. Similar to the association rules method, a supervised machine learning model can be trained on old traffic data and then used to predict future data points. Although a supervised model is straightforward to implement, it requires having labeled data from old attacks and assumes that future traffic will be similar to the training data. For the latter point, we know that this is generally not the case as novel attacks appear due to vulnerabilities found in newer applications. In this case, a model can be trained to detect anomalies in the traffic patterns and signal to an operator whenever non-regular traffic is seen. Anomaly detection methods might also detect other kinds of attacks that are not DDoS, such as hacking attempts. However, they can be used to enhance the capabilities of a detection system by adapting to new attack strategies.

2.3. DDoS Mitigation

In this section, we discuss several mitigation approaches that can be applied whenever a new attack is detected. While there are other methods of defense, the followings are the most commonly used in practice.

2.3.1. BGP Blackholing

The Border Gateway Protocol (BGP) is used by routers to determine the route a packet needs to take while traveling between Autonomous Systems (AS). The protocol consists of exchanging reachability information so that each router receives a list of destinations with their respective distance. Then, all other routers can

update their own distances based on the ones just received, determining the shortest path for each possible host. Routing information is frequently shared since hosts can go down or shorter routes are found. In addition, the protocol can be used to mitigate DDoS attacks, shifting traffic from overloaded servers to infrequently used ones. Alternatively, the traffic can be temporarily sent to a null interface, possibly dropping some legitimate traffic. The latter technique is called blackholing and uses BGP communities to notify the other ASes to drop all packets that belong to a specific IP prefix [21]. In this case, the IP prefix is usually the victim's IP address, but it could also be a larger subnet, such as /28. The downside of using subnets less specific than /32 is that the collateral damage increases while the attack might be targeted at a single IP address. To blackhole an IP prefix, the AS can tag the IP prefix with a number identifying the route as a blackhole, typically the community used is ASN:666 where ASN is the number identifying the AS [22]. However, the specific community number used depends on the AS and some might not have implemented it at all. Figure 2.3 shows how BGP blackholing can be used to limit traffic received during a DDoS attack. In this case, AS2 detects an attack on the IP address 10.0.0.2 and a request to blackhole the prefix 10.0.0.2/32 is sent to neighboring ASes. Then, it is up to each individual AS to decide if they should honor the blackholing request, sending all traffic destined for that IP address to a non-existent interface.

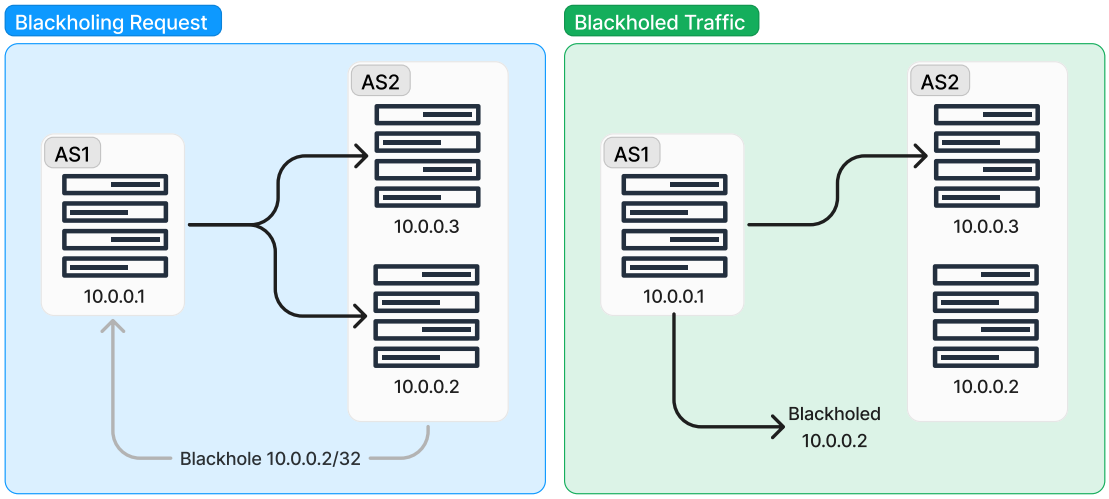

Figure 2.3: Example of using BGP blackholing to mitigate a DDoS attack.

2.3.2. Anycast

Anycast is a technique used to assign multiple physical servers the same IP address and spreading the traffic between them, often sparsely located across the world [23]. In particular, physical networks are divided into different catchment sites and the BGP route information specifies which site should be used. In case of a DDoS attack, anycast enables traffic operators to send traffic to catchment sites with enough bandwidth to absorb it [24]. Figure 2.4 shows an example of using BGP and anycast to route traffic so that physical servers are not overloaded. In particular, the top server is receiving 25 Gbps more than its capacity; hence, the BGP is used to advertise more routes toward the bottom server, able to handle more traffic. Note that the BGP is a distributed protocol, which means that the updated routes will not propagate instantly. Therefore, anycast mitigations cannot be applied frequently and require precision when advertising different routes to avoid causing further outages.

2.3.3. Traffic Scrubbing

While BGP traffic engineering can be used to mitigate DDoS attacks, with enough hardware, a single operator can handle attacks of any size. For example, scrubbing services are tasked with receiving all traffic and forwarding a cleaned version of it, sending the attack traffic to a blackhole [25]. In this case, the scrubbing hardware has enough capacity to process a much larger number of packets and bandwidth than the actual receiver. Since the traffic is sent directly to the scrubbing service, the attacker will need to overload the specialized hardware at the scrubber instead of the generally smaller victim server. The scrubbing service does not need to be con-

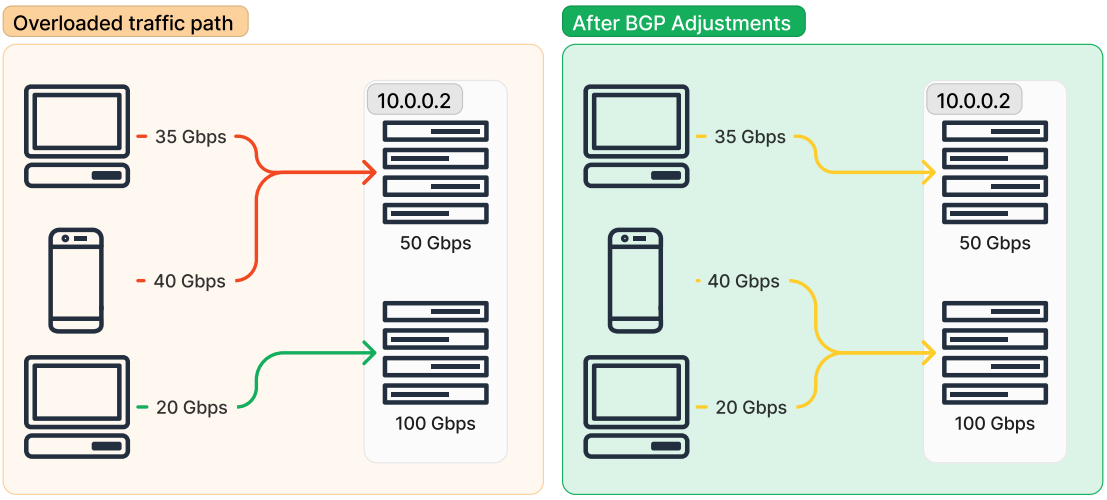

Figure 2.4: Example of using BGP and anycast to mitigate a DDoS attack.

stantly receiving traffic, but could be enabled only when a DDoS attack is detected, for example, by changing a BGP route or a DNS record. Moreover, firewall rules can be specified to drop all traffic for commonly used protocols that are not used by the server applications. For example, the Memcached protocol is vulnerable to amplification attacks when the UDP port is exposed to the Internet. Therefore, disallowing UDP traffic will prevent the exploit from being executed. The advantage of traffic scrubbing over previously explained BGP techniques such as blackholing is that the scrubber can use specialized techniques to detect and remove DDoS traffic. The scrubbing center receives the full traffic; therefore, they can use that data to train a tailored machine learning model. The model is able to differentiate attack packets from regular traffic and thus resulting in a reduced loss of packets that might be dropped if the traffic was instead blackholed.

3 Related Work

This chapter presents previous research work in the area of DDoS attacks. First, we highlight research on recorded attacks and then list different studies on detection and mitigation techniques.

3.1. Analyzing DDoS Attacks

In this section, we list different approaches to the analysis of DDoS attacks. In particular, we differentiate between large-scale events that caused significant damage, Booter-specific studies, and the general impact of DDoS attacks studied from diverse perspectives.

3.1.1. Large-scale Events

Several studies focused on evaluating major DDoS attacks to understand their specific characteristics. Moura et al. [23] analyze the 2015 Root DNS Event from public data; in particular, they look into the resilience of anycast deployments from DDoS attacks. From the analysis, they show how the reachability of the Root Letters depended on the different anycast policies applied by the operators. In some cases, the attack caused collateral damage to closely connected networks and servers. Moreover, the collateral effect of DDoS attacks on neighboring services and DNS infrastructure has been analyzed using data from the Dutch National Scrubbing Service (NaWas) [25]. In particular, attacks directed at websites using a shared hosting platform impact all other websites that share the same IP address. Correlating DNS data from the .NL registry operator (SIDN) with the scrubber information, they show how an increase in DNS traffic can be used to detect whether a DDoS attack is taking place. Antonakakis et al. [8] show how the Mirai Botnet spread across different devices and how more powerful versions were deployed over time. While IoT devices are commonly targeted for Botnets of this size, Mirai's device composition included camera equipment as well as routers, the latter playing a major role in a subsequent version exploiting the CPE WAN Management Protocol (CWMP). Furthermore, the Botnet evolved over time and later variants presented new exploits and introduced techniques to avoid detection.

3.1.2. Booters

Collier et al. [14] examine Booter services and the impact of law enforcement interventions on the DDoS landscape. In particular, they noticed a consistent decrease in the number of attacks following the closure of popular online stressers. Additionally, they show how law enforcement agencies began advertising Booter services themselves, educating the customers on the legality of such service and at the same time reducing the number of attacks. Furthermore, Santanna et al. [26] analyze the DDoS-for-hire landscape by measuring the attack traffic of 14 different online stressers, concluding that the amplifiers used do not generally overlap, and therefore the attacks originating from them could be even more powerful. Jonker et al. [4] use backscatter and honeypot data to characterize the DDoS ecosystem, as well as to demonstrate how victims of an attack migrate to managed security providers that offer DDoS protection in the following months. Similarly, DDoS-for-hire services use managed security providers such as Cloudflare to protect themselves from competitors sending DDoS attacks [26].

3.1.3. Impact of DDoS Attacks

Kopp et al. [5] study DDoS attacks from the point of view of an Internet Exchange Point (IXP), discovering several less known attack vectors that are being exploited. Additionally, they recognize that older attacks and exploits are still abused even though mitigations and patches have been available for a long time. Furthermore, they do not find any impact of COVID-19 on the frequency of DDoS attacks. Sommese et al. [27] correlate active DNS measurement with Internet Background Radiation to understand the increase in round trip time for a DNS query sent during a DDoS attack. They differentiate anycast DNS infrastructure from the rest, showing that anycast services are generally more effective in absorbing attack traffic. Rijswijk-Deij et al. [12] examine the effect of DNSSEC on the amplification factor of DDoS attacks. While the theoretical limit of regular DNS is close to 20x, DNSSEC-signed responses can amplify as much as 50-60x and in some cases up to 179x.

In this paper, we show how both Botnet and amplification based DDoS attacks are evolving over time as well as their main characteristics. In particular, we do not focus on a single event represented by a major attack, but we want to understand the overall picture, such as regularly occurring attacks or frequently targeted organizations.

3.2. DDoS Detection and Mitigation

The following sections present several approaches to the detection and mitigation of DDoS attacks. First, we look at the techniques that can be executed at the level of an IXP. Next, we illustrate techniques based on the BGP that are commonly used to mitigate attacks at any scale. Last, methods based on the new generation of networks, such as cloud infrastructure or software-defined networks (SDN), are analyzed.

3.2.1. Mitigation at the IXP

Wagner et al. [19] investigate whether collaboration between IXPs leads to a better overview and mitigation of detected attacks. In particular, the solution consists of sharing anonymized flow information between IXPs, maintaining the users privacy and improving detection capability. Furthermore, they did not find a significant performance decrease when sharing less private information compared to a high-trust environment. Dietzel et al. [21] introduce a new system to enhance the BGP blackholing capabilities at the IXP level. In particular, they enable the owner of an IP address to blackhole it on a specific port and protocol, filtering the related traffic at the egress of the IXP. This does not disrupt normal traffic operations unlike regular BGP blackholing and allows to completely mitigate most volumetric attack vectors. Wichtlhuber et al. [20] provide detection and mitigation strategies from machine learning methods built using blackholed traffic data. The model is continuously trained at the IXP level based on the signals received from connected ASes, and the resulting association rules are minimized to ensure that they can be interpreted by traffic operators. We instead focus on using signatures obtained from known attacks. In particular, we are interested in reducing the complexity of the rules, allowing us to incorporate detection capabilities in networks of any size.

3.2.2. BGP-based Mitigations

Rizvi et al. [24] investigate how anycast deployments can leverage a BGP playbook when mitigating DDoS attacks. In particular, they provide a method to estimate the size of the attack so that traffic engineers can adopt one of many precomputed rules to shift traffic between catchment sites. Giotsas et al. [22] demonstrate how BGP blackholing is currently used to mitigate DDoS attacks, especially by transit and access providers and IXPs. Further, they infer which BGP communities are used by different ASes when blackholing an IP address and provide methods to do so automatically. In addition, Jonker et al. [28] show how over a period of 1.5 years, the adoption of DDoS protection services using DNS or BGP-based mitigations increased. In particular, they can detect whether a customer is using the always-on option or whether the protection service is only enabled during an attack.

3.2.3. SDN and Cloud-based Mitigations

To overcome the large fixed cost of acquiring anti-DDoS hardware, Fayaz et al. [29] design a flexible and elastic mitigation system leveraging software-defined networking and network functions virtualization. Jia et al. [30] study how to mitigate DDoS attacks in a cloud environment, where spikes in traffic can cause large bills to be triggered due to autoscaling. In particular, they develop an infrastructure to reassign each client to a new server depending on whether their traffic is deemed legitimate; otherwise, the attack traffic is redirected to a few selected instances.

While our efforts are not directed at building a DDoS detection and mitigation system, our results are rele-

vant to further improve currently existing infrastructure. Leveraging our insights, any of the aforementioned techniques can be further improved and tailored to the continuously evolving DDoS landscape.

4

Methodology

In this chapter, we illustrate our approach to the analysis of DDoS attacks from network data. First, we detail the improvements that we have made to the DDoS Dissector software. Next, we show how we can parallelize the execution of the program so that we can run it with lower memory requirements. Lastly, we develop techniques to analyze whether the packets found in a network trace have been spoofed.

4.1. DDoS Dissector

To characterize the DDoS landscape, we need to programmatically process several network traces corresponding to different kinds of attack. In particular, our aim is to analyze at least one network trace for the most common attack exploits. In the following sections, we develop a procedure to analyze each network trace individually. The main piece of software used during the analysis is the DDoS Dissector (Dissector) [31]. We decide to build upon the already developed Dissector software instead of starting from scratch because many feature requirements for our research overlap with theirs. However, the Dissector does not fully satisfy our needs, therefore our aim is to augment the extraction techniques and enable the analysis of attacks of any size. We use the Dissector to create a unique fingerprint given the traffic capture of a DDoS attack, summarizing its main characteristics such as strength or the list of attack vectors used. The fingerprint can be used to identify the same or similarly executed attacks. For our purposes, the fingerprint provides the main characteristics that define the attack. The improved version of Dissector is open source and available at the following link: `https://github.com/paolokazemi/Parallel-DDoS-Dissector`.

Furthermore, we contribute some of our findings back to the original Dissector repository. In particular, the original version did not fully support the parsing of TCP flags when reading the contents of a PCAP file. With our suggestions, TCP flags can be parsed correctly both from PCAP and NetFlow files.

4.1.1. Input Attack Traces

Here, we highlight the different network trace types the original Dissector allows for and what information is extracted. The input can be either a PCAP [32] or a NetFlow [33] file, the only requirement being that the majority of the traffic corresponds to a DDoS attack. Since the Dissector assumes the traffic file represents a single attack, regular traffic should be found in smaller percentages otherwise the fingerprint will match the shape of the normal traffic. Additionally, the traffic can contain multiple attacks as long as they are targeted towards the same victim. In this case, the Dissector considers each attack as a separate attack vector, and the set of attack vectors belong to the same fingerprint. From both file types, we extract traffic information such as start timestamp, duration, protocols used, packet sizes, source and destination ports. Since PCAP files contain more detailed packet information, we collect additional fields that can be used to further define the exact parameters used to execute the DDoS attack. For example, DNS packets contain the query name and type and that can be used to analyze how the DNS amplification attack is executed from the attacker's perspective, e.g., find the query names with the highest amplification factor. In Table 4.1, the features extracted from the PCAP and NetFlow files are listed.

