# Detecting Yo-Yo DoS attack in a container-based environment

Viktor Danielsen

Thesis submitted for the degree of
Master in Applied Computer and Information Technology - ACIT
(Cloud-based Services and Operations)
30 credits

Department of Computer Science
Faculty of Technology, Art and Design

Oslo Metropolitan University — OsloMet

Spring 2021

# Detecting Yo-Yo DoS attack in a container-based environment

Viktor Danielsen

# Abstract

Denial-of-Service (DoS) attacks are an ever persistent treath to IT environments and it occurs today at an ever-increasing rate from year to year. As the world develops and companies migrate their systems from private locations to public clouds, cyber criminals continue to use their classic DoS methods on cloud networks. Some clever cyber criminals also come up with new ways of rendering services unavailable for the intended users by exploring vulnerabilities in novel technologies such as clouds are. Still, typical attacks such as SYN floods and amplification attacks are the most popular attack vectors. There have been a big number of research on these classic DoS approaches, but security firms that operate DoS mitigation solutions continue to observe new trends in the DoS environment. More novel trends differs from the classic attacks by utilizing multiple attack vectors at the same time, hoarding big botnets and low-and-slow attacks among other. A novel DoS attack is the Yo-Yo attack which sends bursts of traffic in order to exploit cloud provider's auto-scaling functionalities. Auto-scaling is a very important line of defense against DoS attacks, but the Yo-Yo attack is specialized at exploiting a mechanism that we first thought only as an advantage that now could be a concern for cloud providers. The Yo-Yo attack is able to impact the victims web server with performance degradation as well as economic loss. This research will try to contribute with a detection mechanism against the Yo-Yo attack, as well as doing so in a container-based environment. This research has not been conducted earlier with a container-based environment as far as the author can tell, and it is highly relevant for the time being as more and more users are choosing containers for their web services, due to their lightweight being. The hypothesis is that (a) the Yo-Yo attack will not be able to determine the scaling phase or scaling policy due to rapid deployment of container instances or (b) that the Yo-Yo attack will induct rapid scaling of container instances and cause further economic loss than previously.

# Acknowledgments

I would like to thank my supervisors, Hårek Haugerud and Anis Yazidi, for guiding me through this master thesis and the last semester of my academic career. Thank you for providing me with your advice.

I would also like to thank Oslo Metropolitan University for giving me the opportunity to take this master's program.

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Introduction

Denial-of-Service, or DoS for short, is a phenomenon within IT environments that occurs when a system becomes partially or completely inaccessible due to overload caused by network traffic. DoS can occur as an eligible cyber attack or accidentally, e.g. due to the release of a new product or an event such as the release of concert tickets that are causing many users going to the same web service in a short time space. For most of the time we talk about Denial-of-Service as a cyber attack where the attacker's goal is to render a system (server/service) unavailable for legitimate users by sending network traffic in such an amount that overloads the target system. The target system can only handle a limited amount of load, defined by the resource capacities in the CPU, memory, disk and network bandwidth. There are many different types of DoS attacks. Foremost, DoS are divided into two main parts which are Denial-of-Service and Distributed-Denial-of-Service (DDoS). The difference between the two is that a DoS attack originates from one machine only, whereas a DDoS attack originates from multiple computers or devices, thus it being *distributed*. DDoS attacks are not surprisingly a bigger issue because it has a many-to-one dimension [20] and a DoS attack has a one-to-one dimension.

DDoS attacks is an ever-growing problem for IT infrastructures and if we look at the last couple of years the number of attacks increased with 50% in Q3 2020 in comparison with Q3 2019. Q2 2020 saw a huge spike with a doubling compared to Q2 2019, believed to be a result of the corona pandemic [22]. In the last three years, we've also seen the record for the biggest DDoS attacks measured in bandwidth broken twice with Github suffering an attack size of 1.3 TB/s in 2018 [42], and Amazon was hit by a 2.3 TB/s DDoS attack in early 2020 [8].

Following the emergence of cloud environments, both cybercriminals and sysadmins have found new techniques and methods to conduct and defend against Denial-of-Service attacks. As figure 1.1 illustrates, cloud computing emerged as a term in late 2006 after Google's CEO Eric Schmidt used the term in a conference. The term had it's peak for google searches in the summer of 2011, and has since then been a hot topic [41].

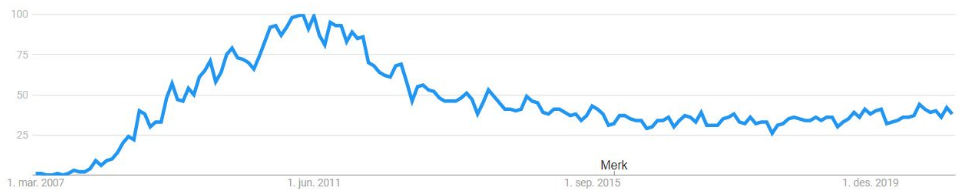


Figure 1.1: Cloud computing search trends in Google [1]

When we want to know who who wins a DoS attack, it all breaks down to the resources competition between the attacker and the victim. The ones that has the most resources will most likely overcome the other part. Especially, when we are talking about traditional flooding DDoS attacks where the attacker sends too much traffic for the target system to handle, resulting in an inaccessible system. There are also the attacks that turns focus to another goal which is Economic Denial of Sustainability, or EDoS for short, where the attacker wants to cause economic impact to the victim. This can happen when a victim is renting compute resources from a cloud provider which usually has some sort of scaling mechanism depending on the load of e.g. a website. The attacker can then exploit the scaling mechanism by launching an attack that will scale up the target system, which will result in a higher billing for the victim [9].

EDoS is typically a result of clever attackers that conduct low-rate Denial of Service attacks. This means that the rate of traffic received at the target victim is of a so low rate that it won't be detected by traditional mitigation or detection mechanisms. An example of a low-rate DoS attack is the Yo-Yo-attack where the attack is conducted in a on-attach/off-attack phase resulting in a scale-up/scale-down at the target's cloud environment as well. This particular attack will first send a burst of overload traffic and then try to detect when the target system is scaling up, and what kind of scaling policy it utilizes. When the attack senses that the victim has started its scale-up phase the overload traffic will stop. When this happens, the scaling mechanism at the victim will first have to wait for the scaling to finish, then it will scale down again to adjust to the drop in traffic. Then the attacker will send overload traffic again. This could go on for a while and the victim will be billed when the environment is scaling up because of additional VMs [9].

