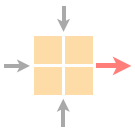
ETH zürich



Networked Systems
ETH Zürich — seit 2015

Exploring Morphing DDoS Attacks

Master Thesis

Author: Nikodem Kernbach

Tutor: Dr. Muoi Tran

Supervisor: Prof. Dr. Laurent Vanbever

October 2023 to May 2024

Acknowledgements

I want to thank Dr. Muoi Tran for the assistance in creating this master thesis, as well as for the opportunity to work on such an interesting topic. His helpful hints and calm attitude carried me through the writing of the thesis. Theo von Arx helped me during the first part of the thesis as a co-tutor and helped me to set up the working environment. Following the midterm presentation, Prof. Dr. Laurent Vanbever provided valuable input in a key discussion about the project's development. I also want to thank my friends, who took some time out of their busy schedules to help me by proofreading this thesis.

Abstract

In recent years, Distributed Denial of Service (DDoS) attacks have significantly evolved and became increasingly sophisticated. A novel form of these are morphing DDoS attacks, which evade standard defenses by dynamically changing their attack vectors. The next evolution step might be adaptive morphing DDoS attacks, which morph based on real-time data about the state of the target system. These attacks remain underexplored in literature.

This thesis addresses this gap by first developing a laboratory environment tailored for executing and analyzing these innovative attacks. It particularly facilitates experiments against various network configurations and defense mechanisms. The experiments demonstrated that the newly developed adaptive morphing DDoS attacks were highly effective across all tested environments and against all implemented defense mechanisms, including state-of-the-art ACC-Turbo and Jaqen. It also surpasses the performance of existing morphing attacks. Moreover, this thesis reveals weaknesses in the ACC-Turbo defense and proposes promising directions for enhancing this defense mechanism.

Contents

Chapter 1

Introduction

This thesis begins with an introductory chapter that establishes the scope of the research, reviews relevant literature, and articulates the project's objectives. Chapter Two delves deeper into the foundational concepts relevant to the study, offering a comprehensive discussion on existing Distributed Denial of Service (DDoS) attacks, defense mechanisms, and the various tools and datasets employed in this domain. The core of the thesis is encapsulated in Chapters Three and Four. Chapter Three details the improvement of the given experimental framework, which was crucial in supporting the execution and validation of the research experiments. Subsequently, Chapter Four presents the novel DDoS attack strategies devised during this study, providing an in-depth analysis of their unique implications and effectiveness. Chapter Five projects potential directions for future research, building upon the findings of this work. The concluding chapter summarizes the key contributions and accomplishments of the thesis, underscoring its improvements over existing DDoS attacks.

1.1 Motivation

In today's digital age, many of our daily activities take place online, reflecting our increasing reliance on Internet-based services. Although we often take the smooth operation of the Internet for granted, it is important to recognize the work that goes on behind the scenes to keep us safe online. Cyber Security professionals are constantly working to ensure not only the reliability of the Internet's infrastructure against hardware and software failures but also to defend it against malicious actors.

One particularly notorious threat are Distributed Denial of Service (DDoS) attacks, which are abundant and occur on a minute-by-minute basis, with large attacks exceeding 1 Tbps occurring almost weekly [13]. A DDoS attack involves using a network of compromised computers (botnet) to flood a targeted system with excessive traffic, disrupting its operations and accessibility for legitimate users [1]. Any digital platform could fall victim to such an attack. This is especially true because DDoS services can be easily acquired on the dark web, even by malicious actors with limited technical knowledge and resources [9].

Complementary to DDoS attacks, there are DDoS defenses, which undergo continuous improvement and refinement. Both are subjects of active research and study. This leads to a perpetual arms race between malicious actors and defenders, with the malicious often being a step ahead. Recently, a new sophisticated kind of DDoS attacks has become more prevalent: Morphing DDoS attacks [12]. These attacks are characterized by their ability to dynamically adapt and evade traditional mitigation measures, posing a challenge to existing defenses. Despite their increasing occurrence,

limited research has been conducted on these attacks [2]. However, understanding their operation is crucial for developing effective defense strategies.

1.2 Thesis Objective

This thesis aims to establish a flexible lab environment with various network configurations and defense methods. Different DDoS attacks and defenses will be compared across diverse network scenarios. Insights into the performance of attacks and defenses will be gained through end-to-end measurements, including latency and data drop rates. Based on this understanding, new adaptive attack strategies will be developed, prioritizing effectiveness. The aim of this research is to contribute to the strengthening of defenses and the protection of digital systems by providing insights derived from the conducted experiments.

1.3 Assumptions

For this research, two fundamental assumptions have been established that significantly shape the experimental framework and methodology. The first concerns the attacker's capabilities, assuming that they can send arbitrary packets directly to the targeted system. This model bypasses conventional Internet routing, allowing the attacker to alter any packet headers, including destination IP addresses, at will. Such capabilities enable the crafting of packets that are maximally disruptive to the targeted system. This assumption is crucial, as it establishes a rigorous testing environment for evaluating the robustness of DDoS defenses under extreme attack conditions.

Secondly, it is assumed that the attacker receives real-time feedback on their attacks' latency and packet drop rates. This feedback is crucial since it enables the attacker to change their tactics adaptively. By understanding the immediate impact of their attacks, the attacker can make informed decisions, allowing for adjustments to the attack vectors to optimize disruption. These assumptions are vital for simulating sophisticated attack scenarios and are instrumental in thoroughly evaluating the proposed morphing DDoS attacks.

1.4 Related Work

As highlighted earlier in Section 1.1, despite their growing prevalence, research into morphing DDoS attacks remains limited, although their existence and impact are widely recognized. Nevertheless, some key studies and tools have emerged, which are relevant to the development of this thesis.

A prominent example is the tool "SODA" (Simulation of DoS Attacks) [8], developed by F5 Inc. and presented at Black Hat USA 2019 [12]. SODA utilizes machine learning techniques for anomaly detection and Quality of Service (QoS) monitoring, which are designed to ensure operational stability under attack conditions. This tool represents a significant advancement in enhancing the robustness of defense mechanisms against dynamic attack vectors.

Another significant contribution is the defense mechanism ACC-Turbo [2], specifically designed to counter "pulse-wave morphing DDoS attacks". These attacks feature short, high-intensity traffic bursts that vary in characteristics, posing a substantial challenge to mitigation techniques. The effectiveness of ACC-Turbo was analyzed using the packet-level simulation tool NetBench [5], which is further explored in Section 2.5 and was developed as part of a Master Thesis at ETH Zurich [6].

While specific frameworks such as SODA and ACC-Turbo address dynamic and morphing DDoS threats, the concept of adaptive morphing attacks—those that respond in real-time to changes in network conditions like latency and packet drop rates—remains underexplored. This thesis aims

to bridge this gap by developing an analysis framework that investigates these types of attacks and proposing novel adaptive attack strategies. By doing so, it contributes to the advancement of defense mechanisms capable of countering these sophisticated and dynamically evolving threats.

While this thesis primarily focuses on adaptive morphing DDoS attacks, it is essential to acknowledge the broad and active research community addressing DDoS attacks in general. Traditional attacks are further explored in Section 2.1.

Chapter 2

Background

This chapter provides insight into existing DDoS attacks and defense mechanisms. It offers an overview of the evolution of these attacks, highlighting their increasing sophistication. An example is the emergence of morphing DDoS attacks, which dynamically alter their tactics to evade conventional defenses. The chapter also reviews datasets containing DDoS attack data. Furthermore, the simulation approach employed by ACC-Turbo, which utilizes NetBench, is discussed. This serves as a foundation for the investigations presented later in this thesis.

2.1 Discussion on DDoS Attacks

DDoS attacks have been a significant problem since the 1990s and, since then, have become increasingly complex. Generally speaking, they can be divided into two main types. Network Layer (Layer 3/4) attacks overwhelm the network infrastructure of a targeted system, with SYN flood attacks being a typical example. Application Layer (Layer 7) attacks exploit weaknesses in the applications running on the system to cause a shutdown, such as HTTP GET floods [12]. The term 'Distributed' in DDoS indicates that these attacks often involve multiple sources, typically the nodes of a botnet controlled by the attacker, aiming to flood the target with overwhelming traffic [1]. An exemplary taxonomy of DDoS attacks, focusing on Network Layer attacks, is illustrated in Figure 2.1 taken from the work of Sharafaldin et al. [11]. These attacks are categorized into reflection and exploitation attacks. In reflection attacks, the attacker manipulates a third party, like a public DNS server in DNS reflection attacks, to send unwanted traffic to the target. In exploitation attacks, the attacker directs a large volume of traffic from his botnet straight at the victim.

As DDoS attacks have evolved, combining various methods for enhanced effectiveness has proven to be effective. The standard attacks outlined in Figure 2.1 can act as 'building blocks' for morphing DDoS attacks, which rapidly alternate between these methods to avoid detection by defense systems. Since these attacks morph quickly, defense systems' response times are crucial. The advent of software-defined switches, which can be reconfigured as the systems are running, enabled this [4]. They are utilized by advanced defense mechanisms such as Jaqen [7] and ACC-Turbo [2], where ACC-Turbo achieves mitigation times as fast as sub-second. For the purposes of this thesis, we differentiate between two types of morphing DDoS attacks:

- **Dynamic/Classic Morphing DDoS Attacks**: These attacks frequently change tactics, posing significant detection and mitigation challenges to defense systems.
- **Adaptive Morphing DDoS Attacks**: They also change tactics frequently but additionally

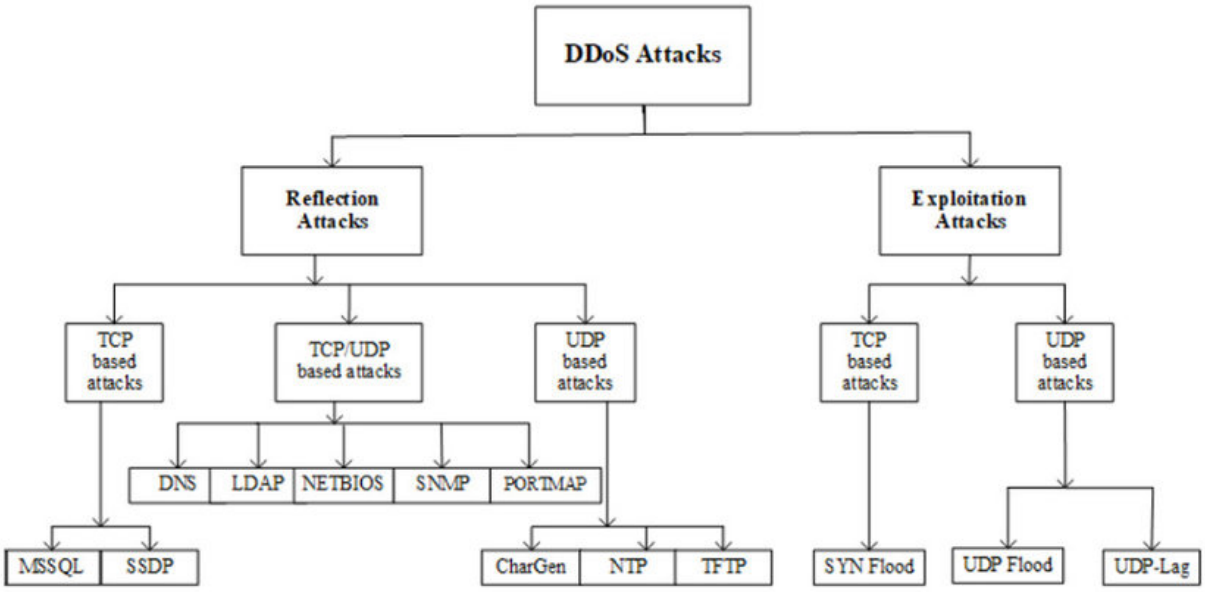


Figure 2.1: DDoS Attack Taxonomy.
Source: Figure 1 from [11]

utilize (potentially delayed) real-time data about the target's network conditions, giving them an advantage.