While the analysis can be carried out using NetFlow files, we opt to primarily use PCAP files because they provide more information and the runtime performance of the Dissector is not affected even for larger attacks. First, the packet information extracted from a PCAP file provides greater insights into how the attack

Parameter	NetFlow	PCAP
Start Timestamp	✓	✓
End Timestamp	✓	✓
Protocol	✓	✓
Source Address	✓	✓
Source Port	✓	✓
Destination Address	✓	✓
Destination Port	✓	✓
Number of Packets	✓	✓
Number of Bytes	✓	✓
TCP Flags	✓	✓
Ethernet Type	✗	✓
ICMP Type	✗	✓
IP TTL and Fragmentation Offset	✗	✓
NTP Request Code	✗	✓
DNS (Query Name and Type)	✗	✓
HTTP (URI, Method, User Agent)	✗	✓

Table 4.1: Information extracted from NetFlow and PCAP files.

works. For example, with the DNS query and name parameters we can figure out which query has the highest amplification factor. Moreover, extra information might reveal new exploits that are currently not yet publicly known. Second, we consider the time it takes to analyze large attacks given the limitations of the Dissector. In this case, the full attack trace is loaded into memory to be analyzed. While NetFlow files could provide a smaller file due to the lower information requirements, they do not result in lower memory overhead during our experiments. In particular, we notice that specific attack traces use an equivalent amount of memory to the original PCAP file, mainly due to spoofing. Since there are several spoofed sources in the attack, the NetFlow file contains a row for each of them, resulting in a memory requirement similar to the original PCAP file, because most of the packet data is removed except for the features contained in Table 4.1. We further explain how we tackle the challenge of analyzing files larger than the available memory and how we can scale the analysis to files of any size in Section 4.3.

However, we find traces that are neither PCAP nor NetFlow in our datasets. Therefore, we extend the Dissector reader functionality to support additional file formats. In particular, we implement support for Extensible Record Format (ERF) [34] files by reusing the PCAP reader. In this case, the tshark [35] utilities are already used to parse PCAP files, and since ERF can be read using tshark, we can apply the same method. Furthermore, we also find Argus [36] formatted traces in our datasets, and they are not compatible with any of our parsing strategies. For this case, we create an Argus reader utility that is similar to the NetFlow one. Instead of using the nfdump utility, it uses the read argus (ra) [37] program to parse the Argus file into our standardized format. The extracted features are the same as those found in Table 4.1, where the NetFlow column is used for the Argus files and the PCAP column for the ERF traces.

In the following sections, we explain how we obtain a fingerprint given an input attack trace.

4.1.2. Target Inference

The target inference algorithm was not altered from the original version and works as follows. Once the traffic information is loaded, the Dissector needs to infer the victim address targeted during the attack. If it is a single IP address, the host that received more than 50% of the traffic, measured in number of packets, is assumed to be the attacked IP address. Otherwise, an appropriate subnet from the destination IP addresses is searched, such that 70% of the attack flows belong to it. Currently, the subnet ranges are limited to /24 for IPv4 or /64 for IPv6. Additionally, the target can be manually specified, and the subsequent analysis is run on the traffic received by the selected destination IP address or subnet, multiple destinations can also be considered.

4.1.3. Extracting Attack Vectors

Given the attacked network, all traffic data is filtered except packets received by the victim hosts. Next, we need to extract the attack vectors found in the attack. For example, we want to differentiate which part of the traffic corresponds to a DNS amplification attack or a TCP SYN flood. And, we need to understand their

individual impact such as the peak strength or the number of sources participating in the attack. To extract the attack vectors, the data are grouped according to the source port and the protocol used. Then, the groupings representing more than 5% of the total packets are added to the list of attack vectors. Similarly, destination port and protocol pairs are collected in separate groups, where the threshold to be considered attack vectors is increased to 10%. To reduce the possible number of false positives, the computed attack vectors are further filtered by removing the ones where the percentage of attack traffic, measured in total bytes, is less than 5% of the total traffic.

The parameters and characteristics of each attack vector are extracted by selecting the values that appear in more than 10% of the total traffic corresponding to the specific vector. If no specific value is found in more than 10% of the traffic, the parameter is considered to have been randomized by the adversary. For example, an attack directed to a web service will likely result in the destination ports to be 80 or 443 for the majority of the traffic. However, without knowledge of the running applications, the attacker might choose to target multiple ports, resulting in a randomized destination ports value. In case we can identify the most common values we report their distribution, that is the percentage of vector traffic containing that specific value, such as 25% to port 80 and 75% to port 443. The distribution calculation is applied to all additional parameters derived from PCAP files found in Table 4.1. Creating a distribution of parameters is useful because sophisticated attacks will attempt to vary their traffic content, trying to avoid the detection systems of the victim. In particular, using multiple DNS queries instead of a single one will make it harder for the DNS operators to differentiate the attack from normal traffic. Moreover, different queries can have variable amplification factors and therefore the attacker can leverage those to control the strength of the attack.

Further, we extend the Dissector with the following characteristics. To understand the attacker's network, whether it is using amplification servers or botnets, we collect statistics of the sources found in the attack. In particular, we group the sources by their IP address and save the total number of bytes they sent, the number of packets, and their average bits per second (bps) and packets per second (pps). Further, after the analysis is complete we can derive the respective AS numbers from the IP address. We use the route views Prefix-to-AS mappings provided by CAIDA [38] based on the date of the attack, since the timestamp value is present both in the NetFlow and PCAP file. Similarly to the IP address statistics, we collect the same source statistics for each AS, as well as counting the total number of IP addresses belonging to the same AS. The mapping of IP to AS can also be used to anonymize the exact IP addresses contained in the attack, maintaining a balance between the privacy of the sources and the information provided to identify the characteristics of an attack.

Although average values provide information on the overall strength of an attack, they are not useful in determining whether a specific attack can be handled by a system because the peak value could be much higher, but for a shorter period of time. Therefore, we also calculate the respective peak values for bps, pps, and bytes per packet (Bpp) similarly to how they are done in the average statistics. In this case, we divide the traffic into fixed-sized buckets of 1 second based on their timestamp value. For each bucket, we calculate the attack strength and then take the maximum value of all buckets as the peak value. Attacks might not begin and end exactly at the start of a second therefore the peak values of those buckets will generally contain lower values. However, using the absolute timestamp value and not a relative one provides a deterministic method to calculate the peak value, which would not be achieved if the trace file would be split into multiple files when scaling the analysis. Moreover, the first and last seconds of an attack rarely correspond to the peak value because there is always a startup period before all nodes involved in the attack start sending packets.

4.1.4. Normal Traffic

When analyzing an attack, we assume that the majority of the traffic corresponds to an attack and only a minor part might be non-malicious traffic. However, for popular services, this minor part can provide further insight into what service is being attacked and what regular traffic levels look like. Therefore, we decide to collect anonymous statistics on the rest of the traffic that is filtered out at the start of the analysis since those were originally discarded in the Dissector. The statistics collected are shown in Table 4.2, the source and destination ports, as well as the protocol parameters, are collected as the distribution of the most common values found in the traffic. This means that we can identify whether a website or a game server might be hosted by the victim, but we cannot infer anything more than that because we do not save their IP address or the packet data.

4.1.5. Fingerprint Anonymization

The attack traces might contain regular traffic and therefore personal data could be found in the packets. In addition, victims of the attack may not want to be identified. To avoid revealing this information, we anonymize fingerprint data in different ways. First, the victim IP address is removed and only a generic description of the

Parameter	Description
Total Packets	Total number of packets.
Total MB	Total number of megabytes.
Average bps	Average number of bits per second.
Average pps	Average number of packets per second.
Average Bpp	Average number of bytes per packet.
Peak bps	Peak strength of bits per second.
Peak pps	Peak strength of packets per second.
Peak Bpp	Peak strength of bytes per packet.
Source Ports	Distribution of source ports found in the traffic.
Destination Ports	Distribution of destination ports found in the traffic.
Protocols	Distribution of protocols found in the traffic.

Table 4.2: Information summarized for the normal traffic characteristics.

Parameter	Description
Key	MD5 of the fingerprint including attack vectors.
Tags	List of keywords identifying the main characteristics of the attack.
Target	IP address or subnet victim of the attack.
Start Timestamp	Timestamp of the first received attack packet.
End Timestamp	Timestamp of the last received attack packet.
Duration	Difference between the end and start of the attack in seconds.
Total Packets	Total number of packets received by the victim.
Total MB	Total number of megabytes received by the victim.
Total IP Addresses	Total number of distinct IP addresses involved in the attack.
Average bps	Average number of bits per second.
Average pps	Average number of packets per second.
Average Bpp	Average number of bytes per packet.
Peak bps	Peak strength of bits per second.
Peak pps	Peak strength of packets per second.
Peak Bpp	Peak strength of bytes per packet.
Attack Vectors	List of attack vectors used during the attack.

Table 4.3: Information summarized in a fingerprint.

entity is kept for the analysis. Regular traffic is completely filtered out during the analysis except for average statistics as described in subsection 4.1.4. This way the output fingerprint will not contain any identifiable information that can link to the parties involved in the attack. Although we could anonymize the sources as well, we decided not to do that because we would like to analyze how the attack originates in detail. Therefore, we need detailed information on their traffic shapes, IP addresses and corresponding AS numbers. However, we already obtain the corresponding ASes for each source IP address, and thus it would be possible to anonymize them, removing the original IP addresses. Alternatively, we could also encrypt each IP address using the Crypto-PAn algorithm [39].

4.1.6. Summary Fingerprint

The generated fingerprint includes the start and end timestamps, the total number of packets, and bytes. Further, the average and peak statistics are calculated for the entire attack summing up the contributions of each attack vector. To uniquely identify the fingerprint, we compute the hash value obtained from the concatenation of the list of attack vectors with the overall characteristics. Table 4.3 and Table 4.4 specify every parameter that is extracted and saved in the fingerprint respectively for the overall statistics and the ones specific to each attack vector.

Parameter	Description	PCAP Only
Service	Detected application or protocol based on the source port.	✗
Protocol	Transport layer protocol.	✗
Start Timestamp	Timestamp of the first received attack packet.	✗
Duration	Duration of this specific attack vector in seconds.	✗
Fraction of Attack	Percentage of the total attack traffic.	✗
Source Port	Distribution of source ports used.	✗
Destination Ports	Distribution of destination ports.	✗
Total Packets	Number of packets belonging to this specific attack vector.	✗
Total MB	Total megabytes sent using this specific attack vector.	✗
Average Values	Average values for bps, pps, and Bpp.	✗
Peak Values	Peak values for bps, pps, and Bpp.	✗
Source IP Addresses	List of IP addresses that sent at least 1 packet to the victim.	✗
Source Statistics	List of traffic statistics grouped by IP address.	✗
Source ASes	Mapping of the list of IP addresses to their corresponding AS.	✗
ASes Statistics	List of traffic statistics grouped by AS number.	✗
TCP Flags	TCP flags found in at least 10% of the packets.	✗
Ethernet Type	Distribution of the protocols encapsulated in the ethernet frames.	✓
ICMP Parameters	Distribution of the ICMP types.	✓
IP Layer Parameters	Distribution of the IP TTLs and fragmentation offsets.	✓
DNS Parameters	Distribution of the DNS query names and types.	✓
HTTP Parameters	Distribution of the HTTP URIs, methods, and user agents.	✓
NTP Parameters	Distribution of the NTP request codes.	✓

Table 4.4: Information summarized for each attack vector.

4.2. Example Attack Fingerprint

To provide a better overview of what happens during the trace file analysis, we explore a complete example in detail by showing how the different parameters are extracted. In Table 4.5, a simplified example of a PCAP file containing 15 packets is shown. The first step of the analysis is to determine the target, which in this case can be found to be the IP address 42.42.42.42. Since more than 50% of the traffic has such a destination, we consider only the traffic received by 42.42.42.42 for the subsequent steps. Next, we extract the individual attack vectors. Grouping the attack traffic by protocol and payload we find two attack vectors, namely DNS and NTP. Therefore, we further split the analysis into two different attack vectors to be considered separately.

We can then collect the statistics for each vector such as the total number of packets and megabytes. In this case, we have 13 packets for a total size of 92 megabytes. Moreover, we calculate their corresponding fraction of the total attack, that is, the number of packets found in the attack vector divided by the total number of packets. In this case, DNS represents 53.8% of the traffic while NTP represents the other 46.2%. The average and peak statistics are also calculated, respectively, by considering only the DNS or NTP traffic. For DNS, the average bits per second is 30.75Mbps while the peak bits per second goes up to 85.15Mbps. For NTP, the average bits per second is 49.49Mbps and the peak bits per second is 171.10Mbps. Further statistics are collected for each IP address involved in the attack. Both the IP addresses and their respective AS numbers are calculated based on the timestamp of the attack. Here, the attack was launched at midnight on January 1, 2023; therefore, the routeviews snapshot corresponding to that day is retrieved from the CAIDA dataset. Moreover, number of packets, megabytes, and average bps as well as pps are computed for each IP address. In the case of DNS, the IP address 8.8.8.8 sent 2 packets with an average bits per second of 6.37Mbps. The computed statistics are grouped by AS number, and the total number of IP addresses per AS is also saved. From the PCAP file, we can derive the DNS query types and names used by the attacker, thus we can compute their distribution. In this case, 57.1% of the attack used an ANY DNS query requesting the domain ddostheinter.net while the remaining 42.9% queried the name servers for the root server. Similarly, we output the distribution of the request codes used in the NTP attack. Since every packet was generated in the same way, the distribution of the request code is 100% for code 42, which corresponds to the monlist command.

Additionally, the statistics for the entire attack are calculated as was done for each attack vector, obtaining an average bps strength of 55.1Mbps and a peak of 171.10Mbps. The timestamps for the start and end of the attack are computed excluding the traffic that is not considered attack traffic. Therefore, the attack starts at

#	Source	Destination	Timestamp	KB	Protocol	Payload
1	216.239.35.4:123	42.42.42.42:80	00:00:01	10543	UDP	NTP monlist
2	94.198.159.10:123	42.42.42.42:80	00:00:01	10344	UDP	NTP monlist
3	1.1.1.1:53	42.42.42.42:80	00:00:02	6555	UDP	DNS NS root-servers.net
4	8.8.8.8:53	42.42.42.42:80	00:00:03	3550	UDP	DNS ANY ddostheinter.net
5	216.239.35.0:123	42.42.42.42:80	00:00:04	9877	UDP	NTP monlist
6	8.8.4.4:53	42.42.42.42:80	00:00:05	3444	UDP	DNS ANY ddostheinter.net
7	42.42.42.42:80	7.7.7.7:34233	00:00:05	2330	TCP	HTTP 200 OK
8	216.239.35.4:123	42.42.42.42:80	00:00:06	9034	UDP	NTP monlist
9	1.1.1.1:53	42.42.42.42:80	00:00:07	6936	UDP	DNS NS root-servers.net
10	8.8.4.4:53	42.42.42.42:80	00:00:07	3459	UDP	DNS ANY ddostheinter.net
11	8.8.8.8:53	42.42.42.42:80	00:00:08	3453	UDP	DNS ANY ddostheinter.net
12	94.198.159.10:123	42.42.42.42:80	00:00:09	9934	UDP	NTP monlist
13	216.239.35.0:123	42.42.42.42:80	00:00:10	10341	UDP	NTP monlist
14	1.1.1.1:53	42.42.42.42:80	00:00:10	6700	UDP	DNS NS root-servers.net
15	42.42.42.42:80	7.7.7.7:34233	00:00:11	1050	TCP	HTTP 404 Not Found

Table 4.5: Example of a simplified PCAP file representing a DDoS attack.

00:00:01 and ends at 00:00:10, packet 15 is not considered because it is sent by the victim. To categorize the type of attack, we generate tags based on the attack vectors. In this case, the tags would correspond to the two attack vectors DNS amplification and NTP amplification, as well as the protocol used, which is UDP.

Lastly, the traffic that is filtered out at start is analyzed in the normal traffic category. Here, we collect the same average and peak statistics as done for the attack. However, we also include distributions on the protocol and ports found in the traffic. This is done to give an idea of what application might be hosted regularly on the victim's server. Since normal traffic is made up of TCP connections on port 80, we can say that it is most likely hosting a web server. The complete fingerprint corresponding to the attack can be found in JSON format in Appendix A.

4.3. Parallel DDoS Dissector

The Dissector loads the entire trace file into the memory and then runs the full analysis to generate the summary fingerprint. However, depending on the machine and the size of the attack, the memory requirements might not be enough. In particular, a one-minute attack with an average bandwidth of 100 Gbps will require more than 500 GB of RAM to be fully analyzed. To solve this issue, we decide to scale the Dissector software by splitting the PCAP file into smaller chunks, their individual fingerprints can then be merged once all of them are analyzed. While the splitting of the original trace file can cause inaccuracies in the final fingerprint due to packet fragmentation, we show through experiments that the error difference is negligible.

The splitting of the PCAP file into smaller ones is executed based on a defined maximum number of packets per file. First, we calculate the number of smaller files that we need by dividing the PCAP file size by a predefined maximum size that each file needs to satisfy. Second, using the total number of packets in the original PCAP file we get the maximum number of packets each chunk will contain, dividing the total packets by the number of chunks required and then rounding up the result. Then, using the editcap [40] utility we split it into the different files according to the calculated limit. The Dissector software is run on each file, and at the end a merging step occurs to combine all the output fingerprints into a single summary fingerprint.

We do not discuss how to split other file types because they usually contain less information than PCAP files, and therefore they are more likely to fit within the available memory requirements. In addition, NetFlow traces are generally not saved as a single file but they are exported over several records. This means that they do not require to be further split and the Dissector software can be run on each NetFlow output. Then, the

merging rules shown in the following section can be applied the same way as they would be for PCAP files.

4.3.1. Merging Rules

The merging works by applying a reduce operation that merges fingerprint pairs together until a single one is left. During the merging, the statistics are appropriately calculated depending on what value they represent. In case of summation values such as duration or total number of packets, the two values are summed together. For averages, the weighted average is calculated based on duration for pps and packets for Bpp. Lastly, the maximum value found for each peak is used, while the rest are discarded.