Container-based cloud environments gives us a new look on virtualization which is more lightweight than virtual machines. Containers can utilize resources more efficiently than VM-based environments and may pave the way for new defensive mechanisms and new attack methods.

## 1.2 Problem statement

*How to create a detection mechanism to prevent low-rate Yo-Yo attacks in a container-based environment*.

There aren't any other attacks than the Yo-Yo attack that explicitly targets auto-scaling functionalities in cloud environments. As more and more users are deploying services in cloud datacenters, an attack such as the Yo-Yo attack could potentially cost huge amounts of reputation and economy, and is putting many users at risk. Therefore, it is crucial that more research is put into creating detection and mitigation techniques, as well as policy evaluations of auto-scalers.

# Chapter 2

# Background and Related Work

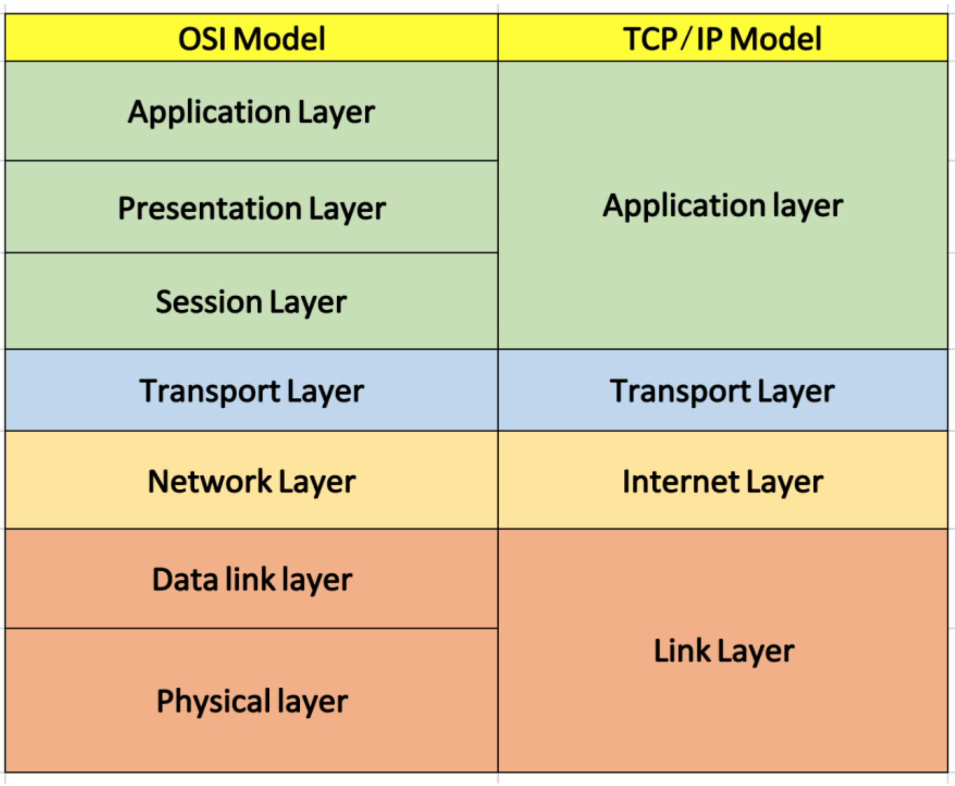
This chapter will provide the reader with all the information that is needed to get a lay of the Denial-of-Service land. It is important to know about different attack techniques and defense methods, as well as the history of Denial-of-Service. The chapter includes DDoS attack techniques, DDoS trends, Famous DDoS attacks, Relevant research and a Glossary to help the reader with understanding certain terms and phrases.

## 2.1 DDoS attack techniques

In order to understand how DDoS attacks occur, we have to have knowledge about the different techniques that are used. Traditionally, DDoS attacking techniques can be categorized based on the layers of the OSI-model as seen in figure 2.1. The layers that we typically observe DDoS attacks within are the network layer, transport layer and the application layer. We can also describe DDoS attacking techniques in matter of size and sophistication, e.g. there are the overwhelming TCP SYN floods that can send massive amounts of traffic, and there are the low-rate based attacks that aims at sending smaller amounts of traffic but more efficient requests that holds up more resources than more basic requests. As cloud became popular over the last decade, the attack landscape of DDoS attacks has changed, introducing novel techniques which we will come back to later in this chapter.

### 2.1.1 Network layer attacks

The network layer is the third layer of the OSI-model and the most important protocol here is the Internet Protocol(IP). Other protocols in Layer 3 are IPsec, ICMP, IGMP and ARP. DDoS attacks in this layer



| OSI Model | TCP/IP Model |
|---|---|
| Application Layer | Application layer |
| Presentation Layer | |
| Session Layer | |
| Transport Layer | Transport Layer |
| Network Layer | Internet Layer |
| Data link layer | Link Layer |
| Physical layer | |

Figure 2.1: The OSI-model and TCP/IP-model combined [2].

typically targets network equipment and infrastructure. Layer 3(L3) DDoS attacks differs from other layer attacks in the way it does not target layer 4 and layer 7 processes, and they doesn't have to use TCP connections with the target first. Layer 3 attacks does not target specific ports [14].

Known L3 attacks are:

- Ping flood: the attacker sends vast amounts of ICMP ping requests.
- Smurf attack: another ICMP based attack where the attacker sends spoofed ICMP packets to network devices, typically a router, which broadcasts the ICMP echo request to every device inside that network. The devices behind the router will respond with an ICMP reply packet, resulting in an efficient increase in number of packets that will be sent to the spoofed IP address. This attack use a technique called amplification, meaning that one request results in a much higher number of replies, thus the request is being amplified. This attack has been restricted in most network equipment today [17][24].
- "Ping of Death": the attacker sends a ping request to the target that is larger than what is allowed. The intermediary network equipment will fragmentize the packets in order to send it forward, but when the target system receives the packet for reassembling, it will crash. This flaw is not vulnerable in modern devices [14].

- DNS amplification attack: Amplification attacks in general are based on exploiting common stateless networking protocols like DNS (UDP based) and NTP by sending queries with spoofed IP addresses to the respective servers, which will then send responses that are many times bigger to the victim's clients or servers. In a DNS amplification attack, also called DNS reflection attack, the attacker exploits open DNS resolvers by querying them with small packets that becomes many times bigger. The source IP address is spoofed with the target's address. A simple demonstration shows that querying for a domain with the ANY record, a DNS resolver will respond with a size that can be 50 times bigger than the initiating request [40].