Side channels or other exploits delivering feedback in real-time are potential sources for data used by adaptive morphing attacks, though research in this area remains limited. This thesis concentrates on adaptive morphing DDoS attacks, exploring their development, mechanisms, and effectiveness. The goal is to enhance defensive strategies against these sophisticated threats.

2.2 Datasets

Two publicly available datasets, which provide DDoS attack data are discussed.

2.2.1 CIC-DDoS 2019

The CIC-DDoS2019 dataset [11], developed by the Canadian Institute for Cybersecurity, is an exhaustive resource for analyzing DDoS attacks and testing detection systems. It covers a variety of attacks. The data was collected using a laboratory setup that included both an 'Attack network' and a 'Victim network' to closely mimic real-world conditions. The setup scheme can be seen in Figure 2.2. The dataset is available in raw PCAP format as well as in CSV files, where network flows have been pre-filtered, facilitating easier analysis. A drawback of this dataset is that it does not provide PCAP traces which contain only benign data for reference. Moreover, the attacks are easily detectable by the source IP address, as all malicious traffic was fed through the same interface to the recording switch. This is also visible in Figure 2.2.

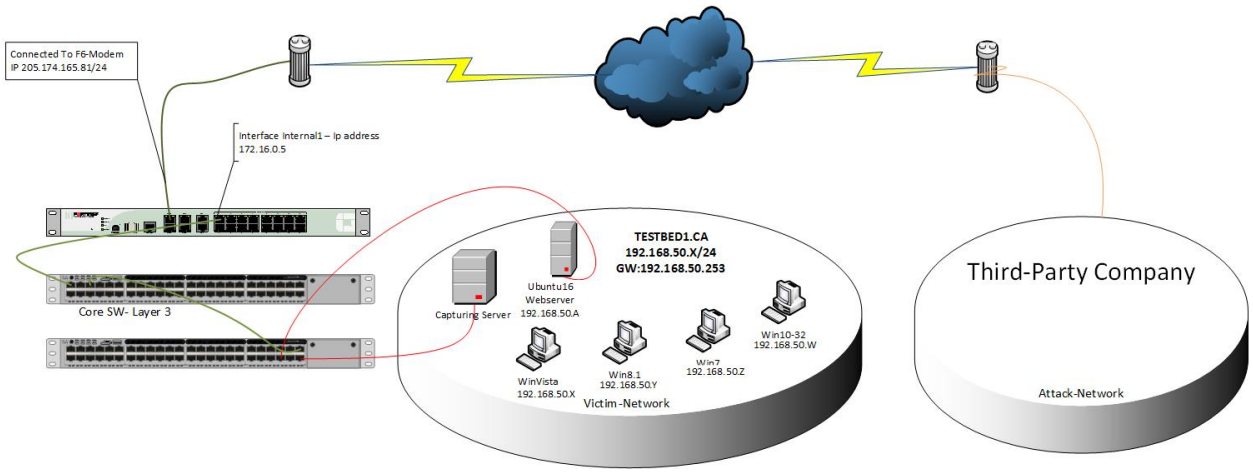


Figure 2.2: Testbed Architecture.
Source: Figure 2 from [11]

2.2.2 CIC-IDS 2017

Similar to the previous dataset, the CIC-IDS2017 dataset [10] is also created by the Canadian Institute for Cybersecurity and is the predecessor of the former [11]. The setup and evaluation is very similar, as well as the employed attacks. However, there is one particular benefit which is important to this thesis: It contains data traces without a DDoS attack. This is important, as this data is later used as a source of benign background traffic in this thesis.

2.3 ACC-Turbo Defense

ACC-Turbo, as proposed in [2], is an innovative DDoS defense, designed to mitigate pulse-wave DDoS attacks, but applicable to any volumetric attacks. The novelty of ACC-Turbo is its capability to run at line rate, mitigating attacks in real-time (≤1s reaction time). To achieve this, ACC-Turbo uses online clustering and programmable scheduling. Online clustering is based on the idea, that one can observe traffic aggregates. Each incoming packet is assigned to one of a predefined number of clusters, based on its features which can be arbitrary packet header fields. Both the number of clusters, as well as the choice of features are parametrized, although limited by hardware capabilities. The authors of ACC-Turbo provide an implementation in P4 for the Tofino 1 programmable switch, which works with 4 clusters and 4 features. Distances between packets and clusters are determined using a fast, linear version of the manhattan distance. For nominal features, a distance of 1 is assumed if the feature does not match, while for ordnial features, the numeric difference is taken. Programmable scheduling is done offline in the control plane. The control plane continuously polls information about the clusters from

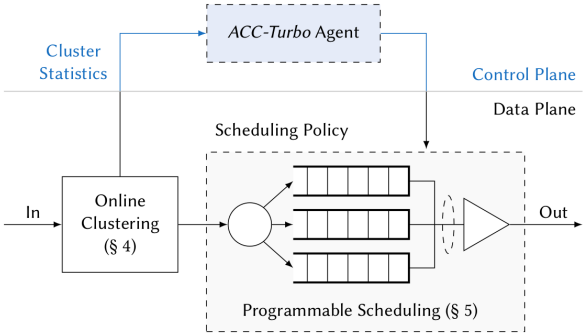


Figure 2.3: ACC-Turbo Architecture.
Source: Figure 4 from [2]

the data plane and determines if there are clusters which are probably part of an attack. This is determined by a ranking algorithm which might be based on the throughput, packet rate and/or size of each cluster. This algorithm maps the clusters to a priority queue which therewith deprioritizes clusters which are probably part of an attack. The whole approach is depicted in Figure 2.3. ACC-Turbo was evaluated using both, a hardware-based and a simulation-based approach. In the hardware-based evaluation, ACC-Turbo was compared to Jaqen [7] and stood out especially by its fast reaction time and a more generic approach. On the other hand, ACC-Turbo suffered from slightly higher benign packet drops, once an attack was mitigated. The simulation-based evaluation showed ACC-Turbo's capabilities beyond the current limitations of commodity hardware. The result was, that using a higher number of clusters and features, the results can be further improved, which will become possible with new generations of commodity hardware soon. Also, a better distance function might be used, if hardware permits. The limitations of ACC-Turbo lie in two main points. The first is, that ACC-Turbo can only prevent volumetric attacks. Secondly, ACC-Turbo assumes, that attacks are detectable by similarity of packets, i.e., they form traffic aggregates. An attacker could break similarity at packet level or at aggregate level.

The simulation-based approach was performed using a framework based on NetBench [5], which was also used and adapted for this thesis. However, the authors' simulations were only done based on a couple of predefined flows, or based on PCAP files preprocessed by a Python script. This approach had to be altered for the purpose of this thesis, as is thoroughly discussed in Section 3.3.1.

ACC-Turbo's most convincing advantages, are the capability to work at line-rate and the genericity of the approach, however, as mentioned by the authors, signature-based defenses employed in parallel could improve the overall defense capabilities even further. One design consideration was to define a distance of 1 for non-matching nominal features. However, this value is chosen arbitrarily and has no relation to the range calculations of ordinal features. Thus, although it seems to work well in practice, more research would be required to find out whether the influence of nominal parameters gets marginal because of this decision or not. Moreover, as for example the spread of packet length is bigger than the one of IP addresses, the former has a bigger influence on the clustering result. As ACC-Turbo does only work for volumetric attacks and traffic aggregates, it is a good idea to use it together with other DDoS defenses, which are designed for low traffic and spread attacks.

2.4 Jaqen Defense (Heavy Hitter)

The basic concept behind the Jaqen defense is that traditional defense methods either require expensive, rigid hardware or slow software solutions, which are not suitable for widespread use. The Jaqen system integrates both detection and defense functions within programmable switches, avoiding the need for extra hardware. It works directly in-line with the traffic flow. Jaqen can both identify and counter threats right in the switch, using a system that is fully built into the switch's regular operations, allowing for an immediate reaction to DDoS threats. This system also offers a versatile API that allows for the creation of varied defense tactics and efficient management of network resources. It employs sophisticated data structures and algorithms that are specially designed for the limited space of programmable switches. A resource manager dynamically assigns detection and defense modules throughout the network depending on the severity and type of the attack, adjusting quickly to any changes. The study shows that Jaqen can handle large-scale attacks very effectively, performing significantly better than current systems. It works at high speeds and can swiftly and effectively manage complex, changing attacks. However, while programmable switches

offer many advantages, they require significant upfront investment in hardware and expertise to implement and maintain such a system. Also, for now, their capabilities are limited as they were developed only recently [2]. The fact, that defenses are still able to achieve such good results, even with only limited resources, is impressive and proves the importance of software-defined switches.

For the purpose of this thesis, similarly as in previous research [2], only one module was further used to examine the new proposed attacks. This is the Jaqen heavy hitter module, which aims to mitigate volumetric attacks, such as elephant flows. This is further discussed in Section 3.3.2.

2.5 NetBench Packet Simulator

NetBench [5] is a packet simulator originally designed for assessing routing in static topologies through discrete event simulation. It simulates physical network components such as links, output ports and network devices and provides a means to simulate custom network configurations and works with arbitrary data. This setup allows for a versatile testing environment where researchers can evaluate various protocols, and other systems and their impact on network performance. NetBench was first used by the authors of ACC-Turbo [2] to build a framework that simulated attacks against their new defense and facilitated the analysis and evaluation thereof. Similarly, this thesis used NetBench to simulate different attacks against different defenses in different network configurations. NetBench was a good baseline, however the attacks and defenses had to be implemented specifically to work with this tool. Moreover, a PCAP extractor had to be additionally implemented. A first version was implemented during the work on ACC-Turbo and the further refined as explained in Section 3.2.

Chapter 3

Design

This chapter explains the setup with which the evaluation was performed, the improvements made to the framework and the configurations used.

3.1 Overview

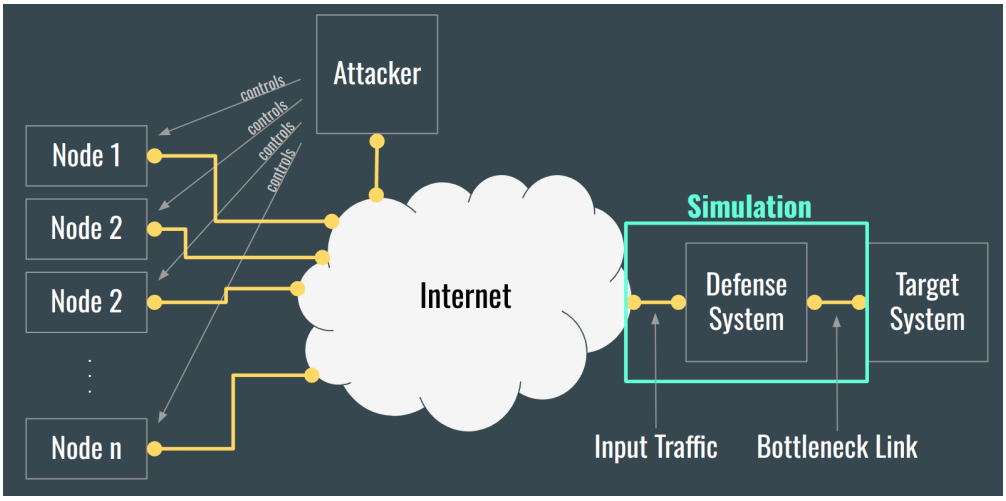


Figure 3.1: Typical simplified DDoS scenario with outlined simulated part in this thesis.

A simplified, typical DDoS attack scenario can be found in Figure 3.1. The attacker controls a botnet of nodes. The attacker, the nodes and the target system are all connected via the Internet. At the attacker's command, the nodes can all direct traffic towards the target, trying to overwhelm the defense system and bottleneck link. This whole structure however is hard to simulate, as the Internet is a dynamic and huge structure. Since the goal of this thesis was to simulate different attacks against different defenses under different network configurations, the elements marked in Figure 3.1 were simulated. This still allows for realistic results, as long as the input traffic can be well represented. The natural starting point for this was the setup used by the authors of ACC-Turbo [2], which used a framework based on NetBench [5]. In particular, the following elements needed to be simulated:

- The input traffic, consisting of malicious traffic and benign (background) traffic.