Next, we formally define how each possible merge can be calculated given a pair of values. For a summation pair, x and y, the output will simply be:

$$merge_{sum}(x, y) = x + y \tag{4.1}$$

In case of a maximization pair, the formula will be as follows:

$$merge_{max}(x, y) = \begin{cases} x, & \text{if } x > y. \\ y, & \text{otherwise.} \end{cases} \tag{4.2}$$

Similarly, a minimization pair can be defined as

$$merge_{min}(x, y) = \begin{cases} x, & \text{if } x < y. \\ y, & \text{otherwise.} \end{cases} \tag{4.3}$$

Finally, for averaging pairs, each value will have a corresponding weight:

$$merge_{avg}(x, y) = \frac{x \cdot x_w + y \cdot y_w}{x_w + y_w} \tag{4.4}$$

where x_w and y_w are respectively the weights for x and y. Some values are not single numbers but distributions of the different values that occurred in the trace with their corresponding percentage values, for example the destination ports of an attack vector. In case of distributions of values, the same operation used for averaging pairs is applied to the parameter values found in the distribution:

$$merge_{dist}(x, y) = \{merge_{avg}(x_v, y_v) : \forall v \in distvalues(x \cup y)\} \tag{4.5}$$

where x_v and y_v are the respective amounts of x and y for a given value, if one distribution does not contain the specific value, the amount will be 0.

Table 4.6 shows the respective merging formula used for each parameter found in the fingerprint, or the default value in the case where it might not be present. Some values cannot be calculated during the merge operation; therefore, the calculation is postponed and applied to the final fingerprint. In the case of the target, we assume that the same value is present in all fingerprints. Attack vectors present in both fingerprints are grouped based on their service, protocol, and source port. Then, the merge operations are applied according to the rules defined in Table 4.7. Similarly, the normal traffic statistics are merged using the corresponding formulas found in Table 4.8.

4.3.2. Merging Example

Table 4.9 shows 3 example fingerprints with the output result obtained when merging them together using the operations previously described. Here, we show how the merge operations are applied on each parameter. The algorithm merges the first 2 fingerprints together and then merges the corresponding output with the third one. For the first merge, the SUM operation is applied to the duration and number of packets, obtaining a duration of $60+40 = 100$ seconds and a total number of packets of $100+100 = 200$. Then, the average bps is calculated using the weighted average based on the duration, obtaining a value of

$$\frac{100 \cdot 60 + 200 \cdot 40}{60 + 40} = 140 \text{ bps}$$

The peak is calculated using the MAX operation, therefore the value for the first merge is 300 bps. Lastly, the DIST operation is applied to the DNS query parameters. In this case, the domain A.COM is only found in the first attack therefore the corresponding value in the second chunk is set to 0:

$$\frac{0.5 \cdot 100 + 0 \cdot 100}{100 + 100} = 0.25$$

Parameter	Merge strategy
Key	Recalculated after merge.
Target	Left value is taken.
Tags	Set union of the values.
Start Timestamp	MIN (4.3)
End Timestamp	MAX (4.2)
Duration	SUM (4.1)
Total Packets	SUM (4.1)
Total MB	SUM (4.1)
Total Flows	SUM (4.1)
Total IP Addresses	Recalculated after merge.
Average bps	AVG (4.4)
Average pps	AVG (4.4)
Average Bpp	AVG (4.4)
Peak bps	MAX (4.2)
Peak pps	MAX (4.2)
Peak Bpp	MAX (4.2)

Table 4.6: Merging strategy for each parameter found in the fingerprint.

For the domain B.COM, both values are present obtaining:

$$\frac{0.5 \cdot 100 + 1 \cdot 100}{100 + 100} = 0.75$$

We can repeat the same operations to merge the third chunk with the values previously obtained that can be found in the table as 1+2. The total duration and the number of packets, respectively, become $100 + 100 = 200$ seconds and $200 + 200 = 400$ packets. The peak bps does not change because the third chunk has a lower value, while the average bps can be calculated as

$$\frac{140 \cdot 100 + 150 \cdot 100}{100 + 100} = 145 \text{ bps}$$

As a last step, we can calculate the distribution of each DNS query, for the domain A.COM:

$$\frac{0.25 \cdot 200 + 0.75 \cdot 200}{200 + 200} = 0.5$$

For the domain B.COM:

$$\frac{0.75 \cdot 200 + 0 \cdot 200}{200 + 200} = 0.375$$

And, for the domain C.COM:

$$\frac{0 \cdot 200 + 0.25 \cdot 200}{200 + 200} = 0.125$$

The final output values are also found in the table row 1+2+3.

4.3.3. Split-and-Merge Pseudocode

Algorithm 1 provides the pseudocode to implement the split-and-merge algorithm using the previously defined building blocks. The parameters passed to the function are the path to the PCAP file (pcapFile) and the maximum amount of memory (maxMemory). In line 2, the capinfos [41] utility is used to retrieve the total number of packets present in the PCAP file. Subsequently, on line 3 the utility getFileSize is used to read the total size of the PCAP file in bytes. When calculating the maximum number of splits, the maximum amount of memory is halved because the Dissector software overhead memory usage is usually 1.5 times greater than the file size. While the number of splits could be optimized, using more memory did not result in an increase in runtime performance as shown in subsection 6.4.1. Then, on line 6 the editcap [40] software is used to split the PCAP files into equally sized traces containing the calculated maximum number of packets each. The Dissector is run on each file after the split, and the fingerprint returned is saved while the smaller PCAP chunk is removed from the disk on line 11. Lastly on line 15, fingerprint pairs are merged together using the previously described rules in subsection 4.3.1.

Parameter	Merging Strategy	Default value
Service	First value is used.	N/A
Protocol	First value is used.	N/A
Source Port	First value is used.	N/A
Fraction of Attack	Recalculated after merge.	N/A
Destination Ports	DIST (4.5)	random
Total Packets	SUM (4.1)	N/A
Total MB	SUM (4.1)	N/A
Start Timestamp	MIN (4.3)	N/A
Duration	SUM (4.1)	N/A
Source IP Addresses	Set union of values	Empty list
TCP Flags	DIST (4.5)	null
Ethernet Type	DIST (4.5)	null
ICMP Parameters	DIST (4.5)	null
IP Layer Parameters	DIST (4.5)	null
DNS Parameters	DIST (4.5)	null
HTTP Parameters	DIST (4.5)	null
NTP Parameters	DIST (4.5)	null

Table 4.7: Merging strategy for each parameter found in the attack vectors.

Parameter	Merging Strategy	Default value
Total Packets	SUM (4.1)	N/A
Total MB	SUM (4.1)	N/A
Total Flows	SUM (4.1)	N/A
Average bps	AVG (4.4)	N/A
Average pps	AVG (4.4)	N/A
Average Bpp	AVG (4.4)	N/A
Peak bps	MAX (4.2)	N/A
Peak pps	MAX (4.2)	N/A
Peak Bpp	MAX (4.2)	N/A
Source Port	DIST (4.5)	random
Destination Ports	DIST (4.5)	random
Protocol	DIST (4.5)	random

Table 4.8: Merging strategy for each parameter found in the normal traffic statistics.

4.4. Spoof Detection

To supplement our analysis, we develop techniques and heuristics to detect the likelihood of spoofing in packets received during a DDoS attack. Although there is no guarantee that the spoofing did not occur, we can determine whether the spoofing did occur with high probability. To do that, we look at the time-to-live (TTL) values present in the IP headers sent by the attackers and analyze the corresponding distributions. In particular, the packets sent by the same machine will follow very similar paths to reach the destination, therefore their TTL values will generally vary by at most a couple hops. We can then group the packets received from a sender's IP address and determine the difference between the maximum and minimum TTL values found in the received packets. In our case, a likely spoofed IP packet will present a difference larger than 4. In addition, we can analyze the overall distribution of TTL values across all packets and identify outliers that are unlikely to occur. For example, low values such as 5 are not common because the default value is usually 64 and that would mean that the packet had to visit 59 hops before reaching the target server. Additionally, looking at the distribution, we can notice the different operating systems that make up the network of amplifiers, since the different operating systems have different default TTL values. While we cannot figure out the exact operating system, we can group them in 3 classes, namely Linux (Default = 64), Windows (Default = 128), and Solaris (Default = 254) [42]. Furthermore, IP addresses belonging to the same AS or network are generally physically located in the same datacenter, and therefore the path taken by packets to reach the same destination will be similar. Therefore, we can further group IP addresses by their /24 subnet and spot inconsistencies in their TTL

Chunk	Duration	Packets	Avg bps	Peak bps	DNS query
1	60	100	100	125	A.COM (50%), B.COM (50%)
2	40	100	200	300	B.COM (100%)
3	100	200	150	200	A.COM (75%), C.COM (25%)
1+2	100	200	140	300	A.COM (25%), B.COM (75%)
1+2+3	200	400	145	300	A.COM (50%), B.COM (37.5%), C.COM (12.5%)

Table 4.9: Example fingerprints with the respective merging results.

Algorithm 1 Split PCAP files, run DDoS Dissector, and merge the results together.

function SPLITANDMERGE(pcapFile, maxMemory)
  $totalPackets \leftarrow capinfos(pcapFile).getTotalPackets()$
  $fileSize \leftarrow getFileSize(pcapFile)$
  $numberOfSplits \leftarrow \lceil \frac{fileSize}{maxMemory/2} \rceil$
  $packetsPerSplit \leftarrow \lceil \frac{totalPackets}{numberOfSplits} \rceil$
  $filesAfterSplit \leftarrow editcap(pcapFile, packetsPerSplit)$
  $fingerprints \leftarrow \{\}$
  for all $file \in filesAfterSplit$ **do**
    $fingerprint \leftarrow ddosDissector(file)$
    $fingerprints \leftarrow fingerprints + fingerprint$
    $delete(file)$
  $finalFingerprint \leftarrow fingerprints[0]$
  $i \leftarrow 1$
  while $i < len(fingerprints)$ **do**
    $finalFingeprint \leftarrow merge(finalFingerprint, fingerprints[i])$
    $i \leftarrow i + 1$
  return $finalFingeprint$

values. However, since the machines might have different default values, we should pay attention to their relative difference from the possible starting TTL rather than the absolute one.

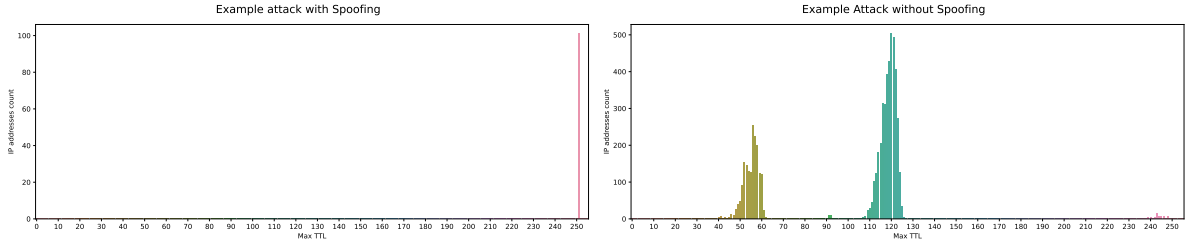

Figure 4.1: Example maximum TTL distributions of two attacks. Left: Spoofed, Right: Not spoofed.

Figure 4.1 shows two example distributions for a likely spoofed attack and one that is unlikely to be. The x-axis represents the maximum TTL value received, and the y-axis displays how many unique IP addresses had this value. In the spoofing case, we notice that all 100 IP addresses had the same TTL value of 251 which is likely to happen when the same server is sending all the packets. On the other hand, the non-spoofed example shows a different distribution with two peaks at 55 and 120. As previously mentioned, these peaks occur because the default TTL value is dependent on the software implementation, and we can attribute the 55 and 120 TTL values, respectively, to Linux and Windows implementations. Moreover, we can also see a smaller peak at around 240 which can indicate that a small percentage of machines in the attack were running on a Solaris system. To test our assumptions on the bimodal distribution of TTL values, we collect 1 day of DNS query traffic sent to the .NL authoritative servers at SIDN [43]. After filtering out invalid DNS packets by checking whether the response code is 0, we are left with 3.8 billion queries over 3.9 million IP addresses. The maximum TTL distribution plot is shown in Figure 4.2. We notice a similar distribution with peaks at 55, 120, and a smaller spike at 240, which confirms our assumptions on the multimodal distribution based on the default TTL values.

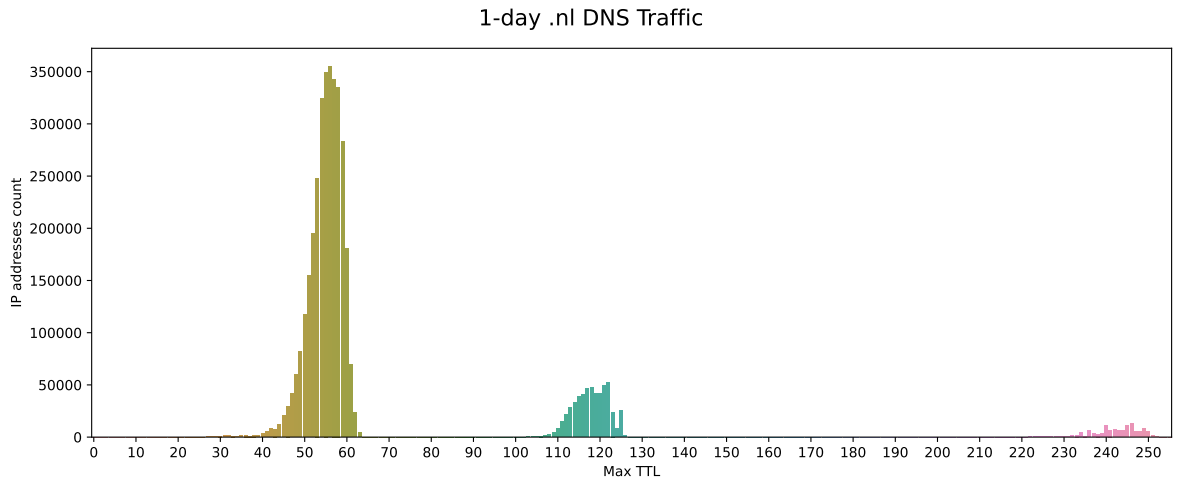

Figure 4.2: Maximum TTL distribution from 1-day of DNS traffic to the .nl authoritative servers.

The spoof detection software is open source and available at the following URL: https://github.com/paolokazemi/Spoof-Detector.

5 Dataset

In this section, we list the different datasets we obtained for our analysis, highlighting their main characteristics.

5.1. Individual Attacks

To validate the developed methodology, we use several datasets containing different types of DDoS attacks. The datasets analyzed are as follows, the characteristics of their attack traces are listed in Table 5.1.

- Traces collected from 9 different Booters containing DNS and CHARGEN amplification attacks [26], downloaded from the Simpleweb Data Repository [44].
- The 2007 DDoS attack towards CAIDA [45].
- Several extracted DoS and DDoS traces provided by the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC) [46].
- Malware DDoS attack [47] and TCP SYN flood [48] extracted from the 2009 DARPA Scalable Network Monitoring (SNM) Program Traffic.
- NTP traffic containing NTP Amplification attacks [49].
- A staged DNS amplification attack [50].

5.1.1. 9 Booters

The 9 Booters dataset was collected by Santanna et al. [26] launching attacks towards their own infrastructure. The recorded PCAP files contain the packets received by their network cards, excluding the ones that were dropped. In particular, their equipment was overloaded during some attacks and therefore some packets were lost. While the original research compensates for lost packets using an algorithm, we analyze only the recorded packets because we do not want to generate artificial packet data that was not actually contained in the received attack. The attack sizes vary between 292 MB and 47 GB, respectively, for B5 and B4. Booters 1 through 7 are DNS amplification attacks and the last two use CHARGEN amplification. The IP addresses were anonymized before making them public therefore we cannot analyze the sources of the attack in depth.

5.1.2. CAIDA 2007 DDoS Attack

The CAIDA 2007 DDoS attack lasted for about an hour on the 4th of August 2007. The attack strength was 80 Mbps, which corresponds to a 21 GB PCAP file where the packet data is stripped. The IP addresses contained in the trace were anonymized with the CryptoPAn algorithm [39]. Although the trace corresponds to a DDoS attack, not all packets are malicious and normal traffic can be found even if in small percentages.

5.1.3. CSE & CIC DDoS Traces

The CSE & CIC DDoS traces are part of a larger dataset generated to train Intrusion and Detection Systems (IDS). The dataset contains several cyberattacks such as brute force attempts, port scans, and DoS attacks. For our analysis, we discard all PCAP traces except the ones corresponding to a DoS attack. We find 6 attacks ranging from less than 1 MB up to 17 GB in size. The attacks were simulated, therefore the IP addresses are local IP addresses, we consider them as if they were anonymized since we cannot extract any meaningful information.

Dataset	Attack	Size (MB)	Anonymized	Staged
9 Booters	B1	11648	Yes	No
9 Booters	B2	7679	Yes	No
9 Booters	B3	11916	Yes	No
9 Booters	B4	47695	Yes	No
9 Booters	B5	292	Yes	No
9 Booters	B6	10759	Yes	No
9 Booters	B7	17608	Yes	No
9 Booters	B8	6675	Yes	No
9 Booters	B9	15322	Yes	No
CAIDA 2007	Single Attack	21087	Yes	No
CSE & CIC DDoS Traces	Hulk	1	Yes	Yes
CSE & CIC DDoS Traces	SlowHTTPTest	4113	Yes	Yes
CSE & CIC DDoS Traces	Slowloris	181	Yes	Yes
CSE & CIC DDoS Traces	LOIC	8097	Yes	Yes
CSE & CIC DDoS Traces	HOIC	2026	Yes	Yes
CSE & CIC DDoS Traces	LOIC-UDP	17429	Yes	Yes
DARPA 2009	Malware DDoS	975	Yes	Yes
DARPA 2009	TCP SYN Flood	2929	Yes	Yes
NTP Amplification	Single Attack	439	Yes	No
Staged DNS Amplification	Single Attack	3187	Yes	Yes

Table 5.1: Overview of the analyzed datasets and their attacks.