- NTP reflection attack: works in the same way as a DNS reflection attack where the attacker sends small queries that results in amplified responses, targeted at a victim. An attacker can query a NTP server that has the 'monlist' command enabled, with a spoofed address, and the NTP server will respond with a size that can be 206 times larger than the request [16].

### 2.1.2 Transport layer attacks

- TCP SYN flood: is by far the most popular DDoS technique observed on the internet today with 94.6% of all DDoS attack types as of Kaspersky's DDoS reports [28]. This technique exploits the TCP three-way handshake by leaving half-open TCP connections with the target system. The TCP three-way handshake is a mechanism for ensuring stateful communication between two devices. In a regular handshake, the initiator sends a SYN packet to start communicating with another client or server. The receiver will answer with a SYN+ACK packet and then wait for an ACK packet back to end the handshake and establish a proper connection between them. The initiator responds to the SYN+ACK packet with an ACK packet. To exploit the three-way handshake, the attacker can choose not to answer the SYN+ACK packet, thus leaving the connection half-open. This will hold up resources at the victim's system and eventually cause a buffer overflow if it can't process the traffic. There are different apporaches to this technique, but the main idea is for the attacker to not send the last ACK packet that ends the handshake [18].

- UDP flood: is about sending a huge amount of UDP packets preferably with spoofed source IP addresses. When the target system receives the UDP packets, it has to check if there is any service on the specific port that was requested and reply with ICMP "host unreachable" message if there isn't. As the target system has to check and respond to each UDP packet, this will quickly hold up resources and eventually a denial-of-service [19].

### 2.1.3 Application layer attacks

Application layer attacks, or layer 7 (L7) DDoS attacks, targets the top layer of the OSI model, popoular is the HTTP protocol. L7 attacks tend to consume more resources than its relatives in L3 DDoS attacks, as they consume both network resources and server resources. Thus, it does not require as much bandwidth to conduct a L7 DDoS attack than in lower layers. Most websites are running on webservers which do a lot of processing even for a regular benign request, this being API calls, authentication or database calls [12].

- HTTP flood: HTTP is the basis of requests made towards browser-based web requests, mostly HTTP GET and HTTP POST requests. Thus, there are two types of HTTP DDoS attacks. HTTP GET floods consists of requests that request web content such as images and files that the targeted server will have to allocate resources in order to reply. HTTP POST is often used with forms that users have to fill out. Attackers can exploit this by sending HTTP POST that the server have to push to a database of some sort. To defend against HTTP POST requests, many webpages can be seen with a CAPTCHA challenge that the user have to respond to. Eventually, if the amount of requests is high enough, this will lead to a denial-of-service [15].

### 2.1.4 Fraudulent resource consumption attacks

Attacks where the goal is to consume resources of a cloud environment, causing financial losses to the victim [6]. A fraudulent resource consumption (FRC) attack differs from all of the traditional denial-of-service attacks described above, in the way that the goal has shifted from causing denial-of-service by overwhelming the target system with network traffic, to instead targeting the finances of the victim. FRC attacks are specialized to target customers of cloud service providers (CSP) and exploit the pricing models that the CSPs utilize. By sending low amounts of network traffic over a longer period of time, the attacker can go undetected by traditional DDoS detection mechanisms because the number of requests isn't high enough. If the attack can go on for a while, the victim will receive a significant bill from its CSP [6].

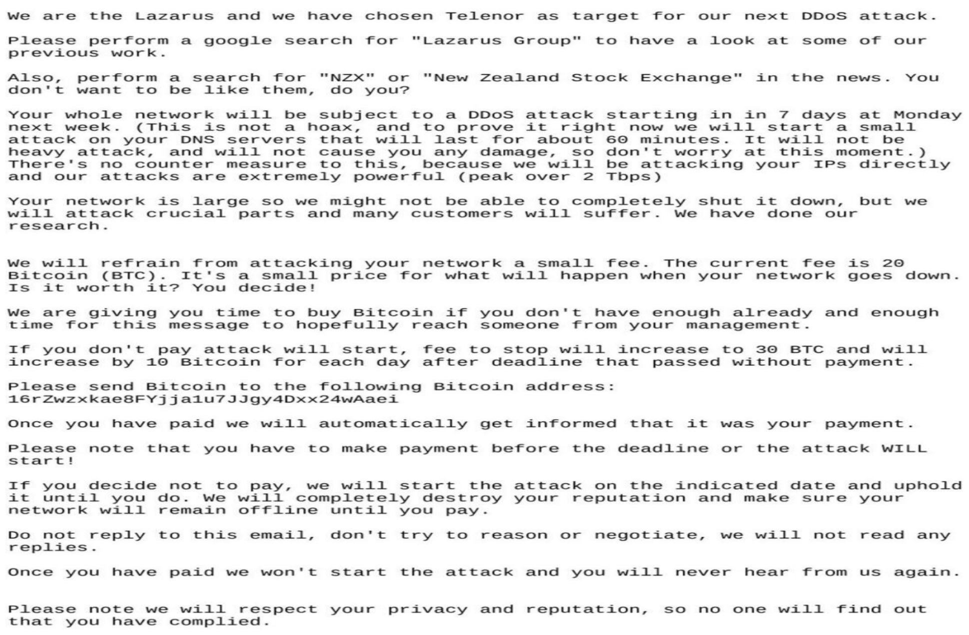
### 2.1.5 Low and slow attacks

Low and slow DDoS attacks are a type of attack that requires little bandwidth and attacks at a low rate to avoid detection by traditional DDoS mitigation mechanisms. Instead of relying on a quick takedown, the attacker unleash a slow trafficstream that is high enough to cause enough interference on a server to deny or slow down service, or inflict economical losses to a company which gets DDoS involved with EDoS. Low and slow attacks are hard to detect because they are very similar to normal traffic. Slowloris and R.U.D.Y are tools that can be used to conduct low and slow attacks. Slowroris for instance, is a tool that doesn't require a lot of bandwidth. It's HTTP based and aims at reserving all available HTTP connections/threads at the targeted server until legitimate users aren't able to. R.U.D.Y (R U Dead Yet?) exploits web forms by sending HTTP POST packets at a very slow pace, holding up resources [21].