- The defense systems such as ACC-Turbo and Jaqen.
- The bottleneck link with the congestion it experiences.

The NetBench setup from ACC-Turbo served as a good baseline, however some changes needed to be made. As the aim was to perform the experiments in a framework which is as realistic as possible, the first design decision was to utilize public datasets to construct the input traffic for the experiment. These usually provide packet traces in PCAP format, which our framework needed to support. The packets could then be extracted and fed into the simulation. The given framework had only limited PCAP support, so it needed to be extended. The work on this is described in Section 3.2.

Next, the defense systems needed to be in place. The setup already featured an implementation of the ACC-Turbo defense, however this implementation did not support clustering based on real packet features like addresses and port, but only on so-called flow IDs. This is due to the fact, that originally, either preprocessed PCAPs, or simplified flow representations were used. Also, there was no NetBench implementation of the Jaqen defense, so it had to be implemented. The work done on this part is described in Section 3.3.

Finally, the framework had to be prepared for analysis and evaluation. This required to parameterize the simulation runs and provide multiple network configurations. As already mentioned in Section 1.3, one crucial assumption was that the attacker gets feedback about the drop rate and latency of the packets send by them. This needed to be enabled in order to implement real adaptive morphing DDoS attacks. These improvements are further described in Section 3.4.

3.2 Input Traffic

The main goal of input traffic design is to ensure that our evaluations are as realistic as possible. The ideal scenario is to simulate real connections from benign machines to the target system in real-time, and then for the attack period from hijacked botnet machines as well. However, this requires a lot of resources. Initial attempts to model input traffic were based on flow-level data, which proved insufficient for the dynamic nature of DDoS attacks. Instead, recorded traces of such traffic can be used, which is a better compromise. In Section 2.2 two datasets containing such traces were presented. They provide the data in PCAP format. In order to use this data for our simulation, the used NetBench framework had to be adapted. The significant component which needed to be rewritten is the extractor which converts PCAP traces into NetBench packets. A comparison of the original and the adapted one is portrayed in Figure 3.2.

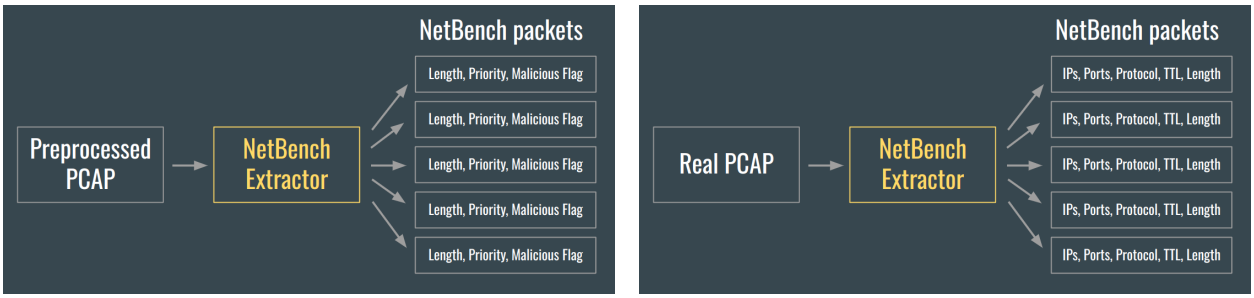


Figure 3.2: Comparison of PCAP extractors used in ACC-Turbo (left) and this work (right).

The original extractor worked on preprocessed PCAPs that were obtained using a Python script, which also contained the ACC-Turbo defense. The original PCAP was read by the script,

run against the defense and repackaged for NetBench. Then, NetBench was used to analyze the trace and create evaluation data.

For higher flexibility and in order to eliminate unnecessary framework dependencies, the decision was made to enable NetBench to handle real PCAPs directly. This was achieved by rewriting the PCAP extractor. The packet features usually important for a DDoS defense are the IP addresses, ports, TTL, protocol and packet length. All of these were obtained and saved in the NetBench packets. From there, the defenses can use them to filter the traffic.

In general, traffic can be categorized into two main types: malicious traffic and benign traffic, which is also called background traffic. Ideally, separate sources for such traffic are required. First experiments were run using the CIC-DDoS 2019 dataset (see Section 2.2.1), but it does not contain traces of benign and malicious traffic separately. Separating this data from only one trace is possible in general, but leads to unwanted anomalies. For example, there might be less benign traffic as it is suppressed by an ongoing attack, or there might be a surplus of TCP reset packets, as the connections are reset. Thus, another source had to be found. This was also important, as later new adaptive morphing DDoS attacks were created, for which the malicious data is generated locally. For the benign traffic, the CIC-IDS 2017 dataset (see Section 2.2.2) was used. In this dataset, DDoS attacks were simulated, but the authors included one day of simulated data traffic for reference. For the purpose of this work, initially 100 100-second traces were extracted, to simulate different background traffic profiles. For the final analysis however, 1000-second traces were extracted and used. This defined the runtime of the experiments to always be 100 seconds and 1000 seconds respectively.

For the malicious traffic, the decision was made to generate it locally. This allowed to include arbitrary and custom attacks. The exact process is further explained in the evaluation chapter (Section 4.2). There, also the distinction between attack traffic and probing traffic (both of them being malicious) is made (see section 4.4).

3.3 Defense Implementation

Next in the framework stack are the defense systems. Two systems were used for the analysis: ACC-Turbo and Jaqen. The given framework already had an implementation of ACC-Turbo in NetBench, but it was not able to handle the real packet features, as it was only used with preprocessed data. However, it did feature a Python version of ACC-Turbo, which was ported to NetBench, as explained in Section 3.3.1. Similarly, for Jaqen, there was no NetBench version available, but there was a version available in the P4 language, used to program software-defined switches [3]. This version was used as inspiration for the implementation of the Jaqen defense in NetBench, developed for this thesis and explained in Section 3.3.2.

3.3.1 ACC-Turbo

ACC-Turbo was already mentioned in Section 2.3. Basically, the ACC-Turbo defense clusters incoming traffic at line rate. In order to achieve this, a fast clustering search approach is used. This approach needs a metric to evaluate the distance between packets and clusters. Each incoming packet is then compared to the current clusters and assigned to the closest cluster. This means, that if the new packet is close, but not inside the cluster, the cluster grows. Each cluster is assigned a priority by the ranking algorithm, whenever the control plane is called. Between these calls, the priorities stay as they are. The incoming packet receives the priority assigned by the ranking algorithm and is put into a priority queue.

The biggest shortcoming of the previously given NetBench implementation is the lack of ability to compare and cluster packets according to the real packet features. The packet features used are divided into two categories: *ordinal* features and *nominal* features. The following 13 features are used:

- ***Ordinal* Features**
  - Source and destination IP address (each byte is one feature)
  - TTL
  - Packet length
- ***Nominal* Features**
  - Source and destination port
  - Protocol

The categorization is based on the fact, that ordinal features indicate the proximity of packets, while nominal ones do not. Given two packets p^1, p^2, the distance d between two feature values p_i^1, p_i^2 (of the same feature i) is calculated as

$$d_i(p^1, p^2) = \begin{cases} |p_i^1 - p_i^2| & \text{if } i \text{ is } \textit{ordinal} \\ \begin{cases} 0 \text{ if } p_i^1 = p_i^2 \\ 1 \text{ if } p_i^1 \neq p_i^2 \end{cases} & \text{if } i \text{ is } \textit{nominal} \end{cases}$$

Then, the distance between two packets is calculated as

$$D(p^1, p^2) = \sum_i d_i(p^1, p^2)$$

Now, packets also need to be compared to clusters. A cluster C can be defined as $C = (C_1, C_2, \cdots C_n)$, where C_i is either a range $[l_i, u_i]$, or a bloom filter $(\text{add}_i(), \text{check}_i())$. The bloom filter has thus the ability to add an element and to check if an element was already observed. Therewith, the distance d_i between the cluster C and the packet p for one feature i can be calculated as

$$d_i(C, p) = \begin{cases} \begin{cases} p_i - u_i \text{ if } u_i < p_i \\ l_i - p_i \text{ if } p_i < l_i \\ 0 \text{ else} \end{cases} & \text{if } i \text{ is } \textit{ordinal} \\ \begin{cases} 0 \text{ if } & \text{check}_i(p_i) \\ 1 \text{ if not } & \text{check}_i(p_i) \end{cases} & \text{if } i \text{ is } \textit{nominal} \end{cases}$$

Lastly, to compare a full cluster C to a packet p one has to calculate

$$D(C,p) = \sum_i d_i(C,p)$$

In summary, we can determine a packet's p closest cluster C^* from a set of clusters $\{C^1, C^1, \cdots C^n\}$ with the formula

$$C^* = \underset{C^i \in \{C^1, C^2, \ldots, C^n\}}{\arg\min} D(C^i, p)$$

Once we assigned the packet to a cluster, the cluster might need to be expanded. Per feature this can be done as follows

$$C_i^{\text{new}} = (C_i \bullet p_i) = \begin{cases} [\min(l_i, p_i), \max(p_i, u_i)] & \text{if } i \text{ is } \textit{ordinal} \\ (\text{add}_i(), \text{check}_i()) & \text{and call } \text{add}_i(p_i) \text{ if } i \text{ is } \textit{nominal} \end{cases}$$

Giving us the full formula

$$C^{\text{new}} = (C \bullet p) = (C_1 \bullet p_1, C_2 \bullet p_2, \cdots, C_n \bullet p_n)$$

As a fast clustering search approach is used, the comparison of two clusters does not need to be calculated. However, the calculations follow directly from the ones above. This was implemented in NetBench. The bloom filter data structure was implemented by using Java's internal *hashCode()* function and instantiated with a *boolean* array of size 1024. The ranking algorithm was left as it was, ordering the clusters in priority according to the number of packets in each cluster. This means that clusters with a lot of packets were deprioritized compared to clusters with few packets. The control loop re-estimating the priorities was run every 0.1 seconds.

The following parameters specific to ACC-Turbo were customizable per NetBench simulation run in the new implementation:

- Number of clusters
- Per cluster buffer size (in number of packets)
- Size of bloom filter used for nominal features

In the evaluation, two variants of ACC-Turbo were used. One was instantiated with 5 clusters and one with 10 clusters. Note, that the total amount of buffer space had to be divided across all clusters. Thus, each cluster received either a fifth, or a tenth of the available buffer. All 13 available features were always used.

3.3.2 Jaqen Heavy Hitter

Similarly to ACC-Turbo, no NetBench implementation of Jaqen was given. However, a P4 program was given, on which the new implementation was based on. More specifically, two Jaqen heavy hitter configurations were implemented. Jaqen heavy hitters use an internal counting bloom filter to count occurrences of observed packets based on a hash function. The input to the hash function determines the features on which the defense matches. For the purpose of this thesis, two versions were implemented, one matching on the source IP address only, and one matching on the 5-tuple of

source and destination IP addresses, source and destination ports and the protocol. The counting bloom filter size was set to 1024 in all cases.

The defense works as follows: Depending on the version either the source address or the 5-tuple of each incoming packet is hashed and put into the counting bloom filter. This is done at line rate. Every n seconds, the control plane is invoked and all the bloom counter entries are compared to a fixed threshold. If a counter exceeds this threshold, all packets matching this counter will from now on be blocked. At the next timeout, the counters are re-evaluated. If a blocked counter did not exceed the threshold within the last n seconds, it is freed again. The counters are reset to 0 at each timeout. This defense aims to protect the target system against so-called elephant flows, which can be malicious.