5.1.4. DARPA 2009

The DARPA 2009 dataset contains synthetic traffic that represents the packets received between a /16 subnet and the Internet. The entire traffic is larger than 6 TB, however, we extract the only 2 DDoS attacks that can be found. In this case, there is a 3 GB TCP SYN flood attack and 1 GB attack corresponding to a DDoS attack generated from a malware that compromised local hosts. Both attack traces contain a small percentage of traffic that is not related to the attack. All IP addresses correspond to private ranges (172.28.0.0/16) and therefore they cannot be further analyzed.

5.1.5. NTP Amplification

The dataset corresponds to 3 months of NTP traffic between two ISPs. The traces are saved as Argus [36] flows. Most of the traffic is regular NTP data generated from ordinary operations of an ISP. However, there are several instances of NTP amplification attacks launched using the monlist command. The IP addresses were anonymized with a prefix-preserving algorithm such as CryptoPAn [39].

5.1.6. Staged DNS Amplification Attack

The attack was staged in a controlled environment between two sites, and it consisted of 6 DNS resolvers, 1 victim, and 1 attacker. The original IP addresses are anonymized with a prefix-preserving algorithm. The network trace is a 3 GB ERF formatted file.

5.2. NaWas Dataset

The last dataset that we analyze is traffic data collected from attacks that were sent to a DDoS scrubbing service. Figure 5.1 shows how the traffic collection is executed. The scrubber service is tasked with receiving all traffic and redirecting only regular traffic to the victim of the attack. The attack traffic is generally blackholed, however, for the purpose of collecting traffic data, samples of the traffic are exported and saved. In this case, the sampling rate is between 1:8129 and 1:64000 depending on the strength of the attack and the switch handling the data. The network trace is saved as an sflow file that is processed through the Dissector. The corresponding fingerprints for each sflow file are used in our analysis. We obtained two datasets from the Dutch National Scrubbing Service (NaWas). The first one contains over 1400 sflow files containing attack traffic sampled between February 2022 and June 2022. The second dataset does not contain any network data but it contains a report for every attack launched between September 2022 and February 2023. The report shows the peak bps and pps, source and

destination ports, the duration of the attack, the list of attack vectors, and optionally the TCP flags found in the packet headers.

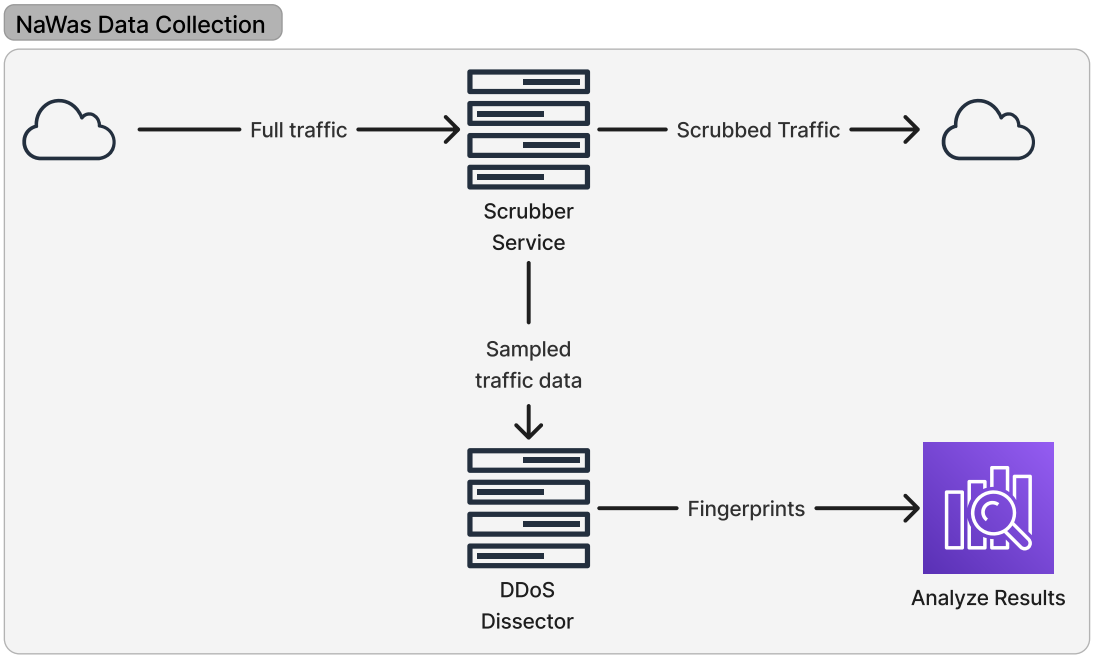

Figure 5.1: NaWas traffic data collection from the scrubber service.

6 Results

In this section, we show the results obtained when running the Dissector and spoof detection software on the previously listed datasets. First, we look at the features extracted from the trace files and analyze the limitations in the quality of the results of the split-and-merge algorithm. Next, we highlight the TTL distributions of some selected attacks obtained from the spoof detection analysis. Then, we benchmark our software by calculating the processing speed and the error introduced when using the split-and-merge algorithm. Lastly, we present our findings on the DDoS landscape for the years 2022 and 2023, obtained from the NaWas dataset.

6.1. Extracted Features

The following results were obtained by running the split-and-merge algorithm on each trace file with the maximum available memory set to 1 GB. Most of the files are less than 4 GB in size, therefore we decide to use 1 GB to ensure that there would be at least 2 chunks that would need to be merged. Moreover, some files contained a significant amount of non-attack traffic directed towards other servers. In those cases, we manually specify the target of the attack, otherwise the inference algorithm would not be able to properly detect the victim. This is a known limitation, as we assume that normal traffic is mostly filtered out before the feature extraction. The victims of the attack are only used during the features' extraction to collect the necessary insights. Then, they are completely removed, and we do not show any results related to specific attacked entities.

6.1.1. 9 Booters

A summary of the main characteristics extracted from the Booter traces can be found in Table 6.1. As shown in the original paper [26], Booters 1 to 7 launch a DNS amplification attack, while the last two are CHARGEN amplification attacks. In some cases, our average Gbps results differ from the original paper because we are extracting the features from the PCAP trace file. Instead, the paper applies an algorithm to compensate for lost packets that could not be collected because their network interface dropped them. The Dissector assumes all packets are available, therefore the strength analysis reports the lower number in case of packet loss. Since the trace file contains only attack traffic packets, we confirm the packet rate using the capinfos [41] utility. In all cases, the packet rate reported by the Dissector is the same as the one reported by capinfos. For the DNS amplification attacks, we find the same query names as in the paper with their respective types. Moreover, we discover an additional domain name SEMA.CZ for Booter 6 which was not mentioned in the paper. The domain SEMA.CZ represented 37.7% of the attack traffic while the rest was attributed to ANONSC.COM. From the DNS queries, we notice different Booters reusing the same domain name, indicating that they might share infrastructure such as the DDoS attack scripts or simply resell from the same provider. The DNS query for the first 3 Booters is <Root> and corresponds to the root name servers [51].

6.1.2. CAIDA 2007 DDoS Attack

Table 6.2 shows the main characteristics that we extracted from the CAIDA 2007 DDoS attack. The attack is an ICMP flood of 1 hour with an average bps of 52.77 Mbps and a peak of 83.07 Mbps. The obtained peak value is also confirmed by the dataset source [45]. Moreover, the packets sent per second were on average 111k, and they peaked at 175k. The main attack vector was an ICMP flood where both the source and destination ports were 0. Port 0 is a reserved port according to the standard [52], therefore, the attack could be detected and

Booter	Avg / Peak Gbps	Avg / Peak pps	Duration (s)	Sources	DNS query
B1	0.301 / 0.737	39970 / 101574	295	4177	ANY <Root>
B2	0.222 / 0.380	31753 / 53660	280	65	ANY <Root>
B3	0.356 / 0.598	47849 / 81325	278	43	ANY <Root>
B4	0.634 / 1.110	58479 / 102207	598	2709	A A.PACKETDEVIL.COM
B5	0.0056 / 0.0076	5545 / 7483	368	7393	A DDOSTHEINTER.NET
B6	0.618 / 1.595	68004 / 174537	138	6972	ANY ANONSC.COM & SEMA.CZ
B7	0.282 / 0.387	32385 / 44405	531	5835	ANY ANONSC.COM
B8	0.324 / 1.034	33471 / 106762	169	251	N/A
B9	0.853 / 2.952	88293 / 302898	155	3507	N/A

Table 6.1: Dissector results obtained from the Booters traces.

canceled by ignoring the traffic intended for reserved ports. Moreover, all ICMP packets were of type 8 which corresponds to the Echo request message. We also find other attack vectors such as HTTP and TCP, however, these do not correspond to attack traffic. In fact, the percentage of packets attributed to the HTTP attack vector compared to the total attack is less than 1%. Therefore, we can ignore them as false positives, and we note that this is mostly due to the split algorithm creating chunks of traffic containing significant normal traffic. In particular, these chunks can be found at the beginning of the file when the attack is not at its maximum strength. A possible solution to this problem would be to filter them out after the merge algorithm has completed.

Attack Vectors	ICMP:0 (92%), TCP (7%), HTTP (1%)
Average / Peak Mbps	52.77 / 83.07
Average / Peak pps	111356 / 175330
Average / Peak Bpp	59 / 884
Duration	3954
Sources	9298

Table 6.2: Dissector results obtained from the CAIDA 2007 DDoS attack.

6.1.3. CSE & CIC DDoS Traces

Table 6.3 displays the attacks extracted from the CSE & CIC DDoS dataset. The attacks were directed to a specific application, therefore we list the attack vectors with the corresponding port number. In particular, we consider the combination of protocol and destination port as a single attack vector. In the case of Hulk, the attack was sent to port 22 in 66.7% of the cases and the rest of the packets contained a randomized selection of ports. All attacks except for Hulk were directed towards port 80 which indicates a running HTTP server. In the TCP flags column, we list the most used combinations of flags with the percentage of packets containing such combination. For example in the Slowloris attack, 37.7% of packets contained both the ACK and PSH flags, while only 32.9% used the ACK flag. In case the percentages do not add up to 100%, the remaining packets used a random combination of flags. In some cases, the average and peak values differ significantly such as for the Hulk or Slowloris attacks. This is due to normal traffic being mixed with the actual attack.

6.1.4. DARPA 2009 Malware DDoS

The DARPA 2009 malware DDoS was a TCP flood sending 160 packets per second at 76.80 kbps, the attack lasted for 131 seconds and its main characteristics can be found in Table 6.4. In this case, the trace contained several packets from other sources, and we needed to manually specify the target. While the source ports were randomized, the attack was sent to a single destination port, namely 499. This is also confirmed by the dataset source [47]. Furthermore, the TCP flags found in the packet headers were evenly split between the SYN and RST flags.

6.1.5. DARPA 2009 TCP SYN Flood

The second attack from the DARPA 2009 dataset is again a TCP flood, and a target was also manually specified in this case. Table 6.5 shows its main characteristics, the average bps was 0.43 Mbps and the peak 1.13 Mbps. The average packets per second was 913 with a peak of 2358, the duration of the attack was 284 seconds. In

Attack	Vectors	Avg / Peak kbps	Avg / Peak pps	TCP Flags
Slowloris	TCP:80 (98.8%)	10.672 / 3997	7 / 3072	ACK+PSH (37.7%) ACK (32.9%)
SlowHTTPTest	TCP:80 (98.9%)	548.873 / 16193	488 / 14164	ACK+PSH (20.9%) ACK (40.5%) SYN (4.7%)
Hulk	TCP:22 (66.7%)	0.165 / 84.576	0 / 53	ACK+PSH (27.8%) ACK (22.2%) SYN (32.6%)
LOIC	UDP:80 (98.4%)	3317 / 59860	5617 / 101116	N/A
LOIC-UDP	UDP:80 (100%)	14161 / 61753	23911 / 104314	N/A
HOIC	TCP:80 (100%)	394 / 1937	422 / 2085	ACK+PSH (20%) ACK (40%) CWR+ECE+SYN (32.6%)

Table 6.3: Dissector results obtained from the CSE & CIC DDoS traces.

Attack Vectors	TCP:499 (100%)
Average / Peak Mbps	0.076 / 0.076
Average / Peak pps	160 / 160
Average / Peak Bpp	60 / 60
Duration	131
Sources	8
TCP Flags	SYN (50%), RST (50%)

Table 6.4: Dissector results obtained from the DARPA 2009 Malware DDoS attack.

this case, the SYN TCP flag was found in 99.5% of the packets; therefore, we can classify it as a TCP SYN flood.

Attack Vectors	TCP (100%)
Average / Peak Mbps	0.438 / 1.131
Average / Peak pps	913 / 2358
Average / Peak Bpp	60 / 60
Duration	284
Sources	104
TCP Flags	SYN (99.5%)

Table 6.5: Dissector results obtained from the DARPA 2009 TCP SYN flood.

6.1.6. NTP Amplification

Table 6.6 summarizes the main characteristics obtained from the NTP amplification attack. In this case, the trace file uses the Argus format and contains both normal and attack traffic. In particular, the entire traffic data contain only NTP packets where some packets correspond to NTP monlist commands used in NTP reflection attacks. Due to the Dissector limitations, both normal and attack traffic are considered malicious, and the resulting fingerprint reflects that. The average rate is 1.538 Mbps with an average pps of 2046. However, these average values take into account both normal traffic and attack traffic. To get the actual attack strength we can consider the peak values, which in this case are 12.905 Mbps and 17k packets per second. Moreover, the detected attack vectors are both NTP and UDP, particularly the latter corresponds to 73%. Although we know that the UDP traffic contains only NTP packets, the Dissector cannot infer that because the Argus format does not store the packet data and the protocol inference uses the source port information. In this case, the source ports do not correspond to the standard TCP port (123) and therefore we can only classify it as UDP traffic.

6.1.7. Staged DNS Amplification Attack

The staged DNS amplification attack is correctly classified by the Dissector reporting an average bps of 10.87 Mbps and a peak of 11.85 Mbps. We display its main characteristics in Table 6.7. The victim consistently

Attack Vectors	UDP:123 (73%), NTP:123 (27%)
Average / Peak Mbps	1.538 / 12.905
Average / Peak pps	2046 / 17161
Average / Peak Bpp	94 / 94
Duration	589
Sources	242191

Table 6.6: Dissector results obtained from the NTP amplification attack.

received 1200 packets per second for the entire 20-minute duration of the attack. The attack was recorded to an ERF file, therefore we could not extract the DNS query names and types used. However, we noticed that there was not a single destination port attacked, but six. The packets were evenly sent to each port, the attacked ports were 37923, 34166, 42587, 45612, 54897, and 42321. We could not find well-known applications that use these ports. However, this peculiarity could be used to automatically detect the attack as part of a detection system. In particular, it seems like the attacker scripts deterministically chose 6 ports when crafting the packets for the amplifiers and did not randomize them. Or, they were running 6 scripts concurrently where each script uses the same port for all the packets sent.

Attack Vectors	DNS:37923 (16.6%), DNS:34166 (16.6%), DNS:42321 (16.6%) DNS:42587 (16.6%), DNS:45612 (16.6%), DNS:54897 (16.6%) NTP (0.1%)
Average / Peak Mbps	10.873 / 11.851
Average / Peak pps	1194 / 1203
Average / Peak Bpp	1129 / 1234
Duration	1011
Sources	7

Table 6.7: Dissector results obtained from the staged DNS amplification attack.

6.2. Source Statistics

From the signatures, we can extract the source statistics that indicate the number of packets and bytes sent by each IP address. For this analysis, we exclude traces that are synthetically generated because the source servers would not be representative of a real infected machine. Moreover, we do not consider traces containing a significant amount of non-attack traffic since the Dissector is not able to effectively distinguish that from the real attack. Figure 6.1 shows the cumulative distribution of the number of packets sent by each source for the 9 Booters and the CAIDA attack.

We notice that all sources send a similar amount of packets, with exceptions for Booter 2, 3 and the CAIDA attack. In the latter case, the distribution is closer to an exponential curve where 40% of the sources sent more than 50k packets and in some cases up to 100k packets. In case of Booter 2, we see two distinct upticks at 110k packets which corresponds to 50% of the cases and a second one at 130k accounting for another 40% of all sources. Lastly, 40% of the Booter 3 amplifiers send less than 225k packets, and the remaining 60% send up to 275k packets each.

The difference in number of packets sent could be attributed to servers being more powerful and therefore able to process a larger amount of information. Furthermore, it could be possible that the amplifier detected the attack and stopped sending response packets to the victim, resulting in a lower number of packets sent.

6.3. Spoof Detection

For the spoof detection analysis, we plot the maximum TTL distribution for each IP address involved in the attack. The x-axis represents the maximum TTL value and the y-axis shows how many IP addresses had such a value. We list six different types of attack and their distributions. Figure 6.2, Figure 6.3, and Figure 6.4 respectively show the distributions of a DNS, NTP, and CHARGEN amplification attack. Figure 6.5 displays the distribution of a slowloris attack. Last, Figure 6.6 and Figure 6.7 depict an ICMP and a TCP SYN flood.