## 2.2 DDoS trends

### 2.2.1 Ransom DDoS-attacks (rDDoS)

An increasing number of DDoS attacks have been known to be related to ransom DDoS-attacks, which will be talked more about in the Kaspersky reports. Figure 2.2 is an excerpt from a ransom letter that Telenor in Norway received as a DDoS treath, allegedly from cyber criminal imposters hiding behind the APT group name «Lazarus Bear Armada»."[38].



```
We are the Lazarus and we have chosen Telenor as target for our next DDoS attack.

Please perform a google search for "Lazarus Group" to have a look at some of our
previous work.

Also, perform a search for "NZX" or "New Zealand Stock Exchange" in the news. You
don't want to be like them, do you?

Your whole network will be subject to a DDoS attack starting in in 7 days at Monday
next week. (This is not a hoax, and to prove it right now we will start a small
attack on your DNS servers that will last for about 60 minutes. It will not be
heavy attack, and will not cause you any damage, so don't worry at this moment.)
There's no counter measure to this, because we will be attacking your IPs directly
and our attacks are extremely powerful (peak over 2 Tbps)

Your network is large so we might not be able to completely shut it down, but we
will attack crucial parts and many customers will suffer. We have done our
research.

We will refrain from attacking your network a small fee. The current fee is 20
Bitcoin (BTC). It's a small price for what will happen when your network goes down.
Is it worth it? You decide!

We are giving you time to buy Bitcoin if you don't have enough already and enough
time for this message to hopefully reach someone from your management.

If you don't pay attack will start, fee to stop will increase to 30 BTC and will
increase by 10 Bitcoin for each day after deadline that passed without payment.

Please send Bitcoin to the following Bitcoin address:
16rZwzxkae8FYjja1u7JJgy4Dxx24wAaei

Once you have paid we will automatically get informed that it was your payment.

Please note that you have to make payment before the deadline or the attack WILL
start!

If you decide not to pay, we will start the attack on the indicated date and uphold
it until you do. We will completely destroy your reputation and make sure your
network will remain offline until you pay.

Do not reply to this email, don't try to reason or negotiate, we will not read any
replies.

Once you have paid we won't start the attack and you will never hear from us again.

Please note we will respect your privacy and reputation, so no one will find out
that you have complied.
```

Figure 2.2: Ransom letter related to a DDoS attack against Telenor in October 2020 [3].

### 2.2.2 Kaspersky reports

These reports are produced by Kaspersky Lab, a security company that delivers security solutions within many fields, including DDoS protection. More than 400 million users are protected by some sort of Kaspersky solution [25]. They collect statistics from the DDoS Protection system that monitors botnets, meaning that the reports are limited to numbers from these botnets. DDoS attacks gets measured by number of attacks, duration, attack vector, geographical origin and target, target sector amongst other.

**TL;DR**

This is a summary of all reports that are provided after the summary.

By looking at Kaspersky Lab's reports over the last 5 years we can safely conclude that the DDoS market is highly dynamic in the number of attacks and sort of stable in other indicators like distribution of countries. The overall number of DDoS attacks are increasing from year to year except from some abnormalities (e.g. 2018), and the most common attack type is by far SYN flooding with a share of over 50% each year.

The use of IoT botnets boomed in 2015/2016 with the emergence of the Mirai botnet that was observed in Q3 2016 which eventually led to huge attacks on DYN's DNS servers in the U.S.

The last couple of years have seen Docker services becoming more popular and DDoS criminals have found their ways to utilize them as well. Kaspersky has observed malware that infects Docker containers with botnet malware and cryptominers.

What motivates DDoS activity is a handful and the target sectors are many. Politics, money, "hacktivism", protests, gaming, educational institutions among others. Ransom DDoS was a popular threat in 2020 where cybercriminals sent out ransom letters by email, demanding bitcoins in exchange of them not attacking the victim. These mails were followed by a demonstration attack to prove the threats were real. The attackers allegedly faked their identity under famous APT groups like Lazarus and Fancy Bear.

China and the United States are topping the charts every year on origin of attack and attack target. The lower places on the list are changing from quarter to quarter but common countries are Great Britain, Hong Kong, South Africa, South Korea and India.

**DDoS attacks in Q4, 2020 [33]**

**News overview** Cybercriminals are always on the lookout for new methods to conduct attacks, and in Q4 of 2020 attackers found an exploit in Citrix products that utilized the DTSL, a protocol for secure connections over UDP, which could be used for amplification attacks, amplifying up to 36 times the request size.

Bitcoin.org was hit by a DDoS attack which is normal when the bitcoin value is increasing.

Q4 retained within the trends of 2020 where ransom DDoS hit Telenor, a telecommunications company in Norway. The attackers pretended to be a famous APT (advanced persistent threat) group, demanding ransom to prevent an upcoming attack.

Further, schools in the US and gaming platforms where popular targets in Q4.

The Internet Engineering Task Force (IETF) published a proposal for NTS in RFC8915, a secure protocol for NTP, just like HTTPS for HTTP. NTP is a popular attack vector for amplification attacks.

**Bitcoin and DDoS symbiosis** Bitcoin and DDoS seems to be living in relation to each other as when one is low the other one is high. The bitcoin value rose in the last quarter of 2020 which resulted in a decrease in DDoS attacks. The reason for this is that cybercriminals that usually utilize their computing power to conduct DDoS attacks, switch over to mining bitcoins instead.

**No decrease in smart attacks** However, smart attacks did not decrease. Smart attacks are mostly conducted by sophisticated criminals that are more focused on other results that financial winnings, so these attacks won't be infcluenced by the bitcoin value as much as other attacks conducted by individual hacksters.

**The number of DDoS attacks slightly less than doubled in 2020 from 2019** while the number of smart attacks was more or less unchanged.

**Average attack duration** declined by a third while the maximum durations increased. Short attacks are getting shorter and long attacks longer.

**The top attack vectors** for Q4 2020 had UDP flooding (15%) at second place and GRE flooding, not previously mentioned in Kaspersky's reports, on fourth.