Similar to the ACC-Turbo implementation, the counting bloom filter was implemented using Java's internal *hashCode()* function. The following parameters specific to Jaqen were made customizable per NetBench simulation run in the new implementation:

- The timeout between control plane calls
- The threshold
- Size of the counting bloom filter

Especially the threshold is a hyperparameter which requires to be hard-coded but influences the result greatly. Choosing the right value is a fine line and can be challenging. For the purpose of our evaluations, the threshold was set by running benign traffic traces without attacks and setting the threshold such that no packets are dropped. This way, benign traffic drops are minimized at times without attacks.

Configurations Used in the Evaluation

Four different configurations were used in the evaluation to ensure the evaluation is as comprehensive as possible. The counting bloom filter size was always set to 1024. The other parameters can be found in Table 3.1.

Table 3.1: Jaqen Heavy Hitter Configurations.

Configuration	Timeout	Threshold
Source IP-based Slow	4.5 seconds	15000
Source IP-based Fast	1.5 seconds	5000
5-tuple based Slow	4.5 seconds	15000
5-tuple based Fast	1.5 seconds	5000

3.4 Environment Setup and Configuration

There are three main components of each simulation run. The attack, the defense and the environment. Additionally also the background traffic can be exchanged. As the goal of designing the framework was to enable a comprehensive analysis and evaluation, multiple network configurations had to be supplied. Moreover, ways to analyze the outcomes of singular simulation runs had to be developed. A simplified view of the framework's inner working can be seen in Figure 3.3. Most and foremost, a way to facilitate adaptive morphing DDoS attacks had to be found. Secondly,

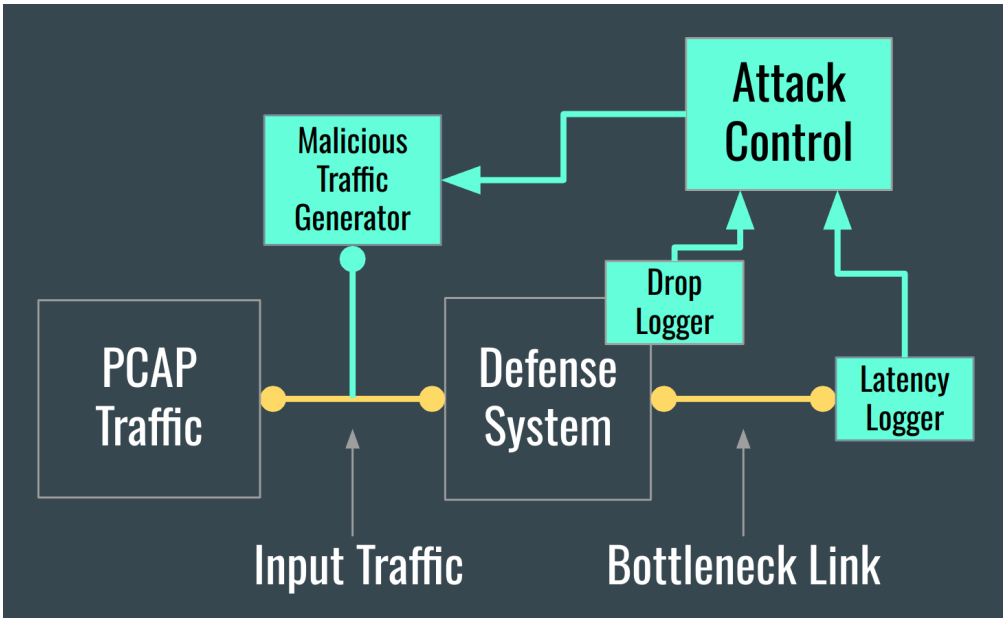


Figure 3.3: A schematic overview of the framework's attack control.

the simulation run had to be logged in order to create visualizations and perform analysis later. Finally, all the remaining other parameters had to be fine-tuned. These three aspects are explained in the following subsections.

3.4.1 Attack Facilitation

To facilitate the later implementation of adaptive morphing DDoS attacks, the necessary arrangements had to be made. This was enabled by designing a new attack control structure. The core module is a new "Attack Control" class, in which new attacks can be implemented. It contains a *handler()* function which is called every n milliseconds. This value is configurable and enables for the implementation of morphing attacks. Usually, this was set to 1 second. The attack control also contains handling functions which are called by the drop and latency loggers whenever a packet on the bottleneck link is seen. Since only feedback of malicious packets should be visible to the attacker, the loggers make sure that only these trigger the event. The loggers do however log all packets for later analysis and for creating visualizations. Another important element is the malicious traffic generator. This generator was implemented to facilitate the development of custom attacks. Both probing and conventional attack traffic can be arbitrarily generated. The attack control was parameterized with the attack traffic rate. At each NetBench run, the maximal allowed rate can be appointed. Usually, this rate is set to 500 megabits per second. Probing traffic of adaptive morphing DDoS attacks is excepted from this throughput cap. However, the rate of the probing traffic is rather low. Usually, 320 probing packets with a maximum length of 1500 bytes are sent per second, which gives a maximal rate of 480 kilobytes per second.

3.4.2 Network and Run Configurations

The performed simulations were run against multiple network configurations. The network configurations include the throughput of the bottleneck link, as well as the buffer size of the link and the delay. In the presented setup, the delay is not important, as nothing happens to the traffic after

it passed the link, except logging. The 4 configurations were defined as shown in Table 3.2. They generally revolve around two dimensions: throughput and buffer size.

Table 3.2: Network configurations.

Configuration name	Throughput in Mbps	Buffer Size
ENV1	50	50 packets
ENV2	50	200 packets
ENV3	200	50 packets
ENV4	200	200 packets

It has to be noted that for ACC-Turbo, the buffer is divided across all clusters equally. The resulting sizes per cluster are presented in Table 3.3.

Table 3.3: Sizes of Buffers per Cluster when Running ACC-Turbo.

Total Buffer Size	Buffer Size per Cluster (5 Clusters)	Buffer Size per Cluster (10 Clusters)
50 packets	10 packets	5 packets
200 packets	40 packets	20 packets

The NetBench framework is accompanied by a lot of different shell, gnuplot and Python configuration scripts, which are not explained here, but can be found in the affiliated code repository. This enabled for efficient batch runs which were parallelized for even more efficiency.

3.4.3 Packet Logging

To properly promote a scientific work, great visualizations are needed. As preparation for these, a complete packet log is saved at each run. This log is then used to analyze the run and evaluate it in a broader context. A short slice of a real exemplary packet log can be found in Table 3.4.

Table 3.4: Exemplary Packet Log.

Enqueue Time	Arrival Time	Latency	Drop?	Length	FID	Mal?	DefDrop?	Prob?
30191999000	30195413000	3414000	false	320	0	false	false	false
30192002000	30195419400	3417400	false	12000	0	false	false	false
30195651000	-1	-1	true	23680	0	false	false	false
30192244000	30195659400	3415400	false	12000	0	false	false	false
30192354000	30195899400	3545400	false	320	0	false	false	false
30192435000	30195905800	3470800	false	23680	0	false	false	false
30196035000	-1	-1	true	23680	0	false	false	false
30196041000	-1	-1	true	320	0	false	false	false
30196162000	-1	-1	true	320	0	false	false	false

The enqueue and arrival times are provided in nanoseconds. Their difference yields the latency. In case the packet was dropped, no arrival time and latency is provided. The drop is also additionally stated in its own column. The length is provided in bits. The FID, which stands for flow ID was used for initial tests, but was overthrown after implementing the PCAP extractor. The last three

fields display whether the packet is malicious, whether it was dropped by the defense or inversely because the buffer was full and if the packet is a probing packet. The second to last field however can only be determined if the Jaqen defense is used, as ACC-Turbo does always cluster and prioritize packets, even in times of no attack. This enables for further analysis of runs against the Jaqen defense, as it is visible whether Jaqen actually mitigated an attack, or if the buffer was simply overflowing.

Chapter 4

Constructed Attacks & Evaluation

This chapter presents the implemented attacks and analyzes their performance and effectiveness. Unless otherwise specified, all attacks were performed at a rate of 500 megabits per second.

4.1 Dataset Attacks

First, before examining adaptive morphing DDoS attacks in detail, it was necessary to establish a baseline using standard attacks. The initial experiments involved replaying PCAP traces from the CIC-DDoS 2019 dataset against various defenses. However, these experiments did not yield conclusive results because there was not enough benign traffic in the traces. The dataset contains captures of executed attacks, but benign traffic is mostly dropped during successful attacks. Consequently, the presence of benign traffic was too sparse to allow for thorough analysis. To illustrate this, two 100-second traces are included in Figure 4.1 as evidence of the dataset's limitations. Especially note the absence of blue data points whenever there was no benign traffic within that second.

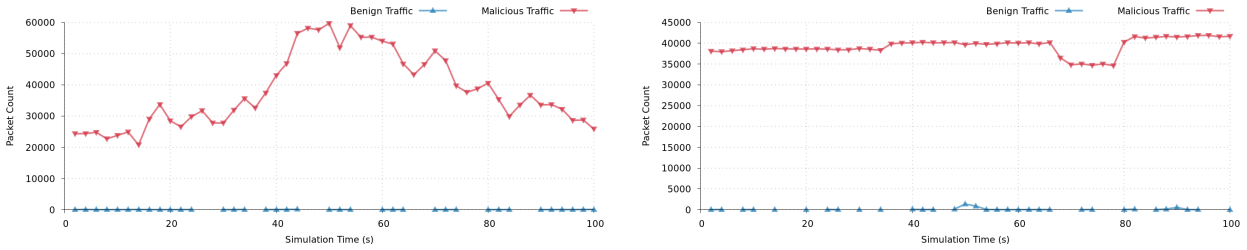


Figure 4.1: Exemplary malicious and benign packet counts of two 100-second traces from the CIC-DDoS 2019 dataset. The attack types are an UDP flood (left) and a SYN flood (right). Missing points mean there were no packets at all within the given second.

This led to the decision to obtain benign and malicious traffic separately. The obtaining of benign traffic was previously described in Section 3.2. The following section will describe the generation of malicious traffic.

4.2 Traffic Generation

While benign traffic was obtained from datasets, the use of recorded malicious traffic from datasets was overthrown. Instead, malicious traffic was generated manually. As previously described in Chapter 3, the NetBench-based framework was expanded to accommodate seven essential packet

features: source and destination IPs, source and destination ports, protocol, TTL, and packet length. This foundation was crucial for constructing later attacks. The initial step involved creating a random packet generator that could produce packets with these features within specified realistic ranges. The ranges selected are visible in Table 4.1.

Table 4.1: Range of packet features.

Feature	**Range**
IP Source and Destination Addresses	0.0.0.0 to 255.255.255.255
Source and Destination Ports	0 to 65535
TTL (Time to Live)	25 to 64
Protocol	UDP (17) or TCP (6)
Packet Length (including headers)	200 to 1500 bytes

The packet length was selected based on the typical Maximum Transmission Unit (MTU) of 1500 bytes on most Ethernet links. The minimum packet size was pragmatically set to avoid performance issues in early framework development stages, as smaller packets would require handling a greater packet volume. The TTL was set with an arbitrary upper limit. In retrospect, utilizing the full range from 0 to 255 would have been more appropriate. At that time, the maximum was set to 64, aligning with the default TTL used by UNIX devices. Since packets usually traverse only a few nodes, the minimum was set to 25. This configuration allowed for random traffic generation, serving as a building block for more advanced traffic sampling in subsequent experiments.

4.3 Baseline Attacks

First, some already known attacks had to be replicated to establish a baseline for the upcoming sophisticated attacks. For each attack, two exemplary runs are shown against *ENV1* and *ENV4*, as defined in Section 3.4. It has to be noted again, that *ENV1* represents a very difficult scenario, where there is not even enough bandwidth for the benign traffic itself.