We notice that the amplification attacks have similar distributions because they are usually not spoofed. In fact, they present a bimodal distribution with peaks at around 55 and 120. Although they are not spoofed, we

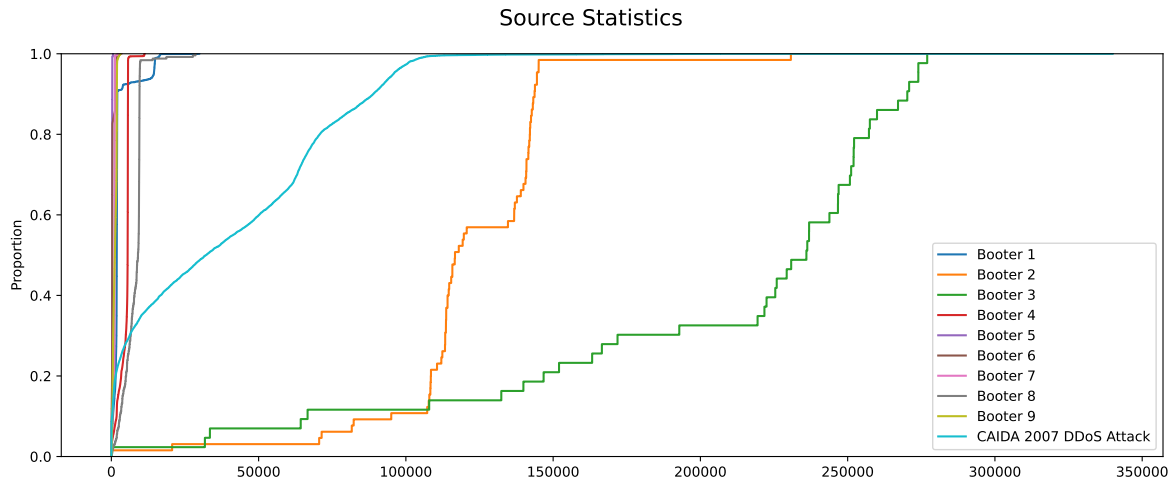

Figure 6.1: Cumulative distribution of the number of packets sent by each source for the 9 Booters and the CAIDA 2007 DDoS attack.

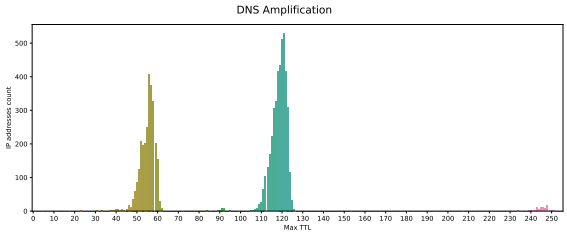

Figure 6.2: TTL distribution of a DNS amplification attack.

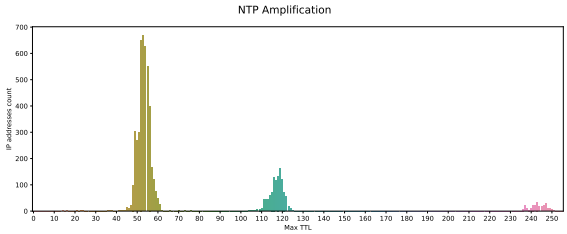

Figure 6.3: TTL distribution of an NTP amplification attack.

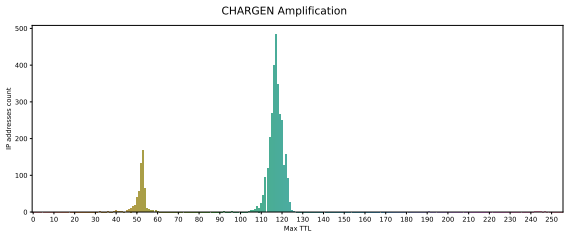

Figure 6.4: TTL distribution of a CHARGEN amplification attack.

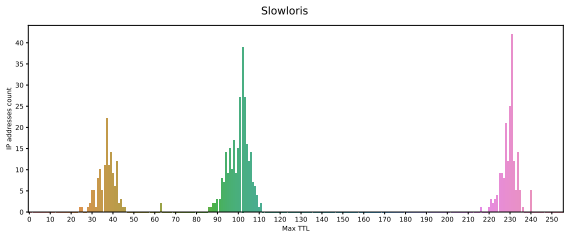

Figure 6.5: TTL distribution of a slowloris attack.

can still infer the operating systems used by considering the well-known default TTL values [42]. On the one hand, the distribution is skewed towards Linux machines for the NTP amplification attack. On the other hand, the opposite is seen from the CHARGEN amplification attack. This provides an insight into the amplifiers that are used by the attackers. In particular, there seem to be more exploitable Linux NTP servers than Windows. For CHARGEN, Windows servers are more prevalent, and for DNS the difference is less accentuated, but it points towards Windows machines too. In the slowloris attack, the distribution is trimodal indicating that a significant number of machines were using Solaris.

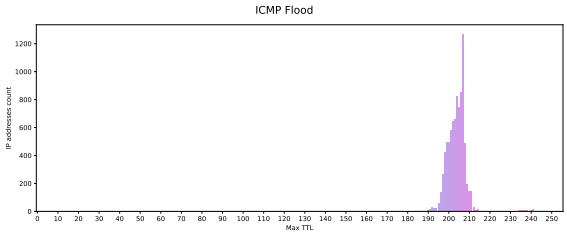

Figure 6.6: TTL distribution of an ICMP flood attack.

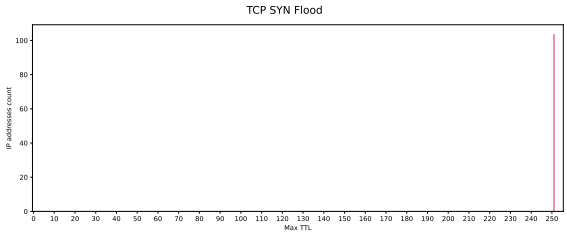

Figure 6.7: TTL distribution of a TCP SYN flood attack.

The ICMP flood does not show a bimodal distribution and instead the TTL values are mostly in the 200-210 range. This is an indication that the sources had a similar configuration, for example, a Botnet affecting a specific device. Due to the wide range of TTL values, the attack is most likely not spoofed. Lastly, the TCP SYN flood had 100 different IP addresses with the same TTL value of 251. This signals that the attack was launched from the same machine and the packets were spoofed.

6.4. Benchmarks

We benchmark the split-and-merge algorithm against the 9 Booter traces by Santanna et al. [26]. We first run the Dissector software on the original file and then using our algorithm with a maximum allowed file size of 5 GB. The tests are executed on a virtual server using 4 cores on top of an Intel(R) Xeon(R) Gold 6148 CPU with 64 GB of available RAM and 256 GB of storage. Therefore, we were unable to execute the original analysis on Booter 4 because its corresponding file size is 47 GB, and when loaded into memory it consumes more than 64 GB. The results for all other Booters can be found in Tables 6.8, 6.9, 6.10, and 6.11. Each table displays a different parameter, such as the duration obtained, the total number of packets and the peak bps, as well as the percentage difference between the original and the parallel version.

The duration error is zero because it is calculated by taking the difference between the maximum and minimum timestamps; since we do not lose the first and last packets during the split, this value will not change. We see that in some cases, such as Booter 6, the number of packets and peak value differ. However, we note that this difference is generally below 0.5% which is an acceptable error for our analysis. Although packets are not lost, we notice that Booter 6 is missing 4 source addresses after the split. We can attribute this to the way the split is done, which is based on the number of packets, and in the case of fragmented packets, they might end up in different chunks, thus corrupting the information contained. Furthermore, the corrupted packets can be seen by the table reporting the total number of packets, where the error difference is close to zero in most cases. However, Booter 6 lost around 0.40% of the packets and that is most likely the reason why those 4 IP addresses are missing from the count. Similarly, the attack strength is preserved after the split both for the bits per second and the packet per second statistics. The two noticeable differences are in Booter 6 and 8, where Booter 6 difference can be explained by the fragmented packet loss. In the case of Booter 8, we note that while the average statistics report an error of 0.60%, the peak strength is exactly the same and for our analysis we particularly focus on the peak value rather than the average strength of the attack.

To validate our assumption about packet fragmentation, we check the percentage of fragmented packets found in each trace and display the results in Table 6.12. We note that for the first 2 Booters the fragmentation is below 20% and the respective error differences are below 0.02%. However, we do not find a strict correlation between the fragmentation percentage and the quality of results. In fact, Booter 8 and 9 contain more than 60% fragmented packets, but their results vary differently. Although Booter 8 error differences are larger than 0.5% in some cases, Booter 9 differences are consistently below 0.02%. Therefore, we cannot find any significant correlation between the packet fragmentation and the quality of results, although a high percentage of fragmented packets that are split between different chunks will result in calculation errors.

Booter	Duration			Total IPs		
	Original	Merged	Difference	Original	Merged	Difference
B1	295	295	0%	4177	4177	0%
B2	281	281	0%	65	65	0%
B3	278	278	0%	43	43	0%
B5	368	368	0%	7393	7393	0%
B6	139	139	0%	6972	6968	**0.05737%**
B7	531	531	0%	5835	5835	0%
B8	169	169	0%	251	251	0%
B9	155	155	0%	3507	3507	0%

Table 6.8: Percentage difference for duration and total IP addresses statistics for the parallel analysis.

Booter	Number of Packets			Total MBs		
	Original	Merged	Difference	Original	Merged	Difference
B1	12706379	12706271	**0.00085%**	11977	11975	**0.0167%**
B2	8996428	8996428	0%	7901	7900	**0.01266%**
B3	13213428	13213428	0%	12279	12278	**0.00814%**
B5	2040578	2040578	0%	258	258	0%
B6	9726155	9686747	**0.40517%**	11087	11043	**0.39686%**
B7	16671064	16659698	**0.06817%**	18172	18160	**0.06603%**
B8	5686681	5686681	0%	6882	6881	**0.01453%**
B9	13060766	13060483	**0.00216%**	15765	15763	**0.01269%**

Table 6.9: Percentage difference for number of packets and total megabytes statistics for the parallel analysis.

Booter	Average bps			Peak bps		
	Original	Merged	Difference	Original	Merged	Difference
B1	324818998	324815943	**0.00094%**	737572704	737572704	0%
B2	224949461	224949461	0%	380167272	380167272	0%
B3	353372702	353372700	0%	598343160	598343160	0%
B5	5621217	5621217	0%	7699784	7699784	0%
B6	638133538	635660685	**0.38751%**	1642181152	1639925896	**0.13733%**
B7	273790566	273634896	**0.05686%**	388161296	387951496	**0.05404%**
B8	325795771	323879325	**0.58823%**	1034913480	1034913480	0%
B9	813681058	813663528	**0.00215%**	2988318752	2988298016	**0.00069%**

Table 6.10: Percentage difference for average and peak bits per second statistics for the parallel analysis.

Booter	Average pps			Peak pps		
	Original	Merged	Difference	Original	Merged	Difference
B1	43072	43070	**0.00464%**	101574	101574	0%
B2	32015	32015	0%	53660	53660	0%
B3	47530	47529	**0.00210%**	81325	81325	0%
B5	5545	5544	**0.01803%**	7483	7483	0%
B6	69972	69687	**0.4073%**	179551	179314	**0.13199%**
B7	31395	31373	**0.07007%**	44478	44449	**0.06520%**
B8	33649	33450	**0.5914%**	106762	106762	0%
B9	84263	84259	**0.00475%**	306620	306618	**0.00065%**

Table 6.11: Percentage difference for average and peak packets per second statistics for the parallel analysis.

Booter	Fragmented	Not Fragmented	Fragmentation Percentage
B1	2045982	10660397	16.10%
B2	1117944	7878484	12.43%
B3	4405990	8807438	33.34%
B5	0	2040578	0.00%
B6	6219336	3506819	63.94%
B7	10681664	5989400	64.07%
B8	3578019	2108662	62.92%
B9	8035480	5025286	61.52%

Table 6.12: Fragmentation percentage of the packets found in the analyzed Booter traces.

6.4.1. Runtime Statistics

Additionally, we measure the time it takes to complete the execution of the entire algorithm when running the Parallel Dissector on the 9 Booters and the CSE & CIC DDoS traces. We measure the time over multiple iterations by varying the amount of maximum memory allocated to the Dissector process. For each group of traces, we determine the maximum amount of memory based on the file sizes. In particular, we show the results for the 9 Booters in Table 6.13 where the maximum amount of memory requirements provided during the experiments are 1 GB, 5 GB, and 30 GB. Since the CSE & CIC DDoS traces are smaller, we reduce the maximum amount of memory to 500 MB, 1 GB, and 2.5 GB with their respective results found in Table 6.14. The tables show the processing speed measured in GB per hour, this is obtained using the total runtime in seconds and the file size. We note that the average speed does not change significantly when using a lesser amount of memory. In some cases, the processing speed decreases when allocating a larger amount of memory. For example, Booter 4 is processed at a speed of 14.28 GB/h with a maximum memory of 5 GB, and when the memory is increased to 30 GB the processing speed drops to 13.81 GB/h. Similar conclusions can be drawn from the results of the other 8 Booters. For the CSE & CIC DDoS traces, we notice the same pattern where in some cases the processing speed slightly increases. For example, the Slowloris and LOIC traces were processed faster with 2.5 GB of memory than 1 GB; however, the difference is less than 0.05 GB/h, which is a small improvement.

Attack	File Size (MB)	GB/h (1 GB)	GB/h (5 GB)	GB/h (30 GB)
B1	11648	10.69	10.82	10.74
B2	7679	10.4	10.33	10.32
B3	11916	11.06	11.12	10.9
B4	47695	13.71	14.28	13.81
B5	292	1.86	1.9	1.87
B6	10759	13.42	13.34	13.15
B7	17608	12.24	12.15	11.91
B8	6675	15.28	15.66	15.29
B9	15322	15.11	15.21	14.99

Table 6.13: Parallel Dissector runtime statistics for the 9 Booters varying the maximum memory available.

Attack	File Size (MB)	GB/h (500 MB)	GB/h (1 GB)	GB/h (2.5 GB)
SlowHTTPTest	4113	3.42	3.65	3.64
Slowloris	181	5.82	5.72	5.77
LOIC	8097	1.29	1.39	1.43
HOIC	2026	2.88	3.16	3.09
LOIC-UDP	17429	1.3	1.36	1.36

Table 6.14: Parallel Dissector runtime statistics for the CSE & CIC DDoS traces varying the maximum memory available.

6.5. NaWas Dataset

We analyze all sflow files in the NaWas dataset without needing to split them further because they already contain a small amount of information and thus do not consume a significant amount of memory. In this case,

the sflow file is processed with the same method used for NetFlow files with the addition that the statistics are automatically scaled based on the sampling rate found in the export. Therefore, the results we obtain do not correspond to the exact numbers of the original attack, but they are an approximation based on how the flows were collected. Table 6.15 displays the characteristics of both datasets, we refer to the datasets based on the time period analyzed. The sflow files contain 1420 attacks in the period February 2022 - June 2022, while we see 962 attacks for the reports corresponding to the September 2022 - February 2023 period. The biggest attacks in the dataset reached a peak of 381 Gbps and 134.8 Mpps, these peak values correspond to different attacks. The longest attack lasted 23 hours and 30 minutes, however, the majority of attacks last between 4 and 10 minutes. Although the average strength of the attacks increased between the first and second period, the median values have decreased over the same period. This is an indication that most attacks have not increased in strength but we see a larger amount of 100+ Gbps attacks, raising the average values. Furthermore, the duration of the measured attacks increased across the entire dataset with a median duration of over 10 minutes in the period September 2022 - February 2023. We note that the measured duration of the attack corresponds to the mitigation period, therefore it does not include the detection phase that can take a few minutes.

	February 2022 - June 2022			September 2022 - February 2023		
	Duration (s)	**Peak Gbps**	**Peak Mpps**	**Duration (s)**	**Peak Gbps**	**Peak Mpps**
min	16	0.221	0.032	60	2.5e-07	2e-05
25%	179	1.370	0.332	300	0.616	0.130
median	239	3.498	0.568	660	1.91	0.299
mean	450	6.432	1.571	3332	8.121	1.905
75%	359	6.217	1.024	1920	4.1	0.878
max	26964	239.29	342.58	83940	381.0	134.8
Total attacks	1420			962		

Table 6.15: Characteristics of the NaWas dataset.

Figure 6.8 shows the cumulative distribution of the duration of attacks in the second period. From the plot, we notice that 10% of the attacks lasted more than 2 hours and 5% continued for over 6 hours. Long-lasting attacks are generally unusual when they are launched from Booter services, because these providers often do not offer such a service. Typically, the Booter provider offers to target a victim for 2 to 10 minutes, sometimes up to 1 hour. However, we seem to find a significant percentage of attacks that are much longer than that which can indicate that the attacker might be using its own infrastructure to launch the attack. We researched the currently available stressers and found out very few offering to send DDoS attacks longer than a few hours. Among the ones offering long-lasting attacks, we find one selling attacks up to 12 hours at a price point of over $6000. Therefore, we can probably attribute the long-lasting attacks to independent attackers rather than someone purchasing it from a Booter service. Furthermore, long-lasting attacks tend to contain more attack vectors than their short-lasting counterparts with the most common vector being a TCP flag attack.

To show the frequency of the attacks and their corresponding strength, we plot in Figure 6.9 all attacks found in both datasets with the corresponding date on the x-axis and peak Gbps on the y-axis. First, we notice that our sflow dataset is missing several days of data corresponding to March 2022. We could not obtain the remaining flows because they were deleted as part of regular deletion schedules run by the scrubber. In particular, the dataset we obtained was previously used for research purposes and therefore the flows were kept for a longer period of time. The report dataset instead was complete and it is not missing any attack in the analyzed period. The number of large-scale attacks observed increased in the second period. In the September 2022 - February 2023 period, we see 7 attacks stronger than 200 Gbps, while only 3 instances of them are found in the previous period. Out of the 7 200+ Gbps attacks, 3 of them reached peaks larger than 300 Gbps, which is more than all attacks in the February 2022 - June 2022 period.