**The number of DDoS attacks per day** increased throughout Q4 with December 31. being the most hectic day with 1349 attacks. The increase in

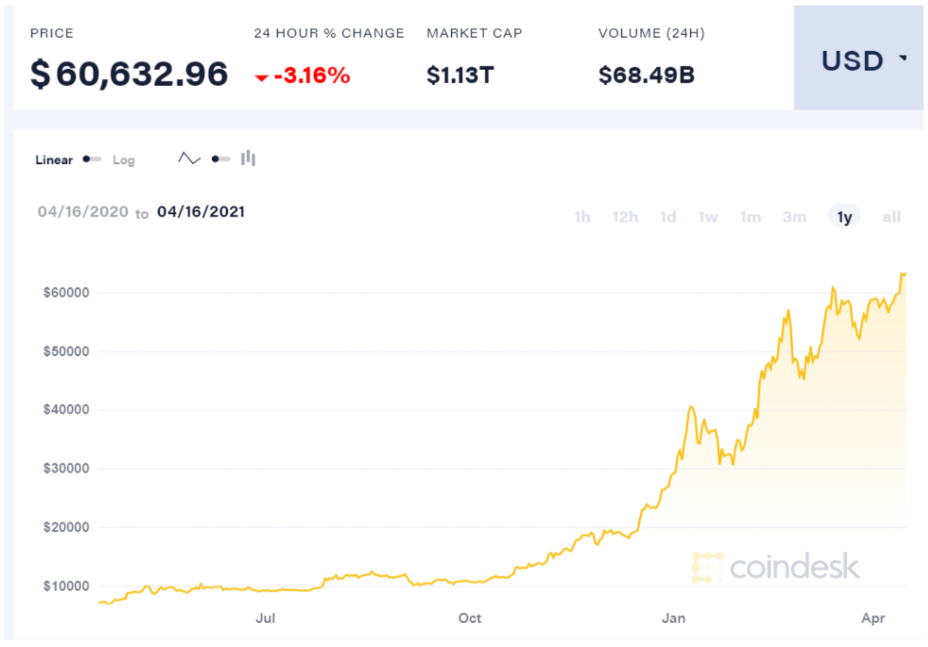


Figure 2.3: Bitcoin value as of 16.04.2021 [4]

December is usually due to the holiday shopping bonanza.

**Distribution of DDoS attacks types:** SYN (78%), UDP (15%), TCP attacks (5%), GRE (0,69%) and HTTP (0,39%).

**Almost all botnets were comprised of Linux machines (99,8%).**

**DDoS attacks in Q3, 2020 [28]**

**News overview** More malware targeting Docker containers, infecting them with cryptominers and botnet malware.

Extortion attacks: attackers that seems to be hiding behind infamous APT groups sends out ransom emails, demanding bitcoin ransom and threatening to launch a DDoS attack.

New Zealand Stock Exchange hit by DDoS and taken offline for days. Also attacks against famous financial companies, media firms and schools.

**Decline in DDoS attacks**, may be due to companies having adapted to the coronavirus pandemic. The cryptocurrency market grew in Q3 and may also be a cause for the decline.

**Distribution of attack vectors:** SYN attacks is by far the most popular attack method with 94,6% usage. Following are ICMP (3,4%), HTTP flooding (0,1%).

**Linux machines** comprise 94,6% of all botnets.

**Number of attacks per day** peaked at 323 attacks on July 2.

**An increase since last year**, but a decrease since last quarter.

#### DDoS attacks in Q2, 2020 [27]

**News overview** Two new amplification methods discovered by Israel and China, utilizing DNS and HTTP.

Docker containers vulnerable to botnet malware and cryptominers: (https://unit42.paloaltonetworks.com/lucifer-new-cryptojacking-and-ddos-hybrid-malware/).

DDoS attacks finding its way to the news were targeting human rights organizations in the U.S, Russian Central Election Commission and Media firms.

**Quarter statistics** Top three most attacked countries: China 65,12%, U.S 20,28% and Hong Kong 6,08%.

The number of attacks tripled from last year's Q2. This is abnormal as Q2 usually sees a drop in numbers from Q1.

Linux botnets holds the majority at 94,78% to Windows at 5,22%.

**Distribution of attack vectors:** SYN flooding 94,7%, ICMP attacks 4,9% and other methods were below 1%.

#### DDoS attacks in Q1, 2020 [26]

**News overview** COVID-19 struck the entire world in march and has affected the internet since. People work, shop and entertain themselves online like never before which is reflected in the DDoS environment. Most DDoS targets in Q1 were websites of medical organizations, delivery services, and gaming and educational platforms.

Attacks that made the headlines in Q1 were attacks against the US Department of Health and Human Services, trying to disrupt the information flow about e.g. quarantine. Other victims were a hospital-group in Paris, two food delivery services in Germany and Netherlands, an educational web platform in Germany on the first day of remote school, gaming servers owned by Battle.net and Eve Online. There were also politically motivated attacks the most noteworthy against Greece.

Ransom DDoS attacks.

**The total number of DDoS attacks nearly doubled** from Q1 2019 (80% up).

**Distribution of attack vector:** SYN flooding 92,57%, ICMP attacks 3,62%, UDP 1,84%, TCP 1,68% and HTTP 0,29%.

### DDoS attacks in Q4, 2019 [32]

**News overview** Ransom DDoS attacks.

**Distribution of attack vectors:** SYN flooding 84,60%, TCP 5,90%, UDP 5,80%, HTTP 2,20% and ICMP 1,60%.

**Quarter and year statistics** Total amount of attacks doubled over the year from Q4 2018.

Compared to 2018, the DDoS activity increased in all indicators in 2019. Total number of attacks increased by 33%, as did smart attacks by 43%. Average duration increased with 35% and average duration with smart attacks increased with 44%.

**Predictions for 2020** Kaspersky's experts predicts that the DDoS market stabilizes in 2020 as there haven't been any exposure of serious vulnerabilities or growth in the cryptocurrency market.

### DDoS attacks in Q4, 2018 [31]

**News overview**

**Distribution of attack vectors:** SYN flooding 58,2%, UDP 31,1%, TCP flooding 8,40%, HTTP 2,20% and ICMP 0,10%.

**Quarter and year statistics** There was a decrease in overall activity in 2018 compared to 2017, by 13%.

### DDoS attacks in Q4, 2017 [30]

**Distribution of attack vectors:** SYN flooding 55,63%, UDP 15,24%, TCP 13,06%, HTTP 12,70% and ICMP 3,37%.

#### DDoS attacks in Q4, 2016 [29]

**News overview** "The year of DDoS" as more cybercriminals are utilizing botnets with the Mirai botnet first observed in Q3 2016 causing a lot of disruptions, ending Q4 with a bang with a series of attacks against DYN's DNS servers in the U.S., cauing 85 web sites to go down, among them Netflix, Twitter and PayPal.

**Amplification attacks** are decreasing and seems to be going out of date, as there have been a decline over the last half of 2016.

**IoT botnets on the rise.** Kaspersky Lab has observed a rise in IoT botnets over 2016 after the Mirai botnet demonstrated how powerful it can be.

**Distribution of attack vectors:** SYN flooding 75,3%, TCP 10,7%, HTTP 10,3%, ICMP 2,2% and UDP 1,4%.

## 2.3 Famous DDoS attacks

This is a list of the most famous DDoS attacks that have occurred. The attacks are famous because of their size, motivation behind the attack, and/or the impact they had on the society.