4.3.1 No Attack

Of course, the framework can also be run without any attack with benign (background) traffic only. For completeness, the characteristics of running the two environments *ENV1* and *ENV4* (see chapter 3.4) without an attack are provided in Figure 4.2. Each line represents one run against a different defense system. The data points are the aggregated values for 10 seconds of the simulation each. In the *ENV4* configuration the traffic doesn't experience any drops and link bandwidth utilization is low. It looks equal for all defenses - as there is no need to drop packets, everything works as expected. In the challenging *ENV1* however, packet drops occur with every defense. This is caused due to the burstiness of the benign traffic trace. As a traffic burst arrives, the buffer fills up and some traffic is dropped. During a short period of time, the bandwidth is utilized at 100%. However, this cannot be seen on the plot, as each data point is an aggregate of 10 seconds of traffic. A special case is the ACC-Turbo defense, experiencing smaller bandwidth utilization, while also suffering from higher benign data drop rates. This is caused by the fact, that although ACC-Turbo has an equal buffer space when compared to the other defenses, the available buffer space per cluster is lower. Under load, singular clusters might clog up, activating the defense.

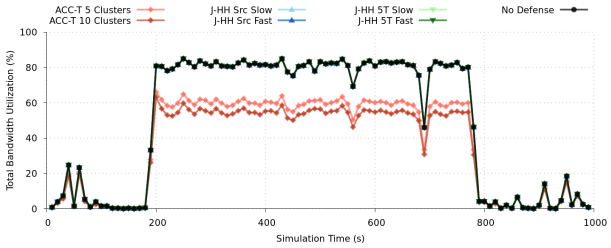
This would be desirable in the case of an attack, but since here we only have benign traffic, it is actually harmful. This is also why the 10 cluster version has even more benign drops and less total bandwidth utilization than the 5 cluster version. Because it can hold even less packets in the buffer per cluster. To be more precise, in *ENV1*, only 5 packets fit in the buffer of a single cluster of the 10 cluster version and 10 packets in the 5 cluster version. In this specific traffic scenario, it is especially painful, as there are a lot of benign packets with high length.

4.3.2 UDP Flood Attack

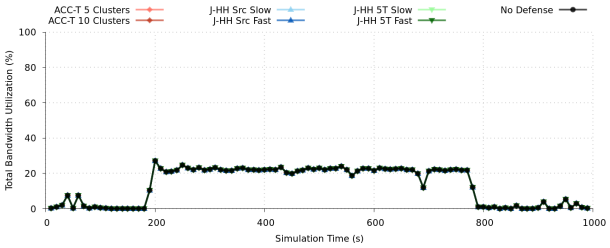
This is the simplest volumetric attack. This attack was implemented by sampling one random packet as described in Section 4.2 and sending it repeatedly at the configured attack rate. In other words, all attack packets share the same packet features. The characteristics of running this attack can be seen in Figure 4.3. It is clearly visible, that the attack is successful in both environments if no defense is employed. Moreover, the Jaqen heavy hitter defense configurations handle the attack with ease. For ACC-Turbo, the bandwidth utilization is at its maximum. This is to be expected, as ACC-Turbo only starts to drop packets as soon as the available bandwidth is exhausted. However, we see that ACC-Turbo struggles way more to defend this attack. It still manages to let through roughly 50% of benign traffic, which for a general purpose defense is still great, but compared to Jaquen's performance it is well behind. An exception is the 5 cluster version in the *ENV4* configuration, which successfully mitigates the attack. If we take a deeper look however, we can find the problem in the initial cluster distribution. It turns out, that there is only 1 (or 2 in the 10 cluster case) clusters which take in packets of high length. Combined with the low buffer sizes, and the bursty nature of the benign traffic, this is detrimental. The other half of benign packets, which are of low length, pass the defense just fine.

4.3.3 Classic Morphing Attack

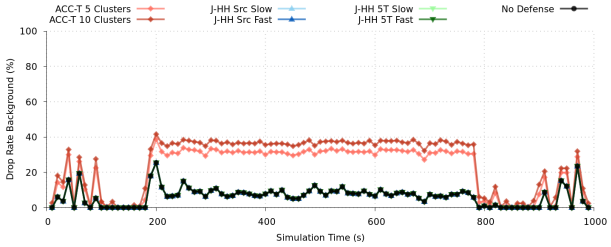
A classic morphing attack is an attack which dynamically morphs over time. For the purpose of comparability all morphing attacks simulated in this work morph once every 5 seconds. This means that this attack will change shape every 5 seconds. At the beginning of each such phase, a random packet is sampled as described in Section 4.2 and repeatedly sent at the attack rate, until the next phase. A representation thereof can be seen in Figure 4.4. The characteristics of running this attacks can be seen in Figure 4.5. Compared to the last attack, one can immediately notice the difference in appearance of the Jaqen graphs. The effect observed here is the delay Jaqen needs to recognize the new attack's signature and block it. If we look closer at the graphs from the last UDP flood attack, we can notice that the first Jaqen datapoint is always an outlier. This is exactly the same effect as here, because Jaqen needs time to figure out the attack. Another important realization is, that in general the fast versions of the Jaqen defenses react faster and thus have lower benign drop rates. The typical zigzag shape results from the length of the morphing attack phases (5 seconds) being offset from the length of the Jaqen phases (1.5 seconds for the fast versions and 4.5 seconds for the slow versions). This is an absolutely realistic scenario. The ACC-Turbo defense behaves similarly to the last attack, behaving better and worse at times. Depending on the cluster the attack lands in (one of the many clusters that contain short packets, or the few which contains long packets), it causes more or less collateral damage by deprioritizing benign traffic. Generally speaking however, ACC-Turbo performs better when faced with this attack. This is expected, as ACC-Turbo was originally developed as a defense against pulse-wave DDoS attacks, which are quite similar to this attack.



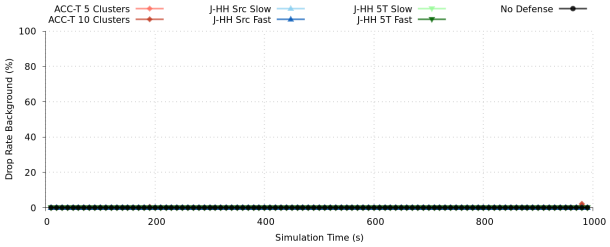
(a) *ENV1* bandwidth utilization.



(b) *ENV4* bandwidth utilization.

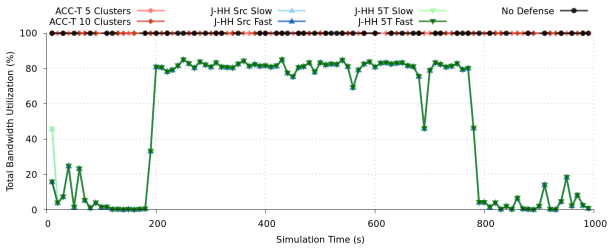


(c) *ENV1* benign packet drops.

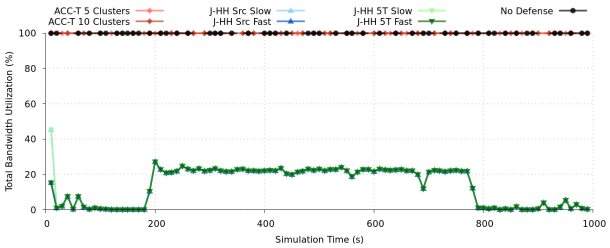


(d) *ENV4* benign packet drops.

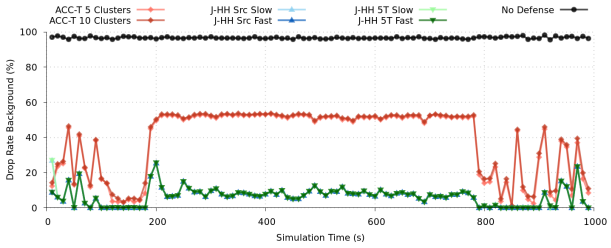
Figure 4.2: No attack run characteristics.



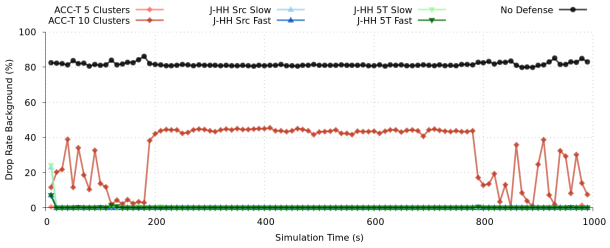
(a) *ENV1* bandwidth utilization.



(b) *ENV4* bandwidth utilization.



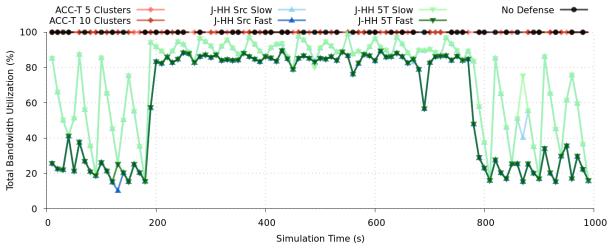
(c) *ENV1* benign packet drops.



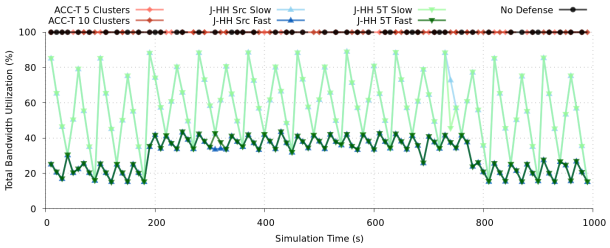
(d) *ENV4* benign packet drops.

Figure 4.3: UDP flood attack run characteristics.

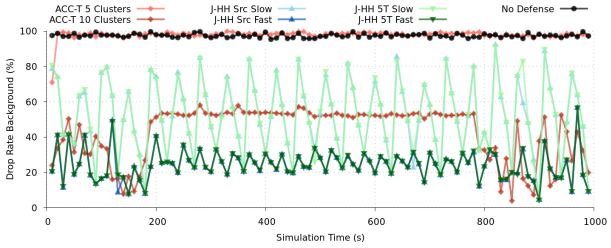
Figure 4.4: Classic Morphing Attack.



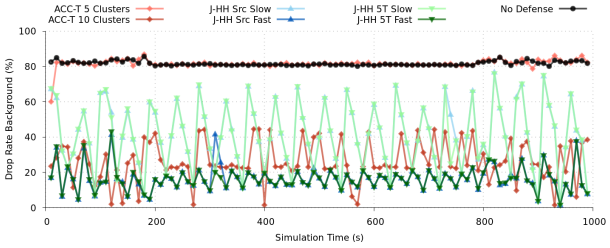
(a) *ENV1* bandwidth utilization.



(b) *ENV4* bandwidth utilization.



(c) *ENV1* benign packet drops.



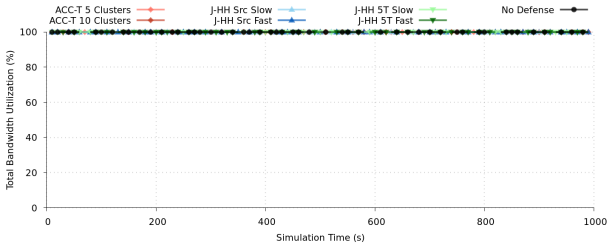
(d) *ENV4* benign packet drops.

Figure 4.5: Classic morphing attack run characteristics.