Next, we look at the attack vectors that make up the attacks. Figure 6.10 shows the distribution of the number of attack vectors found in each attack for both time periods. Although in the first period more than 70% of the attacks used multiple vectors, the second period is characterized by 70% attacks that contain a single attack vector. Nevertheless, 30% of the attacks were multi-vector and we notice some instances where the attacker used up to 8 different vectors in a single attack. Similarly in the February 2022 - June 2022 period, the maximum number of attack vectors was 7, with more than 35% of the attacks using at least 4 vectors.

Furthermore, we analyze the number of attack vectors modifying the definition to include the destination port. In this case, each vector is associated to the destination port that was targeted, in case a significant

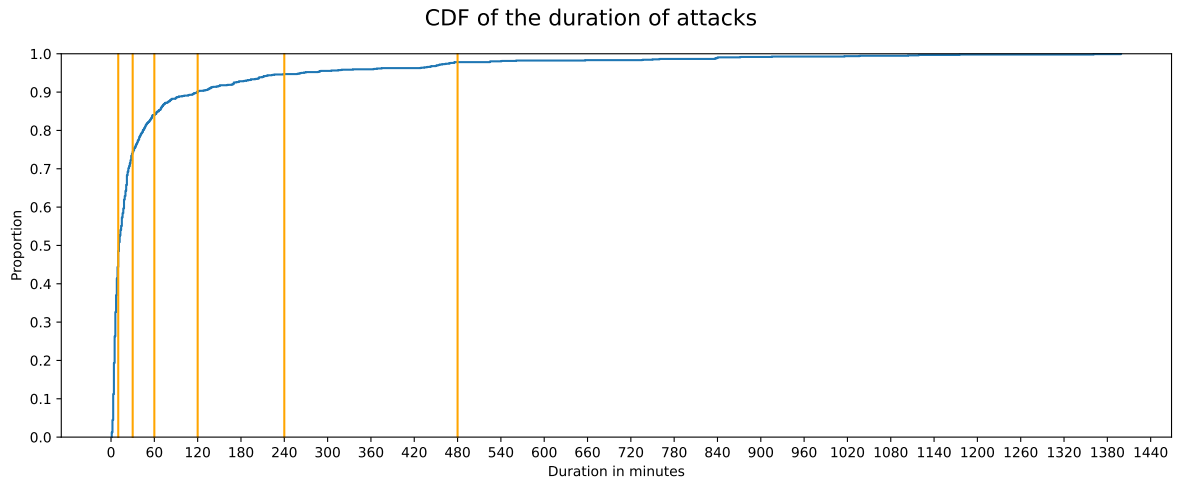

Figure 6.8: Cumulative distibution of the duration of the attacks in the NaWas report dataset. The first two vertical lines represent 10 and 30 minutes.

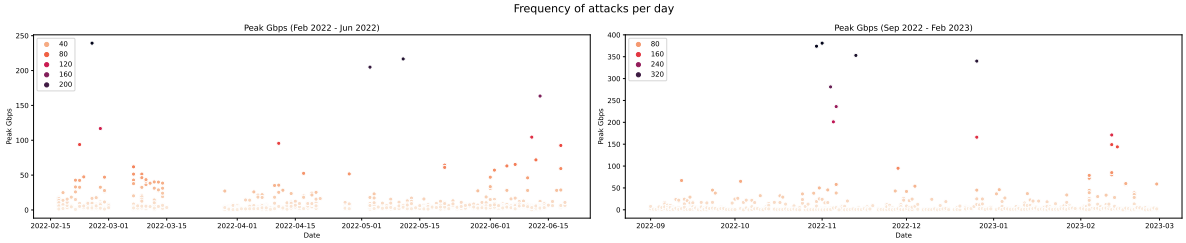

Figure 6.9: Frequency of attacks per day with the corresponding peak Gbps for each attack.

percentage of traffic was sent to that single port. When a set of attacked destination ports cannot be identified, we mark the destination port as being randomized and the attack vector is not split into multiple ones. The results are shown in Figure 6.11 and we notice a similar dominant position of single-vector attacks in the second period. Moreover, the maximum number of attack vectors increases substantially with a peak of 25 attack vectors in a single attack and 18% of all attacks containing 10 different vectors for the first period. Single and dual vector attacks remain the most common, comprising 40.9% and 75.1% of attacks, respectively, for the first and second time period.

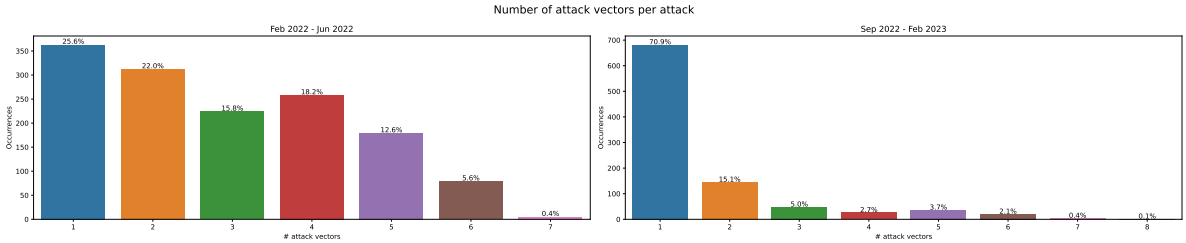

Figure 6.10: Number of attack vectors per attack in the NaWas dataset.

Figure 6.12 and Figure 6.13 show the most common attack vectors as well as the most common pairs and triples respectively for the first and second period. In particular, the first period is characterized by several instances of HTTP-based attack vectors, while the second period does not appear to report them among the most used ones. The difference in results could be explained by the different detection capabilities because the scrubber reports are generated in real time, with the possibility of inspecting the traffic. In the case of the Dissector, the detection is done based on the ports found in the traffic, which is not always reliable because the adversary can choose to send traffic from any port. Furthermore, we note that the discovered attack vectors could correspond to normal traffic, however, we believe that this should not be the case as that would mean

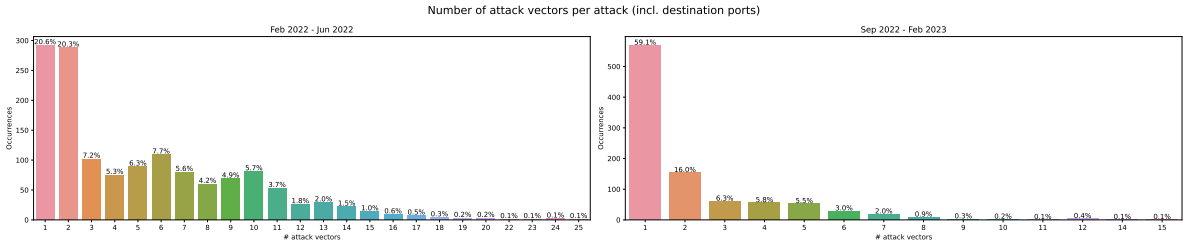

Figure 6.11: Number of attack vectors per attack in the NaWas dataset separated by destination port.

that the scrubber service identified such data as malicious. In any case, since the sampling rate is set to a very high value, it could be that a non-negligible percentage of flows belonging to regular users was exported. We also see similarities between the two distributions where UDP flood, TCP flood, and DNS amplification are the most used attack vectors in both cases. In the second period, we also notice popular attack vectors among the top such as NTP and Memcached. Among attacks with more than 2 vectors, the combination of DNS with another vector is the most popular, found in 39.4% of the cases. The most popular triple of attack vectors is the combination of DNS, SSDP, and WS-Discovery amplification exploits. Simple Service Discovery Protocol (SSDP) amplification attacks leverage Universal Plug and Play (UPnP) [53] devices where port 1900 is exposed to the Internet and responds to discovery requests. Similarly, Web Services Dynamic Discovery (WS-Discovery) [54] is another protocol used to discover devices in a local network but it can also be abused when the device responds to any request from the outside, since it uses SOAP-over-UDP and therefore the request can be spoofed.

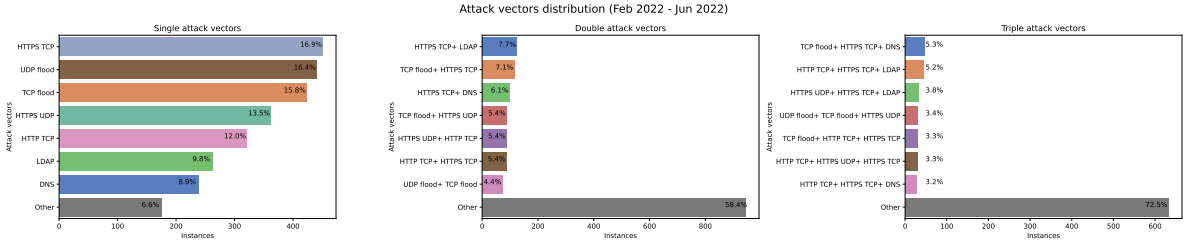

Figure 6.12: Distribution of the attack vectors in the period from February 2022 to June 2022.

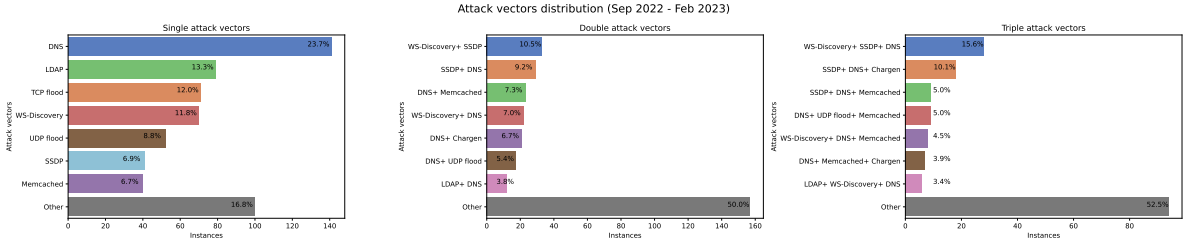

Figure 6.13: Distribution of the attack vectors in the period from September 2022 to February 2023.

Since the attack vectors are identified using the port numbers found in the packets, we plot their distributions in Figure 6.14 and Figure 6.15 respectively for the February 2022 - June 2022 and September 2022 - February 2023 periods. We notice that HTTP-related ports are not among the most used in the second period, which is why the corresponding attack vectors do not appear in the previous analysis. Moreover, port 389 which corresponds to LDAP is found among the top in both periods and port 3702 corresponding to WS-Discovery is also in the most seen ports for the second period. From the destination ports, we note that both port 80 and 443 are popular in both periods, indicating that the victims might be hosting web applications or servers. We also notice two unusual port values, namely 1333 and 6672 that are respectively assigned by IANA to PASSWRD-POLICY and VISION-SERVER [52]. Lastly, a significant percentage of traffic is displayed as port 0 because the received reports use it to indicate that the port numbers were randomized. However, we note that in the first period,

the port numbers are accurate but calculated from sampled values. In particular, we can attribute part of port 0 traffic to fragmented IP packets [55].

In Figure 6.14 and Figure 6.15, we also include the distribution of TCP flags found in TCP-based attacks. In the February 2022 - June 2022 period, 73.4% of the attacks used the ACK flag and 22.9% used it in combination with the PSH flag. In the September 2022 - February 2023, the ACK flag was found in 19.1% of the attacks and the most popular one was actually the SYN flag for 22.8% of the cases. However, in more than 50% of the attacks of the second period the ACK flag was found both by itself and in combination with other flags.

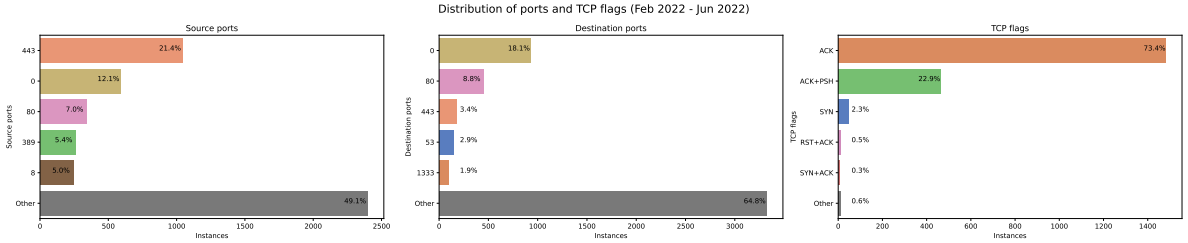

Figure 6.14: Distribution of the source/destination ports and TCP flags in the period from February 2022 to June 2022.

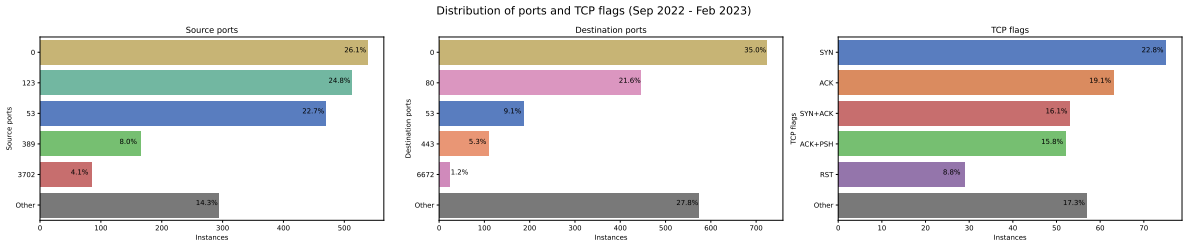

Figure 6.15: Distribution of the source/destination ports and TCP flags in the period from September 2022 to February 2023.

6.5.1. Peak Attacks Case Study

In this section, we analyze the peak of attacks found in the dataset. In particular, we plot the distribution of daily attacks in the February 2022 - June 2022 period in Figure 6.16. The missing datapoints in March 2022 can also be noticed here by the sharp jump on March 15. We notice two days with a much higher number of attacks than the rest of the data. The two days are April 28 with 139 attacks and June 12 with 171 attacks.

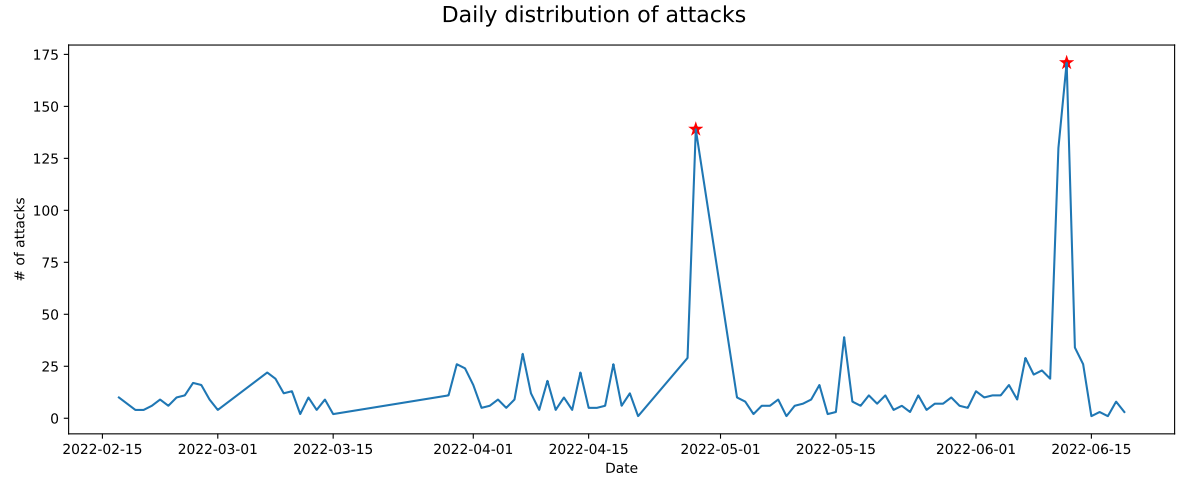

Figure 6.16: Number of attacks launched per day in the period from February 2022 to June 2022.

To investigate the characteristics of these two peaks, we display the distribution of the number of attack vectors for each peak in Figure 6.17. On the one hand, during the April peak we see 90% of the attacks using

more than 1 attack vector, 87.1% of the attacks leveraged more than 4 vectors. In particular, the attacks contained a mix of LDAP amplification, HTTP, and TCP flood, which also correspond to the most common vectors we previously discovered. On the other hand, the June peak was characterized by 93% of single vector attacks, mostly TCP and UDP flood.

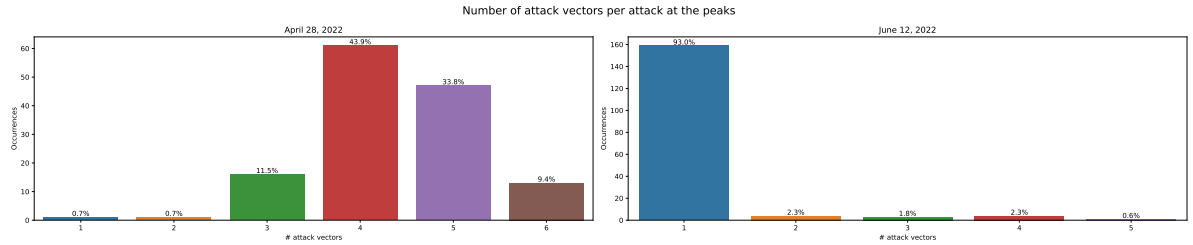

Figure 6.17: Number of attack vectors found in the attacks at the peak days in the period from February 2022 to June 2022.

Furthermore, we consider four individual attacks that were among the strongest based on their peak bps and pps values. In the following sections, we examine each attack in depth to understand their own peculiarities, as well as how many similar attacks can be found in the dataset.