#### Panix, 1996

The DDoS attack against Panix, an ISP in New York, was the first known DDoS attack. The ISP was offline for days, and the attack utilized a SYN flood.

#### Mafiaboy, 2000

A 15-year high school hacker under the nickname "Mafiaboy" amassed the computers of several universities and schools into a DDoS attack that took down big sites CNN, Yahoo!, Ebay and Dell among others. The attack also created chaos on the stock market. Later, several cybercrime laws were created as a direct consequence to this attack [13].

#### Estonia attack, 2007

The Estonian government's web services was hit in 2007 in relevance with a dispute with Russia about a statue called "The Bronze Soldier of Tallin" that commemorates a Soviet World War 2 soldier. Estonia was early on with online governmental information and elections making them susceptible to DDoS attacks. After the attack, several international laws on cyber warfare was created [13].

**The six bank attack, 2012**

An attack that utilized the Brobot botnet targeting U.S banks Bank of America, JPMorgan Chase, U.S. Bank, Citigroup, Wells Fargo, and PNC Bank. The attack was a bit unique at the time because it used several attack methods rather than backing down after one failed method. This costed the banks money and revenue. The attack was allegedly conducted by a group within Palestinian Hamas [37].

**Spamhaus, 2013**

An attack against Spamhaus that peaked at a size of 300 Gbps, conducted by a teenage hacker-for-hire. The web site responded to the attack by signing up to Cloudflare DDoS protection which mitigated the attack. The attacker had to re-target the attack which in turn disrupted internet exchange of London. Spamhaus is an organization that combats spam activity [13].

**Occupy Central, Hong Kong DDoS Attack in 2014**

Largest attack at the time at a peak of 500 Gbps hitting Occupy Central's web sites, a pro-democracy ogranization in Hong Kong. Two other sites, Popvote and Apply Daily, that were supporting Occupy Central's cause were also hit by the attacks. Presumably conducted by Chinese government or someone else that doesn't like the democracy concept in Hong Kong [37][39].

**The CloudFlare DDoS Attack in 2014**

A single customer of CloudFlare was hit by a DDoS attack in 2014 that created implications in CloudFlare's infrastructure as well. The attack reached a size of 400 Gbps, a record at the time. The attack was a NTP reflection attack[37].

**Github, 2015**

Another attack against Github, and the largest recorded at the time. It's said that the attack originated in China as the attack targeted two specific URLs that led to Github projects against Chinese state censorship. The attack was a Javascript injection attack where the infected browsers sent HTTP requests [13].

**Dyn DNS, Mirai, 2016**

In 2016, an attack against Dyn's DNS servers occurred, disrupting the services of about 80 major web sites in the U.S such as PayPal, Netflix, AirBnB. The attacker utilized the Mirai botnet which consists of Internet-of-Things devices, and demonstrated the powers of such a botnet, reaching a size of up to 1,5 Tbps. The Mirai botnet had unleashed attacks earlier the same year, and the source code for the Mirai malware had been released to the public. The motivation behind the attack is unknown [13][37].

**Google, 2017**

The biggest attack known to us today is a DDoS attack against Google services in 2017. The attack lasted for about six months, culminating at a size of 2,54 Tbps. The attack was allegedly conducted by Chinese actors, according to Google. The attack was an UDP amplification attack where the attackers used spoofed packets targeted at exposed CLDAP, DNS and SMTP servers [11][13][23].

**Github, Memcached, 2018**

Github contracted a big DDoS attack in 2018 where the attackers exploited the memcached service. The attackers did not use a botnet, instead the memcached service made it possible to amplify the requests sent to the servers up to 50000 times. The attack reached 1,3 Tbps, but was mitigated quickly, after 20 minutes. Memcached is a database service used to speed up web sites and networks [13][37].

**AWS, 2020**

Amazon Web Services mitigated an attack peaking at a size of 2,3 Tbps which utilized the CLDAP protocol [13][37].

## 2.4 Timeline

- 1989
  - Tim Berners Lee at CERN, invented the World Wide Web (W3C, n.d.).
- 1993
  - Mosaic, the first web browser to display text and images.
- 1995
  - First phishing attack, targeting America Online (Nollinger, 1995).
  - CERT/CC released an advisory about the growing phenomenon TCP SYN flooding and IP spoofing [10].
- 1996
  - DDoS attack against Panix.
  - Internet Explorer 3 was launched, becoming the most popular web browser.
  - Microsoft released the email service Hotmail.
- 1997
  - The first blacklist, created by Paul Vixie as a response to spam (Caldwell, 2017).
- 1998
  - Google was founded.
- 2000
  - Mafiaboy DDoS attack.
  - Nokia 3310.
  - mnemonic was founded.
- 2001
  - The dotcom bubble bursts (Wollscheid, 2012).
  - 3G mobile broadband.
- 2003
  - APWG, The Anti-Phishing Working Group was founded.
  - Number of spam emails exceed the number of legitimate emails for the first time (Wallace B. , 2019).
- 2004

  - Mozilla Firefox.
  - Facebook was founded.
  - FireEye was founded.

- 2005
  - Palo Alto was founded.

- 2006
  - Rustock Botnet, infected systems acted as proxy servers to send further spam. Was responsible for almost all spam emails worldwide. Taken down in 2011 (Leyden, 2011).
  - Gmail and Google Apps.
  - Twitter was founded.

- 2007
  - Estonia DDoS attack.
  - First email phishing filter (Fette, Sadeh, & Tomasic, 2006).
  - Safari.

- 2008
  - Google Chrome

- 2009
  - Stuxnet attack on Iranian nuclear factories (Zetter, 2014).

- 2010
  - Operation Aurora, a series of cyberattacks originating in China, targeting firms Adobe and Akamai among others.
  - 4G mobile broadband.

- 2011
  - RSA breached by phishing attacks, exposing master keys for all RSA SecureID security tokens (Markoff, 2011).
  - Office 365 cloud services.

- 2012
  - The six banks DDoS attacks.
  - Launch year of IP version 6.

- 2013
  - DDoS attack against Spamhaus, 300 Gbps.
  - Target's data breach. 110 million customers affected (Bing, 2013).