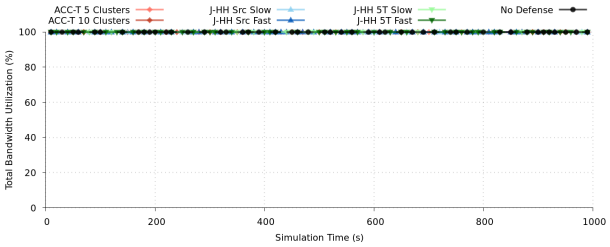
4.3.4 Random Attack

This attack is the exact opposite to the flood attack. It is performed sending attack traffic at the attack rate, where each and single packet is completely randomized using the method explained in Section 4.2. This attack is ideal to combat signature-based defenses such as heavy hitters, completely circumventing them. However, this attack is not ideal in general, as the ideal attack would consist of a flood of benign-looking packets, not random packets. For clustering approaches, it opens the door for reverse defenses. While usually, the traffic consists of several aggregates, one of which might be malicious, here we have some aggregates swimming in a flood of noise. The clustering algorithm needs to be prepared to detect the clusters in this noise and distinguish them from it. Here, the limit is the amount of noise sent by the attacker. The run characteristics are depicted in Figure 4.6. As expected, the bandwidth utilization is through the roof for all defenses. When looking at the benign packet drops, most of the rates closely follow the black line, which depicts the no defense run and therewith the level at which a defense can be declared completely defeated. However, there is a surprise regarding ACC-Turbo. It is the only defense that reaches benign drop rates below the 90% line. This has an interesting reason. A lot of benign packets are quite short, while the random packets have random lengths. In the setups presented, there are a lot of clusters for long packets, while small benign packets are funnelled through a specific cluster. This creates a loophole for very small benign packets, as the chance of a generated attack packet to

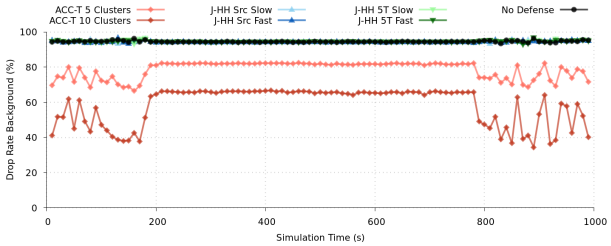
have such a small length is very low. Even though this is better than completely failing to protect benign traffic, it must be underlined, that the drop rates are still around 60% and 80% respectively, which would be considered as insufficient.



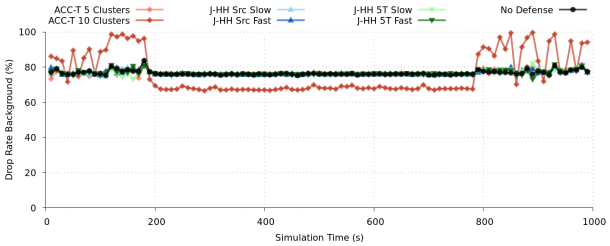
(a) *ENV1* bandwidth utilization.



(b) *ENV4* bandwidth utilization.



(c) *ENV1* benign packet drops.



(d) *ENV4* benign packet drops.

Figure 4.6: Random attack run characteristics.

4.4 Adaptive Morphing Attacks

Adaptive morphing attacks are attacks which can take advantage of real-time feedback about the targeted system. Specifically, it was assumed that for each packet the attacker sends towards the targeted system, either the latency, or the drop of this packet would be reported back to them. The attainability thereof is discussed in Chapter 5. The assumption was made that the attacker experiences a delay of two seconds when obtaining this data. While the original assumption was that the attacker can obtain feedback for all packets sent by themselves, in the following attacks, actually only data from a few packets is needed. The distinction between attack traffic and probing traffic was introduced:

- Attack traffic: The volumetric traffic that is meant to overwhelm the targeted system
- Probing traffic: The traffic that was sent to learn about the real-time state of the target system. In contrast to the volumetric attack traffic, only few packets of this kind are sent. The following attacks always send a packet rate of 320 packets per second. The packets however have different lengths.

This reduces the complexity in making such attacks possible. An attacker who is running a distributed attack from a botnet might use some of their nodes to maintain benign-looking connections to the target system and send the probing packets from these to avoid detection. The exact probing traffic differs from attack to attack and is explained accordingly. The two following presented attacks are the final versions and the culmination of the search process.

4.4.1 Adaptive Source Morphing Attack

This is the first out of two attacks which withstood the test of time and is included in the final evaluation. It attacks the source IP address. The source address is split into 4 features, each byte being looked at individually. Each feature ranges in values between 0 and 255. This range is split in 8 so-called 'buckets', as can be seen in Table 4.2. In total, we get 32 'groups' (i.e. feature-bucket combinations). The probing traffic is then sampled as follows: Per second, 10 probing packets are sampled per group (320 in total). Each probing packet is sampled completely at random, as explained above. Then, according to the group, the feature is replaced with a random value from within that bucket. All other features remain untouched. The packet is then labelled through a distinct group ID, which NetBench keeps track of. These probing packets are not all sent at once, but spaced out as far as possible (to not by accident fill any buffer). Two seconds after they are sent, the attacker gets either the information that the packet was dropped, or the latency of the packet. Since this is again a morphing attack, traffic is morphed every 5 seconds. For each group ID, the attacker keeps the estimated average latency and drop rate. This is then reset every 5 seconds.

Table 4.2: Structure of Source Buckets.

Feature	**Bucket 0**	**Bucket 1**	$\cdots$	**Bucket 6**	**Bucket 7**
Source IP Byte 0	0-31	32-63	$\cdots$	192-223	224-255
Source IP Byte 1	0-31	32-63	$\cdots$	192-223	224-255
Source IP Byte 2	0-31	32-63	$\cdots$	192-223	224-255
Source IP Byte 3	0-31	32-63	$\cdots$	192-223	224-255

At this time, the attacker decides on one bucket per feature, which looks most promising to attack. This is determined by the drop rate. The bucket with the lowest drop rate wins. This is done for all the 4 features. Then, a new packet is sampled for the actual volumetric attack, determining all features except the source IP for the next 5 seconds. In each attack packet sent out at the attack rate, the values of the 4 features are randomly replaced by values from their corresponding buckets.

An example: After sending out the probing traffic, one ends up with the distribution seen in Table 4.3. Then, each attack packet for the next 5 seconds has the same destination IP address, ports, TTL, packet length, and protocol value, but the source IP address is different, chosen at random from the values in Table 4.3.

Table 4.3: Structure of Source Buckets.

Feature	Chosen Bucket	Value Range
Source IP Byte 0	Bucket 3	96-127
Source IP Byte 1	Bucket 7	224-255
Source IP Byte 2	Bucket 0	0-31
Source IP Byte 3	Bucket 6	192-223

Since morphing happens every 5 seconds, but the feedback has 2 seconds delay, the data obtained is overlapping between the last attack and the second-to-last, as shown in Figure 4.7.

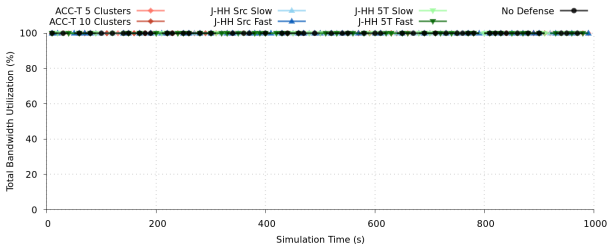
Figure 4.7: Time frame of feedback data considered for morphing.

The run characteristics can be seen in Figure 4.8. Neatly to say, all defenses are rendered completely useless against the attack, as all lines follow the black line which is the reference for having no defense at all. The only characteristic that stands out, is the slightly lower benign drop rate of ACC-Turbo in the first 10 seconds of the attack. This is further discussed in Section 4.6.

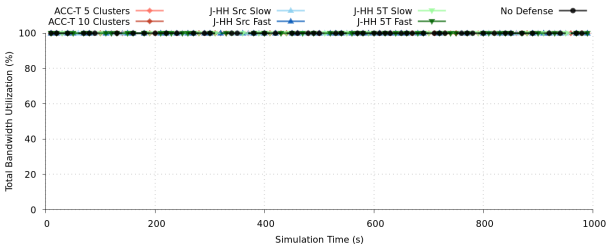
4.4.2 Adaptive Full Morphing Attack

Even though the last attack was highly successful, another attack was developed, aiming to unlock the full potential of adaptive morphing DDoS attacks. In essence, this attack is very similar to the last one. However, not only the source IP is determined adaptively, but also all other ordinal features. Specifically, these are the following 10 features: source IP (each byte), destination IP (each byte), TTL and packet length. The ports are not suitable, as their proximity does not relate to a theoretical proximity between the data. They are often assigned arbitrary. Also the probing traffic was obtained in a similar fashion as in the last attack.

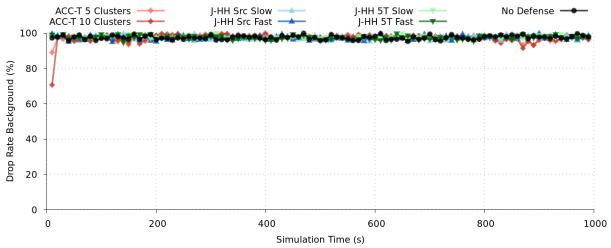
There are 10 features, each divided into 8 buckets. Per second, 4 probing packets are sent per feature and bucket, to keep it consistent with the last attack. Both send 320 probing packets per second. However, since there now are 10 features and 8 buckets, 80 groups are used. The probing packets were obtained exactly as in the last attack, i.e., sampled randomly and independently



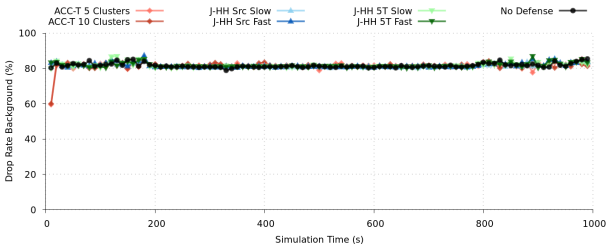
(a) *ENV1* bandwidth utilization.



(b) *ENV4* bandwidth utilization.



(c) *ENV1* benign packet drops.



(d) *ENV4* benign packet drops.

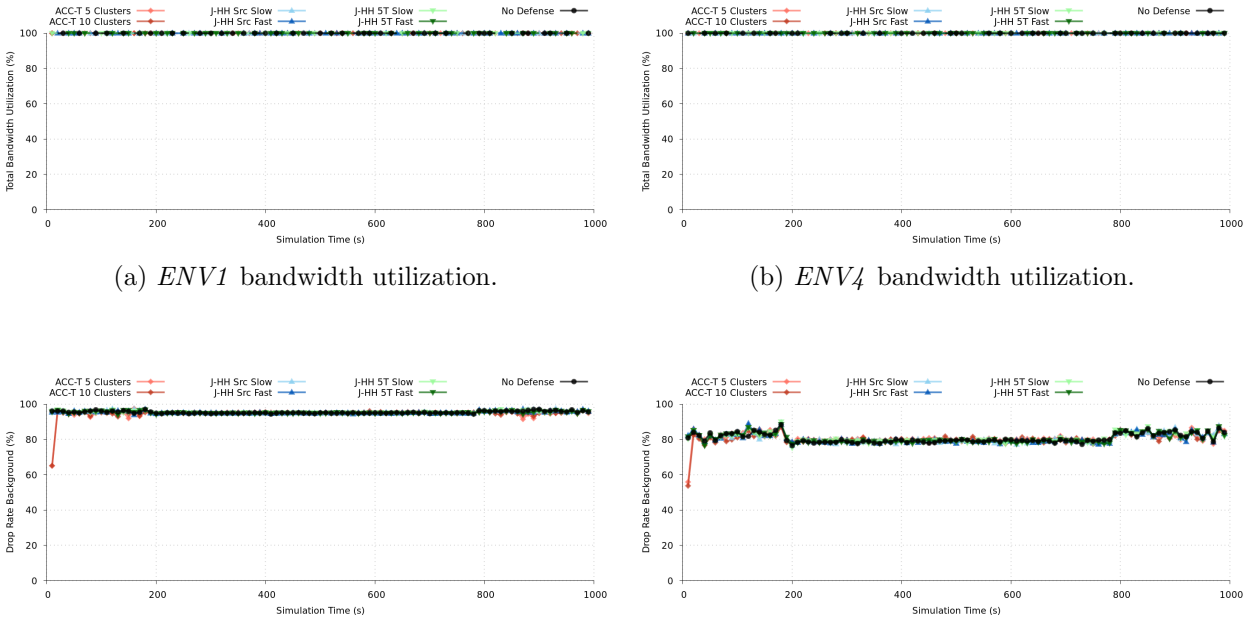
Figure 4.8: Adaptive source morphing attack run characteristics.

of each other and then only the assigned feature would be substituted by a number out of the according range. For completeness, the structure is depicted in Table 4.4.