DNS + Hikvision + NTP + SNMP Amplification (381 Gbps)

This is the strongest attack recorded in the dataset peaking at 381 Gbps and 25.7 Mpps. It consisted of 4 different attack vectors, DNS, Hikvision, NTP, and SNMP amplification. In addition, the attack lasted for more than 15 hours, indicating that the attacker probably owned the infrastructure to launch such a powerful attack. In the report, we see that the destination port is 0 and therefore the attacked ports were randomized. We find 9 similar attacks leveraging the DNS, NTP, and SNMP vectors at the same time. Moreover, we observe only four attacks that peaked at more than 300 Gbps.

6-Vector DDoS Attack (281 Gbps)

Among attacks with a large number of vectors, this was the strongest, with a peak of 281 Gbps and 32.6 Mpps. It lasted for 3 hours and included 6 different vectors, which become 14 when we include the destination port. The attack vectors were a TCP flag attack combined with DNS, NTP, Chargen, Memcached, and Apple Remote Desktop amplification. The Apple Remote Desktop amplification attack is similar to the discovery exploits such as SSDP and WS-Discovery, where a Mac OS machine responds to discovery requests made on port 3283. This is not the only instance of Mac OS based exploits, as we find 15 attacks leveraging those. Furthermore, more than 100 attacks in our data set used six or more combined attack vectors.

Malware TCP Flood (205 Gbps)

This attack was a 3 minute TCP ACK flood with peaks of 205 Gbps and 26 Mpps. The destination port was 80, thus we can assume that it was intended for a web service. The source port was instead 65535 which is not reserved by any application, however, we find that several trojans use such a port [56], [57]. This means that the attack could be originating from a botnet of infected devices. We do not find evidence that this botnet is used by a Booter provider because there are only 3 attacks in the dataset that use it. However, all these attacks were directed towards port 80 or 443, showing that the possible owner of the botnet is repeatedly using it to attack a certain web service. Since we remove the target information, we could not analyze this further.

DNS Amplification (105 Gbps)

This DNS amplification attack was unique based on the set of destination ports that were attacked. It lasted for 8 minutes with peaks of 105 Gbps and 10 Mpps. In particular, some of the attacked ports seem to be related to different games, indicating that the victim might be hosting several game servers. Among the ports we could associate, we find port 40881 related to Brothers in Arms, port 27586 for QuakeWorld, port 4603 for Men & Mice. Each port represented 10% of the attack, the rest of the ports could not be associated with any specific game. In addition, the victim received traffic from over 50k IP addresses. In the dataset, DNS amplification attacks usually leverage around 2k amplifiers with a maximum of 80k servers.

6.5.2. Booters Takedown Case Study

Lastly, we decide to look at the effect that the recent takedown of 48 Booter services had on the DDoS landscape [58]. We analyze the report from September 2022 - February 2023 dividing it into 2 separate periods, before and after the takedown of December 14, 2022. In this case, we plot the distribution of the most common attack vectors seen before and after the takedown in Figure 6.18. NTP, WS-Discovery, LDAP, and UDP remain among the most commonly used attack vectors. However, we see Memcached amplification attacks disappearing from the list and Chargen amplification attacks taking the spot with 10.7% of attacks launched between December 2022 and February 2023. Furthermore, we see a decrease in the strength and frequency of TCP flood and flag-based attacks and an increase in SSDP amplification attacks ranging from 5.0% to 9.8%.

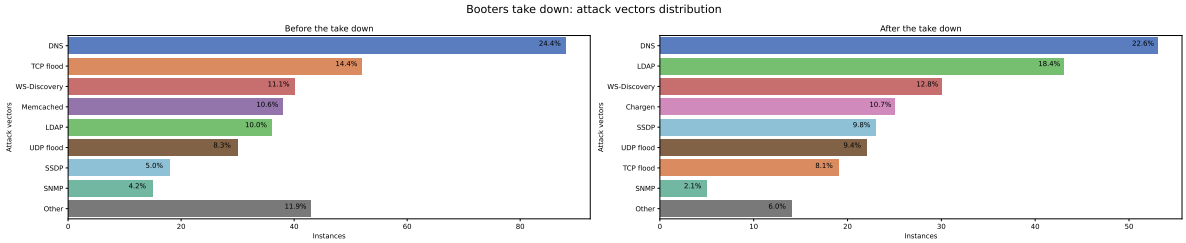

Figure 6.18: Attack vectors distribution changes before and after the Booters takedown.

Additionally, we analyze the distribution of single- and multi-vector attacks for every week in the September 2022 - February 2023 period, shown in Figure 6.19. First, during the 51st week of 2022 there were no multi-vector attacks and that corresponds to the week of December 14, 2022. Second, the weekly number of attacks dropped to less than 40 in the 2 weeks preceding the takedown and jumped back to more than 80 only in the second week of January. While it seems that the takedown had an effect on the frequency of attacks for a few weeks, the overall impact does not seem to suggest that this had a significant effect on the DDoS landscape. Moreover, we find similar drops below 40 weekly attacks in our dataset, such as the 41st week of 2022.

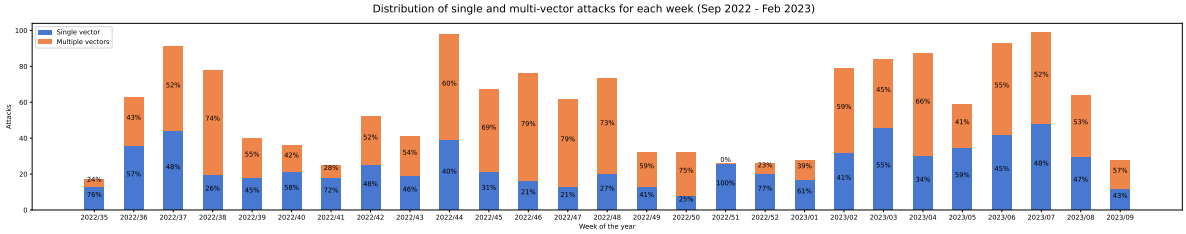

Figure 6.19: Distribution of single and multi vector attacks for every week in the September 2022 - February 2023 period.

Lastly, Figure 6.20 displays the attack vector distribution for each month in the second period. As previously mentioned, DNS remains the most common attack vector for every month except for January, where LDAP took the first place with 29.3% of attacks. In particular, we could attribute the decrease in DNS amplification attacks to the takedown, indicating that LDAP amplification attacks are either launched by other Booters or by individuals owning the infrastructure. We notice less popular attacks appearing only for a single month. For example, the Steam protocol amplification accounted for 6.7% of the attacks in the month of October or SNMP amplification representing 10.4% of December's attacks.

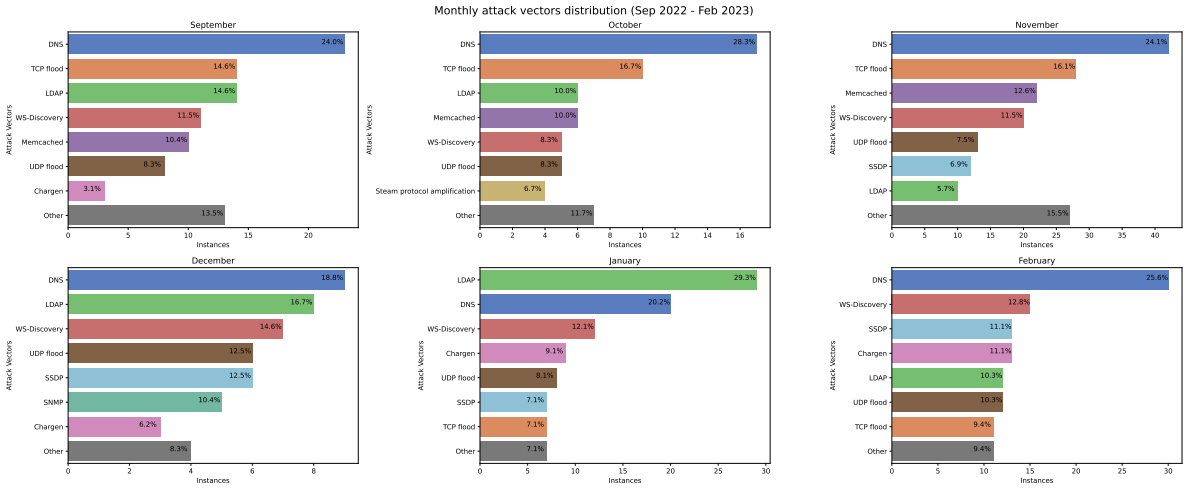

Figure 6.20: Distribution of attack vectors for each month in the September 2022 - February 2023 period.

7 Discussion

In this section, we address some ethical issues that arise from the handling of sensitive data in our analysis. Next, we reflect on the results obtained from our analysis. In particular, we focus on its limitations and how they can be fixed in future iterations.

7.1. Ethical Considerations

The datasets used contain network data, therefore the IP addresses of the servers involved can be found in the packets. We do not analyze the target IP addresses further than necessary, which in our case corresponds to the target inference step and the filtering of attack traffic for the attack vectors' analysis. Once the attack traffic is extracted, the victim address is removed and it is not found in the fingerprint. On the other hand, we analyze the source IP addresses in detail, these usually correspond to the attacker and their packet data mostly contains malicious information. Both the source IP addresses and their respective ASN are found in the generated fingerprint, and they are used for the features' extraction as well as in the spoofing analysis. Although we do not believe that the sources of an attack could be considered personal information, we do not include their IP addresses in our final report. The IP addresses found in the report are used for example purposes and generally correspond to well-known services, they are not the actual IP addresses we extracted from the datasets. Moreover, several datasets we use have already been anonymized, thus we do not analyze the source IP addresses for those cases.

7.2. Reflection and Limitations

Using the methodology developed, we analyze the characteristics of various DDoS attacks. In particular, we can extract the attack vectors that compose the attack and compute the respective statistics for each one of them. Furthermore, we parallelize the Dissector execution such that we can run it in low-memory environments. For example, a regular computer can be used to analyze large-scale attacks that would not be possible due to the high amount of computing power required. The architecture of the Dissector is built such that it does not require any specialized hardware components.

In addition, we show how the DDoS landscape changed over time with the increase in strength and number of attack vectors. We notice how the recent takedown of 48 Booters affected the operations of a scrubbing service with a decrease in weekly attacks. From the entire analysis, we see that after a decade the attacks are much stronger and often contain multiple attack vectors. Comparing the 2007 CAIDA attack peaking at 80 Mbps with the NaWas dataset, we see how even the weakest attacks are larger than 1 Gbps and can reach peaks of 381 Gpbs.

Although we successfully classify the attacks in our dataset, there are some limitations to our analysis, especially when analyzing traces containing both attack and normal traffic. In most cases, normal traffic belongs to a small percentage of packets that is automatically excluded from the statistics. However, this is not always the case and if a large percentage of packets are non-attack traffic they will be considered part of the attack. This is crucial for detection systems because they should not identify regular traffic as an attack and start the mitigation procedures. Furthermore, the target inference consists of basic heuristics based on the number of packets sent to an IP address, and non-attack traffic could influence the decision. In particular, we notice throughout our experiments that when the target cannot be inferred automatically, we have to manually specify

it. Overall, the impact of non-attack traffic can be reduced so that the Dissector could also be deployed in production environments.

From the spoofing analysis, we are able to determine whether the attacker likely spoofed the packets and used a single machine to launch the attack. For non-spoofed attacks, we can still use the maximum TTL distribution to discover the operating systems running on the infected machines. This could be used to understand which operating systems are more vulnerable to a specific exploit. In addition, we can use this information to understand if new attack exploits are leveraging a specific type of device, especially if they are executed using a botnet. Unfortunately, the limited PCAP dataset we obtained did not have a large variety of distributions and the sflow data does not contain the TTL values. Therefore, we could not determine the distribution of a broad spectrum of attacks. However, we expect such analysis to be applicable to more recent attacks and provide useful insights explaining how novel attacks are executed.

8

Conclusion

In conclusion, we show how we can study DDoS attacks from the corresponding network capture. Our work builds upon previous research and improves the tooling to provide deeper insights into the attack. In particular, we expand the software to calculate the peak strength of the attack, and we include additional insights for each attack source to the final fingerprint. Furthermore, we present the split-and-merge algorithm used to parallelize the analysis of large-scale attacks. The algorithm consists of splitting the attack network trace into smaller chunks that can be individually analyzed and later merged to obtain a single summary. Our experiments indicate that the algorithm's output error compared to the original file is generally less than 0.5% and therefore could be used to generate meaningful results with lower memory requirements.

To complement our research, we analyze several attacks from previously studied datasets and a recent selection of launched attacks. We show the results of our analysis by highlighting the discovered attack vectors and the characteristics of the DDoS landscape. Furthermore, we show how the recent takedown of 48 Booters affected the frequency of weekly attacks for only a couple of weeks. In addition, we elaborate on the limitations of our tooling and the effect that non-attack packets have on the results quality. Lastly, we use the TTL value found in the packet headers to analyze whether spoofing occurred and demonstrate how to infer the operating systems used by the attack sources from the TTL distribution.

8.1. Future Work

In the report, we focus on the analysis of PCAP files, although other formats are also supported. In particular, NetFlow files are usually found in large-scale operations and they are already supported by the Dissector. The performance improvement obtained when splitting a NetFlow file into multiple chunks has not been looked into. Our merge algorithm could already be applied in case there are multiple NetFlow files corresponding to the same attack. However, future research could explore the optimal way to analyze NetFlow exports such that the output quality is similar to the results obtained from PCAP files. Although PCAP files contain more information such as the packet data, the Dissector analysis of NetFlow files could be augmented by inferring what the original traffic looked like.

The automated target inference used by the Dissector follows basic rules and can find a single IP address or a large subnet (/24 for IPv4 and /64 for IPv6). In all other cases, we need to manually specify the target to avoid creating the wrong fingerprint. Future work could focus on automating the inference using more advanced techniques to differentiate attack traffic from normal traffic. For example, a larger selection of subnet ranges could be supported with defined thresholds based on the number of packets. Moreover, machine learning methods could be used to train a model that can extract the traces corresponding to a DDoS attack from a larger network capture. For example, the model could use the packet size distribution to separate malicious traffic from regular traffic.

The main limitation of the Dissector is the need to load the entire network trace into memory to start the analysis. This limitation could be addressed by changing the architecture used to analyze the attacks. For example, packets could be incrementally analyzed, storing an intermediate summary that is sufficient to compute the statistics. This means that we could process the trace as a stream and compute the fingerprint without requiring a significant amount of memory. In addition, this could be used to generate summaries of real-time traffic. Alternatively, storing the information with a dedicated format optimized for data analysis could greatly

improve the speed and memory requirements. In particular, big data analytics engines for large-scale analyses, such as Apache Spark [59], are optimized for this use-case and therefore could be leveraged to re-implement the feature extraction using their framework.

References

[1] R. E. Calem, "New york's panix service is crippled by hacker attack," 2018, Accessed: 2023-07-04. [Online]. Available: https://www.senki.org/wp-content/uploads/2017/10/New-Yorks-Panix-Service-Is-Crippled-by-Hacker-Attack.pdf.

[2] Public Access Networks Corporation, *Panix*, https://panix.com/, Accessed: 2023-07-04. [Online]. Available: https://panix.com/.

[3] International Risk Governance Council. "Cyber Security Risk Governance." Accessed: 2023-07-04. (2015), [Online]. Available: https://irgc.org/wp-content/uploads/2018/09/Cyber-Security-Risk-Governance-29-30-October-2015-Workshop-Report.pdf.

[4] M. Jonker, A. King, J. Krupp, C. Rossow, A. Sperotto, and A. Dainotti, "Millions of targets under attack: A macroscopic characterization of the dos ecosystem," in *Proceedings of the 2017 Internet Measurement Conference*, ser. IMC '17, London, United Kingdom: Association for Computing Machinery, 2017, pp. 100–113, ISBN: 9781450351188. DOI: 10.1145/3131365.3131383. [Online]. Available: https://doi.org/10.1145/3131365.3131383.

[5] D. Kopp, C. Dietzel, and O. Hohlfeld, "DDoS never dies? An IXP perspective on DDoS amplification attacks," in *Passive and Active Measurement - 22nd International Conference, PAM 2021, Virtual Event, March 29 - April 1, 2021, Proceedings*, O. Hohlfeld, A. Lutu, and D. Levin, Eds., ser. Lecture Notes in Computer Science, vol. 12671, Springer, 2021, pp. 284–301. DOI: 10.1007/978-3-030-72582-2_17.

[6] E. Cooke and F. Jahanian, "The zombie roundup: Understanding, detecting, and disrupting botnets," in *Steps to Reducing Unwanted Traffic on the Internet Workshop, SRUTI'05, Cambridge, MA, USA, July 7, 2005*, D. Katabi and B. Krishnamurthy, Eds., USENIX Association, 2005.

[7] B. Krebs. "Who is anna-senpai, the mirai worm author?" Accessed: 2023-07-04. (2017), [Online]. Available: https://krebsonsecurity.com/2017/01/who-is-anna-senpai-the-mirai-worm-author/.

[8] M. Antonakakis, T. April, M. Bailey, *et al.*, "Understanding the Mirai Botnet," in *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, USENIX Association, 2017, pp. 1093–1110.

[9] L. Franceschi-Bicchierai. "How 1.5 Million Connected Cameras Were Hijacked to Make an Unprecedented Botnet." Accessed: 2023-07-04. (2016), [Online]. Available: https://www.vice.com/en/article/8q8dab/15-million-connected-cameras-ddos-botnet-brian-krebs.

[10] O. Yoachimik. "Cloudflare thwarts 17.2M rps DDoS attack — the largest ever reported." Accessed: 2023-07-04. (2021), [Online]. Available: https://blog.cloudflare.com/cloudflare-thwarts-17-2m-rps-ddos-attack-the-largest-ever-reported/.