  - CryptoLocker ransomware attacks.
- 2014
  - Cloudflare DDoS attack, 400 Gbps.
  - Occupy Central, Hong Kong DDoS Attack, 500 Gbps.
  - Emotet, first discovered. A trojan that primarily spreads through email and Is still active today (Malwarebytes, n.d.).
- 2015
  - Github DDoS attack,
  - Ukrainian power grid attack conducted by Russian cyberintelligence. The initial attack vector was email phishing (BBC, 2017).
- 2016
  - Dyn DNS, Mirai DDoS attack, 1,5 Tbps.
- 2017
  - Google DDoS attack, 2,54 Tbps.
  - WannaCry ransomware attacks.
- 2018
  - Github Memcached DDoS attack, 1,3 Tbps.
- 2019
  - Close to 4.7 billion phishing emails are sent every day (Wallace, 2019).
  - Threats to cloud-security in Office 365 have increased by 63% since 2017 (Wallace, 2019).
  - 90% of businesses are using cloud hosted email or are planning to (Wallace, 2019).
  - 5G mobile broadband.
- 2020
  - Amazon Web Services DDoS attack, 2,3 Tbps.

## 2.5 DDoS attack on cloud auto-scaling mechanisms

Bremler-Barr et al.[9] demonstrated how clever attackers could bypass detection mechanisms with low-rate DDoS attacks and exploit auto-scaling mechanisms in cloud environments. The nature of a low-rate DDoS attack is that it sends out a smaller amount of network traffic than the more classical flooding DDoS attacks where the victim is overwhelmed with traffic (add reference).

Auto-scaling mechanisms is an important weapon to counter DDoS attacks in cloud environments. With an unlimited budget, a user of a cloud provider basically has unlimited resources. Because of this, many believe that DDoS rather turns over to Economic Denial of Sustainability (EDoS) attacks since it is the cost of scaling up more machines that the victim suffers from.

The authors present and analyze a new kind of DDoS attack, the YoYo-attack, which penalises both performance and economy. This attack exploits the use of auto-scaling in the cloud and is doing so by sending bursts of overload traffic to trigger the scale-up and scale-down phases.

Cloud auto-scaling:

Auto-scaling is a cloud computing feature that automatically adds or removes computing resources depending on the actual usage, usually measured in CPU utilization and other criteria (reference). Every cloud solution comes with its own auto-scaling service which the user can customize to adapt to their own needs by tweaking thresholds.

Users need to define when the auto-scaling should happen by defining rules for the scale-up and scale-down phases, as well as the maximum and minimum number of machines allowed. The user also need to choose auto-scaling policy, discrete or adaptive. The discrete policy increases or decreases the number of machines iteratively according to a predefined value, e.g. 10. If the load is too high, the auto-scaling service will scale-up and add 10 machines to the environment. Then the service have to determine if the issue have been resolved or not. The adaptive policy increases or decreases differentially, and tries to adapt to the load. This is achieved by defining different increase and decrease in machines according to different thresholds, e.g. increase with 10 machines at a total load of 50%, and increase with 30 machines at a total load of 90%.
After a scale-up decision has been made it takes time before the machines are ready to be utilized. This is called the Warming time and differs depending on the infrastructure provider, operating system, service initialization time and other factors [36]. Scaling down takes time as well.

The YoYo-attack:

The YoYo-attack switches between an on-attack phase and an off-attack phase In the on-attack phase the attacker is sending bursts of overload traffic, causing the victim's auto-scaling feature to scale up its environment. The attacker is able to determine when the victim is scaling up and down, so when a scale-up has been triggered in victim's cloud, the attack will switch to the off-attack phase, waiting for the victim to scale down again, and so it continues. The attacker is in this case able to exploit the benefits from auto-scaling.

There have been several attempts at combating EDoS attacks but most, or all of them, are ignoring the auto-scaling effects, and the impact of an attack such as the YoYo-attack.

The YoYo-attack can also be described as a kind of Reduction of Quality (RoQ) attack, and other mechanisms, such as load balancing are vulnerable to such attacks. Several papers discuss how to have an efficient auto-scaling policy but they do not keep an eye on the security vulnerabilities.

Analysis of the YoYo-attack

Econmic and performance damages are larges when the victim is using an adaptive auto-scaling policy. The YoYo-attack potentially does more damage per unit cost than the DDoS attack. The DDoS attack causes the auto-scaling to deploy more machines, but the load on each machine does not increase noteworthy, while the YoYo-attack increases the load by 100%.

Detecting scale policy

The attacker can send probe packets and check their response time in order to detect when the scale-up process has ended. Higher response time on each request tells the attacker that there aren't enough machines in the victim's environment yet, thus the scale-up process hasn't ended. When the response time dropped beneath 1000ms the attacker in this case would stop the attack in order for the victim's auto-scaler to scale down. A simmilar method is used to detect when scale-down has ended.

Defense strategies against the YoYo-attack

- Scale up early - Scale down slowly
- Restrictions - Limiting the resources This will be to avoid sudden expenses, butt may cause Denial of Service

They conclude that system administrators have to compromise between DDoS and EDoS, high service cost or low performance. [9]

## 2.6 Exploring New Opportunities to Defeat Low-Rate DDoS Attack in Container-Based Cloud Environment

In this reasearch paper, Li et al.[35] investigates mitigation methods in container-based cloud enviornments, specifically methods against low-rate DDoS attacks. As container-based clouds are growing in numbers, it is natural to find out if such environments are better at resource usage and if there are new methods to mitigate attacks.

Containers are more lightweight than virtual machines, therefore it will be faster to scale up and down a cloud service. Together witt a microservice architecture, there are other scaling possibilities than with a monolithic VM architecture.

Resources competition. Containers can scale quicker and more efficiently than VMs. Intrusion Detection Systems can be used to detect flood based DDoS, but low-rate DDoS attacks can evade such systems. Each request in a low-rate attack can consume more resources than a traditional attack.

Many mitigation techniques have been developed due to the shortcomings of detection systems. Resource-scaling, to automatically extend additional resources to the victim instance. The isolation mechanism, isolates the victim service. The limitation mechanism, limits the resources of the victim service to ensure usability of other services. With these techniques, one can successfully mitigate a low-rate DDoS attack, but the research these techniques have come from are all based on virtual machine based cloud environments, not container-based environments, and that's why Li et al. wants to do this research.

Contributions:
"We explore the possibility that utilizing the new features in container-based cloud environment defeats the low-rate DDoS attack. And we point out the strengths and weaknesses to mitigate low-rate DDoS attack in the container-based cloud environment.

We establish a mathematical model based on queueing theory to formalize the low-rate DDoS attack scenario in container-based cloud environment and analyze the capacity of container-based cloud environment in defeating against low-rate DDoS attack.