Table 4.4: Structure of All Buckets.

Feature	**Bucket 0**	**Bucket 1**	···	**Bucket 6**	**Bucket 7**
Source IP Byte 0	0-31	32-63	···	192-223	224-255
Source IP Byte 1	0-31	32-63	···	192-223	224-255
Source IP Byte 2	0-31	32-63	···	192-223	224-255
Source IP Byte 3	0-31	32-63	···	192-223	224-255
Destination IP Byte 0	0-31	32-63	···	192-223	224-255
Destination IP Byte 1	0-31	32-63	···	192-223	224-255
Destination IP Byte 2	0-31	32-63	···	192-223	224-255
Destination IP Byte 3	0-31	32-63	···	192-223	224-255
TTL	0-31	32-63	···	192-223	224-255
Packet Length	0-186	187-374	···	1125-1311	1312-1500

The attack traffic is then constructed tn the exact same way as in the adaptive source morphing attack. After each 5 seconds, a prototype packet is sampled, then for each attack packet, the 10 features are substituted by values of the group with the lowest drop rate for each feature. The attack is performed at an attack rate of 500 megabits per second. The run characteristics can be found in Figure 4.9. The discussion is similarly short to the one of the adaptive source morphing DDoS attack: All defenses fail at protecting against this attack.



(a) *ENV1* bandwidth utilization.

(b) *ENV4* bandwidth utilization.

(c) *ENV1* benign packet drops.

(d) *ENV4* benign packet drops.

Figure 4.9: Adaptive full morphing attack run characteristics.

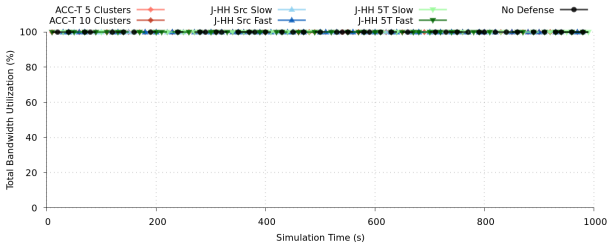
4.5 Analysis of Adaptive Full Morphing Attack

The most advanced and promising attack is the last one presented. Now follows a further analysis of its advancements.

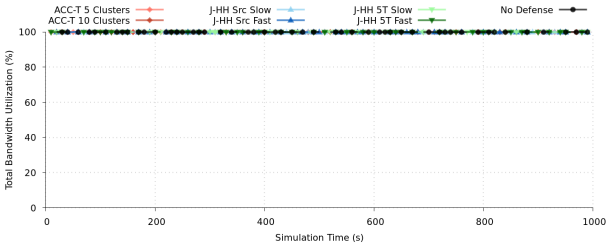
4.5.1 Evaluating Effectiveness against All Defenses and Environments

The effectiveness of the attack was already seen in Section 4.4.2. To confirm these findings, the attack has been tested against all environments, all defenses, and using two different background traffic traces. The result is that it is effective against all defenses and environments with a very good rate, staying at the same level as the no defense attack. It is crucial to emphasize that the value of interest here is not the absolute drop rate but rather its relative value compared to the baseline scenario, which utilizes a random dropping approach (no defense scenario). The result of the comprehensive analysis can be found in Figures 4.10 and 4.11. There are two outliers to this rule, namely Figures 4.10h and 4.11c. Overall, out of the 16 runs performed against the ACC-Turbo defense (4 environments, 2 traffic traces, 2 ACC-Turbo variants), ACC-Turbo managed to (partly) overcome the defense twice. This calls for taking a closer look. By looking at the bandwidth share between benign (background) and attack traffic for these two cases, we cannot see anything unusual (see Figure 4.12). We see that the attack was successful in monopolizing on the bandwidth, maintaining over 90% of it. So how is it possible to get a drop rate of benign traffic of only 50%? The answer is that in both of these cases, all clusters but one were clustering for large packets. These clusters would catch all of the attack traffic, as well as benign traffic. But the one cluster specializing for very small packets remained untouched and managed to let the small packets of the benign traffic through. This didn't result in the defender freeing up the bandwidth from the attack, but it did cause a high absolute number of benign packets to come through. Why was the adaptive attack not able to detect this cluster and attack it? Since the cluster is very specific, and the adaptive attack does only search groups of each feature independently, it has

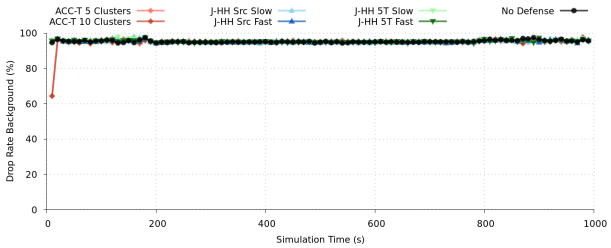
a blind eye for small clusters, compared to big ones. This is one direction for possible further improvement: Develop an adaptive morphing DDoS attack which does not estimate the drop rate for each feature independently, but in relation to each other, such that smaller clusters cannot be found. One limitation thereof however, is the need for more probing packets, which might affect the practicability of such attacks.



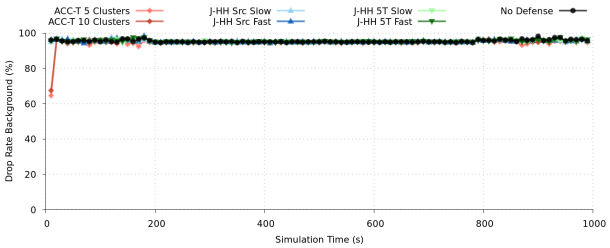
(a) *ENV1* bandwidth utilization, trace 1.



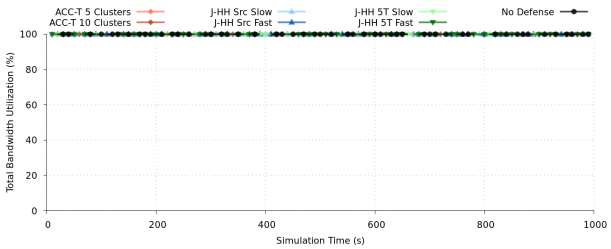
(b) *ENV2* bandwidth utilization, trace 1.



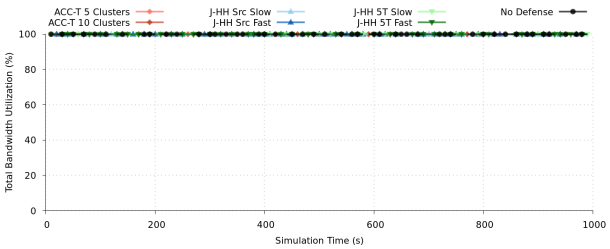
(c) *ENV1* benign packet drops, trace 1.



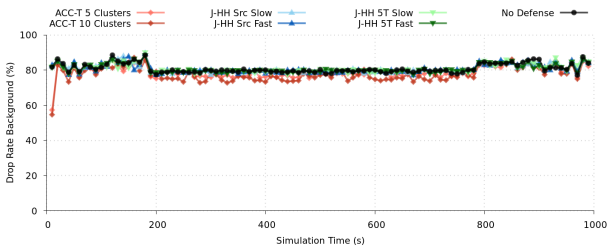
(d) *ENV2* benign packet drops, trace 1.



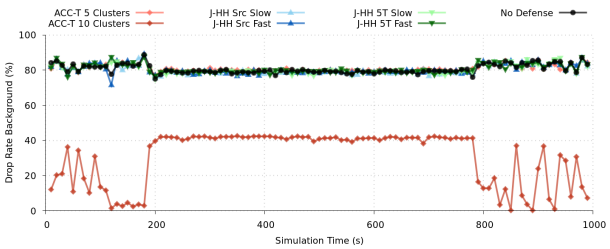
(e) *ENV3* bandwidth utilization, trace 1.



(f) *ENV4* bandwidth utilization, trace 1.

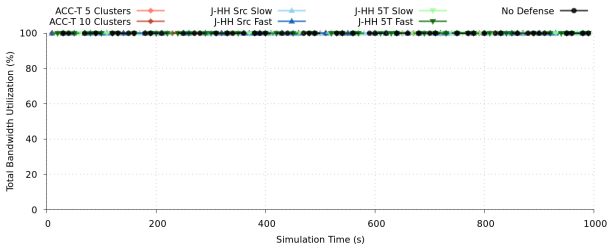


(g) *ENV3* benign packet drops, trace 1.

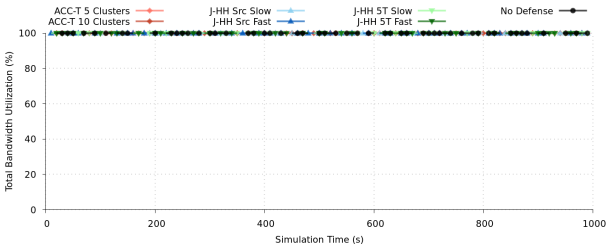


(h) *ENV4* benign packet drops, trace 1.

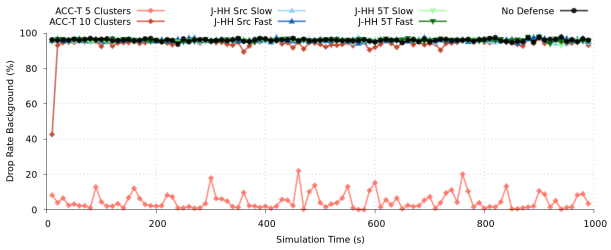
Figure 4.10: Adaptive full morphing attack against all environments and defenses and two background traffic traces, part 1.



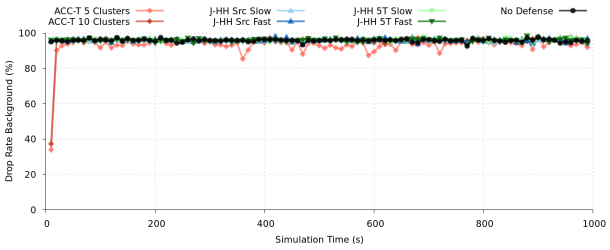
(a) *ENV1* bandwidth utilization, trace 2.



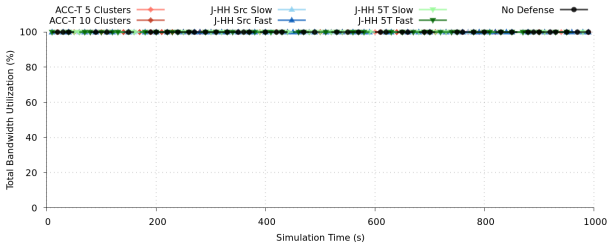
(b) *ENV2* bandwidth utilization, trace 2.



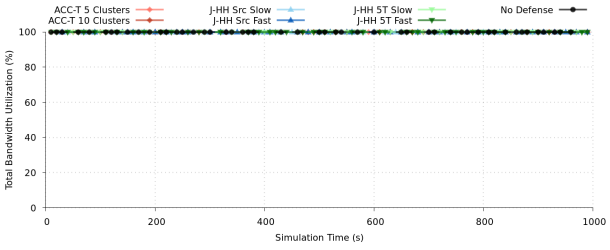
(c) *ENV1* benign packet drops, trace 2.



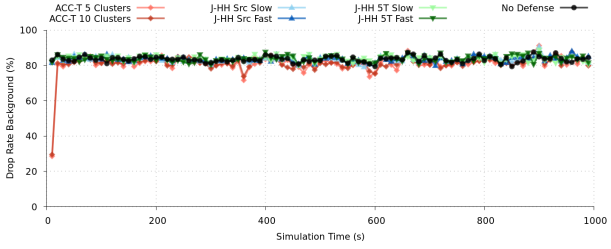
(d) *ENV2* benign packet drops, trace 2.