[11] L. Tung. "Microsoft: Here's how we stopped the biggest ever DDoS attack." Accessed: 2023-07-04. (2022), [Online]. Available: https://www.zdnet.com/article/microsoft-heres-how-we-stopped-the-biggest-ever-ddos-attack/.

[12] R. van Rijswijk-Deij, A. Sperotto, and A. Pras, "DNSSEC and its potential for DDoS attacks: A comprehensive measurement study," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, ser. IMC '14, Vancouver, BC, Canada: Association for Computing Machinery, 2014, pp. 449–460, ISBN: 9781450332132. DOI: 10.1145/2663716.2663731.

[13] M. Nawrocki, M. Jonker, T. C. Schmidt, and M. Wählisch, "The Far Side of DNS Amplification: Tracing the DDoS Attack Ecosystem from the Internet Core," in *Proceedings of the 21st ACM Internet Measurement Conference*, ser. IMC '21, Virtual Event: Association for Computing Machinery, 2021, pp. 419–434, ISBN: 9781450391290. DOI: 10.1145/3487552.3487835.

[14] B. Collier, D. R. Thomas, R. Clayton, and A. Hutchings, "Booting the booters: Evaluating the effects of police interventions in the market for denial-of-service attacks," in *Proceedings of the Internet Measurement Conference*, ser. IMC '19, Amsterdam, Netherlands: Association for Computing Machinery, 2019, pp. 50–64, ISBN: 9781450369480. DOI: 10.1145/3355369.3355592.

[15] NETSCOUT, *What is a ddos extortion attack?* https://www.netscout.com/what-is-ddos/what-is-ransom-ddos-attack, Accessed: 2023-07-04. [Online]. Available: https://www.netscout.com/what-is-ddos/what-is-ransom-ddos-attack.

[16] Imperva, *Cyber security leader | imperva*, https://www.imperva.com, Accessed: 2023-07-04. [Online]. Available: https://www.imperva.com.

[17] Cloudflare, *Cloudflare - the web performance & security company*, https://www.cloudflare.com, Accessed: 2023-07-04. [Online]. Available: https://www.cloudflare.com.

[18] J. Czyz, M. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. Bailey, and M. Karir, "Taming the 800 pound gorilla: The rise and decline of NTP DDoS attacks," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, ser. IMC '14, Vancouver, BC, Canada: Association for Computing Machinery, 2014, pp. 435–448, ISBN: 9781450332132. DOI: 10.1145/2663716.2663717.

[19] D. Wagner, D. Kopp, M. Wichtlhuber, *et al.*, "United We Stand: Collaborative Detection and Mitigation of Amplification DDoS Attacks at Scale," in *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, ACM, 2021, pp. 970–987. DOI: 10.1145/3460120.3485385.

[20] M. Wichtlhuber, E. Strehle, D. Kopp, *et al.*, "IXP Scrubber: Learning from Blackholing Traffic for ML-Driven DDoS Detection at Scale," in *Proceedings of the ACM SIGCOMM 2022 Conference*, ser. SIGCOMM '22, Amsterdam, Netherlands: Association for Computing Machinery, 2022, pp. 707–722, ISBN: 9781450394208. DOI: 10.1145/3544216.3544268.

[21] C. Dietzel, M. Wichtlhuber, G. Smaragdakis, and A. Feldmann, "Stellar: Network attack mitigation using advanced blackholing," in *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '18, Heraklion, Greece: Association for Computing Machinery, 2018, pp. 152–164, ISBN: 9781450360807. DOI: 10.1145/3281411.3281413.

[22] V. Giotsas, G. Smaragdakis, C. Dietzel, P. Richter, A. Feldmann, and A. Berger, "Inferring BGP blackholing activity in the internet," in *Proceedings of the 2017 Internet Measurement Conference*, ser. IMC '17, London, United Kingdom: Association for Computing Machinery, 2017, pp. 1–14, ISBN: 9781450351188. DOI: 10.1145/3131365.3131379.

[23] G. C. M. Moura, R. de Oliveira Schmidt, J. S. Heidemann, *et al.*, "Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event," in *Proceedings of the 2016 ACM on Internet Measurement Conference, IMC 2016, Santa Monica, CA, USA, November 14-16, 2016*, ACM, 2016, pp. 255–270.

[24] "Anycast agility: Network playbooks to fight DDoS," in *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA: USENIX Association, Aug. 2022.

[25] G. C. M. Moura, C. Hesselman, G. Schaapman, N. Boerman, and O. de Weerdt, "Into the DDoS maelstrom: A longitudinal study of a scrubbing service," in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2020, pp. 550–558. DOI: 10.1109/EuroSPW51379.2020.00081.

[26] J. J. Santanna, R. van Rijswijk-Deij, R. Hofstede, *et al.*, "Booters - an analysis of ddos-as-a-service attacks," in *IFIP/IEEE International Symposium on Integrated Network Management, IM 2015, Ottawa, ON, Canada, 11-15 May, 2015*, R. Badonnel, J. Xiao, S. Ata, F. D. Turck, V. Groza, and C. R. P. dos Santos, Eds., IEEE, 2015, pp. 243–251. DOI: 10.1109/INM.2015.7140298.

[27] R. Sommese, K. Claffy, R. van Rijswijk-Deij, *et al.*, "Investigating the impact of ddos attacks on dns infrastructure," in *Proceedings of the 22nd ACM Internet Measurement Conference*, ser. IMC '22, Nice, France: Association for Computing Machinery, 2022, pp. 51–64, ISBN: 9781450392594. DOI: 10.1145/3517745.3561458.

[28] M. Jonker, A. Sperotto, R. van Rijswijk-Deij, R. Sadre, and A. Pras, "Measuring the adoption of DDoS protection services," in *Proceedings of the 2016 Internet Measurement Conference*, ser. IMC '16, Santa Monica, California, USA: Association for Computing Machinery, 2016, pp. 279–285, ISBN: 9781450345262. DOI: 10.1145/2987443.2987487.

[29] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic DDoS defense," in *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*, J. Jung and T. Holz, Eds., USENIX Association, 2015, pp. 817–832.

[30] Q. Jia, H. Wang, D. Fleck, F. Li, A. Stavrou, and W. Powell, "Catch me if you can: A cloud-enabled DDoS defense," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2014, pp. 264–275. DOI: 10.1109/DSN.2014.35.

[31] DDoS Clearing House. "DDoS dissector." Accessed: 2023-07-04. (2022), [Online]. Available: https://github.com/ddos-clearing-house/ddos_dissector.

[32] The Tcpdump Group. "Libpcap." Accessed: 2023-07-04. (2022), [Online]. Available: https://github.com/the-tcpdump-group/libpcap.

[33] B. Claise, "Cisco systems netflow services export version 9," RFC Editor, RFC 3954, Oct. 2004, Accessed: 2023-07-04. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3954.txt.

[34] Wireshark Foundation, *Extensible record format (ERF)*, https://wiki.wireshark.org/ERF, Accessed: 2023-07-04. [Online]. Available: https://wiki.wireshark.org/ERF.

[35] Wireshark Foundation, *Tshark manual page*, https://www.wireshark.org/docs/man-pages/tshark.html, Accessed: 2023-07-04. [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html.

[36] QOSIENT, LLC., *Argus*, https://openargus.org/, Accessed: 2023-07-04. [Online]. Available: https://openargus.org/.

[37] QOSIENT, LLC., *Ra manual page*, https://qosient.com/argus/man/man1/ra.1.pdf, Accessed: 2023-07-04. [Online]. Available: https://qosient.com/argus/man/man1/ra.1.pdf.

[38] CAIDA, *Routeviews prefix to as mappings dataset for ipv4 and ipv6*, https://www.caida.org/catalog/datasets/routeviews-prefix2as, Accessed: 2023-07-04. [Online]. Available: https://www.caida.org/catalog/datasets/routeviews-prefix2as.

[39] J. Fan, J. Xu, M. H. Ammar, and S. B. Moon, "Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme," *Computer Networks*, vol. 46, no. 2, pp. 253–272, 2004, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2004.03.033.

[40] R. Sharpe. "Editcap manual page." Accessed: 2023-07-04. (2023), [Online]. Available: https://linux.die.net/man/1/editcap.

[41] I. Schorr. "Capinfos manual page." Accessed: 2023-07-04. (2023), [Online]. Available: https://linux.die.net/man/1/capinfos.

[42] S. Siby, *Default TTL (time to live) values of different OS*, https://subinsb.com/default-device-ttl-values, Accessed: 2023-07-04. [Online]. Available: https://subinsb.com/default-device-ttl-values.

[43] SIDN. "Sidn." Accessed: 2023-07-04. (2023), [Online]. Available: https://www.sidn.nl/en/theme/about-sidn.

[44] J. J. Santanna, R. van Rijswijk-Deij, R. Hofstede, *et al.* "Datasets for booters - an analysis of DDoS-as-a-service attacks." Accessed: 2023-07-04. (2015), [Online]. Available: https://www.simpleweb.org/wiki/index.php/Traces#Booters_-_An_analysis_of_DDoS-as-a-Service_Attacks.

[45] The CAIDA UCSD, *DDoS attack 2007 dataset*, https://www.caida.org/catalog/datasets/ddos-20070804_dataset, Accessed: 2023-07-04. [Online]. Available: https://www.caida.org/catalog/datasets/ddos-20070804_dataset.

[46] Canadian Institute for Cybersecurity, *A realistic cyber defense dataset (cse-cic-ids2018)*, https://registry.opendata.aws/cse-cic-ids2018/, Accessed: 2023-07-04. [Online]. Available: https://registry.opendata.aws/cse-cic-ids2018/.

[47] *DARPA Scalable Network Monitoring (SNM) Program Traffic, IMPACT ID: USC-LANDER/DARPA_2009_malware-DDoS_attack-20091104/rev4384 . Traces taken 2009-11-04 to 2009-11-04. Provided by the USC/LANDER project (http://www.isi.edu/ant/lander).*

[48] *DARPA Scalable Network Monitoring (SNM) Program Traffic, IMPACT ID: USC-LANDER/DARPA_2009_DDoS_attack-20091105/rev4383 . Traces taken 2009-11-05 to 2009-11-05. Provided by the USC/LANDER project (http://www.isi.edu/ant/lander).*

[49] *FRGP (www.frgp.net) Continuous Flow Dataset, IMPACT ID: USC-LANDER/FRGP_NTP_Flow_Data_anon-20131201/rev7965 . Traces taken 2013-12-01 to 2014-02-28. Provided by the USC/LANDER project (http://www.isi.edu/ant/lander).*

[50] *Scrambled Internet Trace Measurement dataset, IMPACT ID: USC-LANDER/DoS_DNS_amplification-20130617/rev5529 . Traces taken 2013-06-17 to 2013-06-17. Provided by the USC/LANDER project (http://www.isi.edu/ant/lander).*

[51] IANA, *Root servers*, https://www.iana.org/domains/root/servers, Accessed: 2023-07-04. [Online]. Available: https://www.iana.org/domains/root/servers.

[52] IANA, *Service name and transport protocol port number registry*, https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml, Accessed: 2023-07-04. [Online]. Available: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.

[53] A. Donoho, B. Roe, M. Bodlaender, *et al.* "UPnP device architecture 2.0." Accessed: 2023-07-04. (2020), [Online]. Available: https://openconnectivity.org/upnp-specs/UPnP-arch-DeviceArchitecture-v2.0-20200417.pdf.

[54] OASIS. "Web services dynamic discovery (WS-Discovery) version 1.1." Accessed: 2023-07-04. (2009), [Online]. Available: https://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html.

[55] A. Maghsoudlou, O. Gasser, and A. Feldmann, "Zeroing in on port 0 traffic in the wild," in *Proceedings of the 2021 Passive and Active Measurement Conference*, Cottbus, Germany: Springer, 2021. DOI: 10.1007/978-3-030-72582-2_32.

[56] F-Secure. "Linux/adore." Accessed: 2023-07-04. (2023), [Online]. Available: https://www.f-secure.com/v-descs/adore.shtml.

[57] Speed Guide, Inc. "Port 65535 details." Accessed: 2023-07-04. (2023), [Online]. Available: https://www.speedguide.net/port.php?port=65535.

[58] B. Krebs. "Six charged in mass takedown of DDoS-for-hire sites." Accessed: 2023-07-04. (2022), [Online]. Available: https://krebsonsecurity.com/2022/12/six-charged-in-mass-takedown-of-ddos-for-hire-sites/.

[59] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, *et al.*, "Spark: Cluster computing with working sets.," *HotCloud*, vol. 10, no. 10-10, p. 95, 2010.

A

Example Fingerprint in JSON Format


```
{
    "key": "7f5f4199504b689b585a33d030d348c3",
    "tags": [
        "UDP",
        "Amplification attack",
        "DNS amplification attack",
        "NTP amplification attack"
    ],
    "target": "42.42.42.42",
    "time_start": "2023-01-01T00:00:01",
    "time_end": "2023-01-01T00:00:10",
    "duration_seconds": 10,
    "total_packets": 13,
    "total_megabytes": 92,
    "avg_bps": 55102902,
    "avg_pps": 1.3,
    "avg_Bpp": 7417698,
    "peak_bps": 171106304,
    "peak_pps": 2,
    "peak_Bpp": 10796032,
    "total_ips": 6,
    "normal_traffic": {
        "total_packets": 2,
        "total_megabytes": 3,
        "avg_bps": 2516582,
        "avg_pps": 2,
        "avg_Bpp": 2385920,
        "peak_bps": 19087360,
        "peak_pps": 1,
        "peak_Bpp": 1572864,
        "destination_ports": "random",
        "source_port": {
            "80": 1
        },
        "protocol": {
            "TCP": 1
        }
    },
    "attack_vectors": [
        {
```

```
"service": "DNS",
"protocol": "UDP",
"source_port": 53,
"fraction_of_attack": 0.538,
"nr_packets": 7,
"nr_megabytes": 33,
"time_start": "2023-01-01T00:00:02",
"duration_seconds": 9,
"avg_bps": 30758229,
"avg_pps": 0.77,
"avg_Bpp": 4943286,
"peak_bps": 85155840,
"peak_pps": 2,
"peak_Bpp": 7102464,
"destination_ports": {
    "80": 1
},
"ethernet_type": {
    "IPv4": 1
},
"dns_query_name": {
    "ddostheinter.net": 0.571,
    "root-servers.net": 0.429
},
"dns_query_type": {
    "ANY": 0.571,
    "NS": 0.429
},
"sources_ips": [
    "1.1.1.1",
    "8.8.8.8",
    "8.8.4.4"
],
"source_statistics": [
    {
        "ip": "1.1.1.1",
        "nr_packets": 3,
        "nr_bytes": 20675584,
        "avg_bps": 18378296,
        "avg_pps": 0.33
    },
    {
        "ip": "8.8.8.8",
        "nr_packets": 2,
        "nr_bytes": 7171072,
        "avg_bps": 6374286,
        "avg_pps": 0.22
    },
    {
        "ip": "8.8.4.4",
        "nr_packets": 2,
        "nr_bytes": 7068672,
        "avg_bps": 6283264,
        "avg_pps": 0.22
    }
],
```

```
        "source_ases": [
            "13335",
            "15169",
            "15169"
        ],
        "ases_statistics": [
            {
                "as": "13335",
                "nr_packets": 3,
                "nr_bytes": 20675584,
                "avg_bps": 18378296,
                "avg_pps": 0.33,
                "total_ips": 1
            },
            {
                "as": "15169",
                "nr_packets": 4,
                "nr_bytes": 14239744,
                "avg_bps": 12657550,
                "avg_pps": 0.44,
                "total_ips": 2
            }
        ]
    },
    {
        "service": "NTP",
        "protocol": "UDP",
        "source_port": 123,
        "fraction_of_attack": 0.462,
        "nr_packets": 6,
        "nr_megabytes": 48,
        "time_start": "2023-01-01T00:00:01",
        "duration_seconds": 10,
        "avg_bps": 49492787,
        "avg_pps": 0.6,
        "avg_Bpp": 10310997,
        "peak_bps": 171106304,
        "peak_pps": 2,
        "peak_Bpp": 10796032,
        "destination_ports": {
            "80": 1
        },
        "ethernet_type": {
            "IPv4": 1
        },
        "ntp_requestcode": {
            "42": 1
        },
        "sources_ips": [
            "216.239.35.0",
            "216.239.35.4",
            "94.198.159.10"
        ],
        "source_statistics": [
            {
                "ip": "216.239.35.0",
```

```
                    "nr_packets": 2,
                    "nr_bytes": 20703232,
                    "avg_bps": 16562585,
                    "avg_pps": 0.33
                },
                {
                    "ip": "216.239.35.4",
                    "nr_packets": 2,
                    "nr_bytes": 20046848,
                    "avg_bps": 16037478,
                    "avg_pps": 0.33
                },
                {
                    "ip": "94.198.159.10",
                    "nr_packets": 2,
                    "nr_bytes": 20764672,
                    "avg_bps": 16611737,
                    "avg_pps": 0.33
                }
            ],
            "source_ases": [
                "15169",
                "15169",
                "1140"
            ],
            "ases_statistics": [
                {
                    "as": "15169",
                    "nr_packets": 4,
                    "nr_bytes": 40750080,
                    "avg_bps": 32600064,
                    "avg_pps": 0.66,
                    "total_ips": 2
                },
                {
                    "as": "1140",
                    "nr_packets": 2,
                    "nr_bytes": 20764672,
                    "avg_bps": 16611737,
                    "avg_pps": 0.33,
                    "total_ips": 1
                }
            ]
        }
    ]
}
```