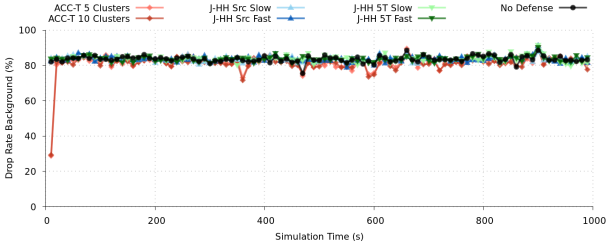
(e) *ENV3* bandwidth utilization, trace 2.



(f) *ENV4* bandwidth utilization, trace 2.



(g) *ENV3* benign packet drops, trace 2.

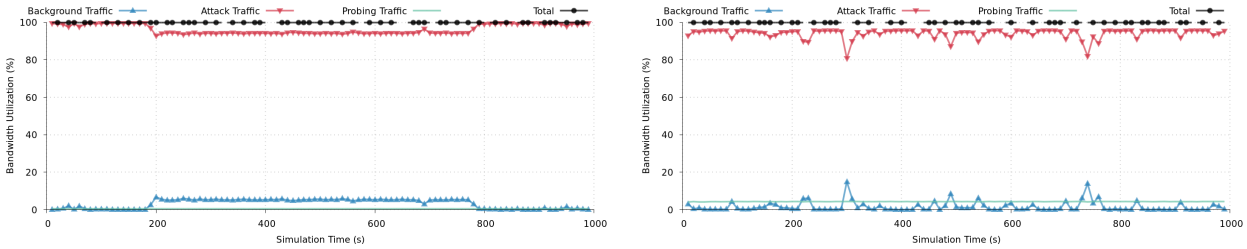


(h) *ENV4* benign packet drops, trace 2.

Figure 4.11: Adaptive full morphing attack against all environments and defenses and two background traffic traces, part 2.

4.5.2 Comparing Effectiveness to Other Attacks

Next, the attack was compared to the other attacks. From the introduction of all attacks and the discussion thereof, it is clear that the attack is more effective than the other attacks, while using the same attack rate of 500 megabit per second. The plots created to prove this point were used



(a) *ENV4* bandwidth share, trace 1.

(b) *ENV1* bandwidth share, trace 2.

Figure 4.12: Adaptive full morphing attack against two chosen cases to take a closer look at the bandwidth distribution.

to introduce the attacks, so they are not included again. While even the basic attacks, such as classic morphing attacks and random attacks could defeat Jaqen defenses, ACC-Turbo was harder to beat. Finally, ACC-Turbo showed the potential to be able to not fully mitigate the attack, but at least performed better than having no defense at all. For now, it is left to randomness whether ACC-Turbo is successful or not, but further discussion can be found in Section 4.6.

4.6 Possible Improvements of ACC-Turbo

It has to be noted that we used the 'vanilla' version of ACC-Turbo to evaluate our attacks. This means, that the clusters are empty before running the simulation (i.e., are not predefined) and that fast clustering search was used. Also, the clusters are never reset, despite them growing all the time. During the research done in Section 4.4.1, it was noticed that the attack had some success in the very beginning (first 10 seconds), only then falling back. This brought the idea of resetting the clusters. The clusters were reset every 10 seconds and then allowed to grow naturally. The result can be seen in Figure 4.13.

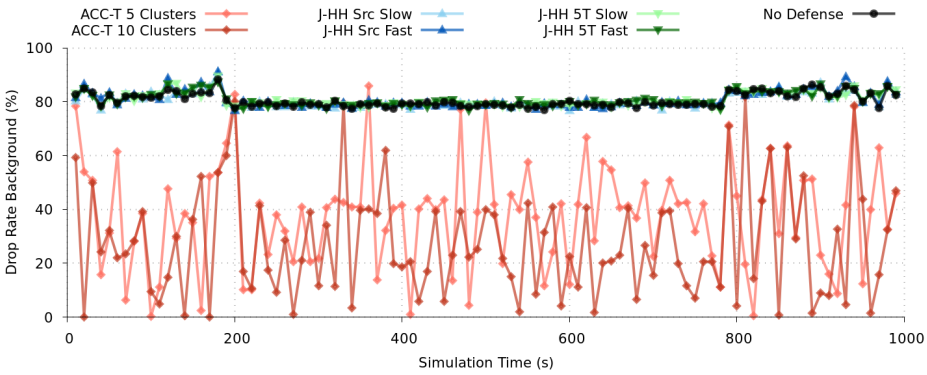


Figure 4.13: Time frame of feedback data considered for morphing.

It is clearly visible, that ACC-Turbo manages to let way more benign traffic through during this approach. This means that ACC-Turbo could be potentially improved by looking at the initialization of clusters and resetting them. This is a promising research direction. It would consist of two steps. First, ensuring that ACC-Turbo starts with a good initialization of clusters to begin with, and second, find methods to recover from suboptimal states. Such states could be detected through the variance in cluster size of the cluster sizes itself.

Chapter 5

Outlook

Further work in the area of adaptive morphing DDoS attacks is required in order to prove their practicality in practice. The provided framework, utilized and enhanced in this thesis, is suitable for testing more defenses and attacks for an even more comprehensive analysis. Different sources of background traffic could also easily be included to strengthen the findings of this thesis and stabilize the results even more. Although the adaptive attacks performed well in the simulated environments, the next step is to employ them on real hardware to see their effectiveness outside the virtually simulated lab.

Validating the two assumptions made for this work should be a critical element of future work. A side-channel attack or another exploit could extract real-time feedback on drop rates to satisfy the first assumption. Secondly, the presumption that traffic can be arbitrarily generated can only be partially accurate. While there are constraints on traffic generation, the ability of attackers to produce a significant variety of traffic should be sufficient to perform the attacks. Nonetheless, this aspect needs further investigation.

Lastly, this thesis discovered that the ACC-Turbo defense could be refined. While the authors already suggested the combination of signature-based defenses for known attacks with ACC-Turbo as a general defense against new attacks [2], there are two underlying open questions about using ACC-Turbo with fast clustering search as opposed to exhaustive search. They are essential, as fast search remains more feasible to implement on software-defined switches. First, special attention is needed on how clusters are initialized in ACC-Turbo. Currently, clusters are formed based on the initial packets observed, which may bias the system's behavior. Second, clusters could get stuck at a suboptimal distribution, decreasing ACC-Turbo's effectiveness because clusters only expand and never shrink.

A possible solution could be to detect this (utilizing the cluster size, variance or similar) and reset the clusters if needed. This approach, unlike exhaustive search, could be performed at line rate. However, further research and testing should be performed in this area.

Chapter 6

Summary

The aim of this thesis was to combine various DDoS attacks, defenses, and network configurations within a laboratory environment in order to facilitate the development of novel adaptive morphing DDoS attacks. To achieve this, the NetBench [5] framework utilized in the ACC-Turbo [2] study was extended and enhanced. Adjustments were made to the defense implementations of ACC-Turbo and Jaqen heavy hitters in order to enable them to process traffic data from PCAP files as well as an internal malicious traffic generator. This setup enabled the simulation and comprehensive analysis of standard, morphing, and newly developed adaptive morphing DDoS attacks, which rely on the assumption that real-time feedback about packet drop rates can be effectively utilized.

The experiments demonstrated that the newly developed adaptive morphing DDoS attacks were highly effective across all tested environments and against all implemented defense mechanisms, surpassing the performance of existing attacks.

As DDoS attacks become more sophisticated, it is crucial for defenses to evolve accordingly. Insights from this research were not only able to show that the new adaptive morphing DDoS attacks surpass the performance of existing attacks, but also to identify weaknesses in the ACC-Turbo defense and suggest possible improvement directions.

Bibliography

[1] Adedeji, K. B., Abu-Mahfouz, A. M., and Kurien, A. M. DDoS attack and detection methods in internet-enabled networks: Concept, research perspectives, and challenges. *Journal of Sensor and Actuator Networks 12*, 4 (2023), 51.

[2] Alcoz, A. G., Strohmeier, M., Lenders, V., and Vanbever, L. Aggregate-based congestion control for pulse-wave DDoS defense. In *Proceedings of the ACM SIGCOMM 2022 Conference* (2022), pp. 693–706.

[3] Budiu, M., and Dodd, C. The $p4_{16}$ programming language. *ACM SIGOPS Operating Systems Review 51*, 1 (2017), 5–14.

[4] Eliyan, L. F., and Di Pietro, R. DoS and DDoS attacks in Software Defined Networks: A survey of existing solutions and research challenges. *Future Generation Computer Systems 122* (2021), 149–171.

[5] Kassing, S. NetBench. GitHub repository, 2017. Available at `https://github.com/ndal-eth/netbench`. Last accessed on May 6, 2024.

[6] Kassing, S. Static yet flexible: Expander data center network fabrics. Master thesis, ETH Zurich, Zurich, Switzerland, 2017. Available at `https://doi.org/10.3929/ethz-a-010890129`. Last accessed on May 6, 2024.

[7] Liu, Z., Namkung, H., Nikolaidis, G., Lee, J., Kim, C., Jin, X., Braverman, V., Yu, M., and Sekar, V. Jaqen: A High-Performance Switch-Native approach for detecting and mitigating volumetric DDoS attacks with programmable switches. In *30th USENIX Security Symposium (USENIX Security 21)* (2021), pp. 3829–3846.

[8] Nguyen, K. (F5 Inc.) SODA Lab: Simulation of DoS attacks laboratory repository, 2019. Available at `https://github.com/KEN-Ver1/soda-lab` Last accessed on May 6, 2024. Main publication at [12].

[9] Santanna, J. J., van Rijswijk-Deij, R., Hofstede, R., Sperotto, A., Wierbosch, M., Granville, L. Z., and Pras, A. Booters—An analysis of DDoS-as-a-service attacks. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)* (2015), IEEE, pp. 243–251.

[10] Sharafaldin, I., Lashkari, A. H., Ghorbani, A. A., et al. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp 1* (2018), 108–116.

[11] Sharafaldin, I., Lashkari, A. H., Hakak, S., and Ghorbani, A. A. Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In *2019 international carnahan conference on security technology (ICCST)* (2019), IEEE, pp. 1–8.

[12] Tyagi, M., and Fedorov, M. Defense against rapidly morphing DDoS attacks. In *Proceedings of the Black Hat USA 2019* (2019). Available at `https://i.blackhat.com/USA-19/Wednesday/us-19-Tyagi-Defense-Against-Rapidly-Morphing-DDOS.pdf`. Last accessed on May 6, 2024.

[13] Yoachimik, O., and Pacheco, J. DDoS threat report for 2024 Q1. Available at `https://blog.cloudflare.com/ddos-threat-report-for-2024-q1`. Last accessed on May 6, 2024.

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every written paper or thesis authored during the course of studies. In consultation with the supervisor, one of the following three options must be selected:

○ I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used no generative artificial intelligence technologies[1].

○ I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used and cited generative artificial intelligence technologies[2].

◉ I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used generative artificial intelligence technologies[3]. In consultation with the supervisor, I did not cite them.

Title of paper or thesis:

Exploring Morphing DDoS Attacks

Authored by:
If the work was compiled in a group, the names of all authors are required.

Last name(s):	First name(s):
Kernbach	Nikodem

With my signature I confirm the following:
- I have adhered to the rules set out in the Citation Guide.
- I have documented all methods, data and processes truthfully and fully.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for originality.

Place, date	Signature(s)
Zürich, 06.05.2024	

If the work was compiled in a group, the names of all authors are required. Through their signatures they vouch jointly for the entire content of the written work.

[1] E.g. ChatGPT, DALL E 2, Google Bard
[2] E.g. ChatGPT, DALL E 2, Google Bard
[3] E.g. ChatGPT, DALL E 2, Google Bard