Guided by this model, we propose a dynamic mitigation mechanism to optimize and coordinate the resource allocation and the number of containers for mitigating the low-rate DDoS attack."

Related work:
DDoS Attacks Mitigation in Cloud Environment
The main goal of a DDoS attack is to exhaust the victim's resources to ensure Denial of Service. Several techniques have been developed to fight attacks in cloud environments. Using an abundance of resources is one technique. Using idle resources and a queueing theory based model is another. Since resources aren't free in cloud environments, DDoS attacks usually turn over to EDoS attacks. Sqalli et al. proposed a filtering approach with a Turing test to counter EDoS, and amazon has its own service, "Cloudwatch" to reduce the impact of EDoS. Victim migration and resource management are other techniques. Few studies have been conducted that focus on DDoS mitigation in container-based cloud environments, they have only found one related in Ye et al.

low-Rate DDoS Mitigation in Cloud Environment:
low-rate DDoS attacks differ from traditional flood-based attacks and the detection mechanisms fail so reasearchers have developed other detection mechanisms. These are divided into two categories, which are based on network traffic or application vulnerabilities. Techniques in network based: generalized entropy and information distance, mathematical model based on behaviors of victim TCPs congestion, and mathematical model combined MF-DFA algorithm with Holder exponent. In application vulnerability based, the vulnerabilities or logical errors must be detected and fixed before an attack occur. (I think this is more about testing). Low-rate DDoS mitigation techniques are still necessary.

Security of Container-Based Cloud Environment:
Today(2019), security issues in containers are focused on the security drift problem and the isolation problem. Done on security drift: "analyses on the security issues brought by the high degree of agility, reusability, and portability with container", and about isolation: "systematically identified the information leakage problem and investigated potential container-based power attack threats built upon these leakage channels" and another one on isolation: "used the SGX to enhance the isolation between containers and protect containers from outside attacks".

DDoS mitigation mechanism:
A dynamic DDoS mitigation system that maintains microservice availability during a low-rate attack and maximizes Quality of Service (QoS) with limitied resoures. They experiment with an e-commerce website that comprise of microservices with containers. Users are able to browse the website's products without authenticating, and users can also log in. During a DDoS attack, the mitigation mechanism will use a whitelisting of users, where the authenticated users' requests are prioritized over the unknown request, meaning they will be assured enough resources. The unknown

requests will consist of both malicious and benign requests, so there will be some legitimate users that may not be able to access the website's content. The amount of resources/containers needed is calculated in accordance to the number of whitelisted users. By doing this whitelisting, the microservice is logically divided/isolated. The mitigation mechanism rely on the same principles of an auto-scaling system, that scales up or down due to a threshold of resource consumption; in this case at 80

Poission distribution and M/M/c queuing model to calculate service capacity.

## 2.7 Towards Yo-Yo attack mitigation in cloud auto-scaling mechanism

[43] In this article, Xu et al.[43], builds on the research conducted by Bremler-Barr et al.[9] which is also described in this chapter. This article aims at creating a detection and mitigation mechanism agains the clever Yo-Yo attack that Bremler-Barr et al. demonstrated. The attack exploits auto-scaling mechanisms in cloud environments, which is thought of as a functionality that is inevitable to counter a DDoS attack in the first place. The authors create a trust based system to identify benign and malicious users, as well as manipulating request responses to deceive the attacker.

**Intro** Cloud has become highly attractive after its arrival and most cloud providers offers auto-scaling as a service in order to dynamically scale a customer's services relative to the load. This is huge benefit to customers as they wont run out of resources, which might have been an issue before they converted to the cloud. As bought Bremler-Barr er al. and Xu et al. concludes is that there haven't been done much research on the downsides of such an auto-scaling mechanism.

The Yo-Yo attack is unique in the way that it punish both performance and the economics of the victim. The reason for this is that the auto-scaling mechanism, which is there to ensure Quality of Service for the users, is exploited by sending overload traffic in sequences that triggers both the scale-up process and scale-down process.

Their approach is TASD (Trust-based Adversarial Scanner Delaying). This system identifies benign users by adding a trust value to the user. They also manipulate the response time of requests in order to deceive the attacker, as the Yo-Yo attack relies on sending probe packets to find out which scaling phase the victim system is at.

**Approach/method** The TASD consist of a Detection module and a Defence module. The detection module keeps a list of users and a score/trust value related to the user. The trust value is measured by looking at when the user is requesting data, i.e. is the user requesting data in the scale-up or scale-down phase. The load each user has according to scaling phase is also evaluated.

**Conclusions** The TASD system is able to detect many malicious users by 80%. It also decrease the malicious scale up and down by 41%.

**Cons** Their detection module wont be able to detect and mitigate a Yo-Yo attack if the attack is distributed as there can be many unique users/IP-addresses.

Their research is based on a VM environment which has a higher warm-up time than containers.

They add a delay to suspicious requests, but if every malicious user is new to the module, will it detect that?

## 2.8 Glossary

- Amplification attacks: attacks where a method of amplifying a packet is used by exploiting vulnerabilities. For example, there are certain vulnerabilities in the DNS protocol that allows attackers to get a much bigger response in size (50 times) than the original request was.
- Botnet: a group of computer devices that have been infected with malware and is under control by a malicious actor. The term "botnet" is a combination of the words "robot" and "network". The botnet is intended to be used as a tool in cyberattacks, e.g. to produce massive amounts of network traffic or distribute malware. All devices that are connected to the internet are possible botnet candidates [34].
- Container technology – An alternative virtualization technology to Virtual Machines. Opposed to VMs, which are running a complete OS, containers share kernel with the host system and support minimum runtime requirements of the application. Containers don't use hypervisors. Containers rely on namespaces and CGROUPS to achieve isolation and resource control. Containers have better resource control than with VMs. Containers are much more lightweight which gives faster startup time, superior I/O performance and lower latency than VMs. Containers needs fewer system resources to run which can make single server host more containers than VMs obviously. Containers also provide a portable environment that don't rely on the different cloud environments. Hypervisors on the other hand are more heavyweight, slow boot-time and has a higher runtime overhead [44].

# Chapter 3

# Approach

For this project I will be using VirtualBox to set up the experimental environment. This way, there wont be any charges for using cloud service providers such as Amazon EC2, and I can scale up and down as much as wanted. Testing is much more convenient in a simulated environment as it is easy to repeat tests and get the same results.

The aim of the simulation is to set up a container-based environment that is running a scalable web service, which will be needing a reverse proxy. There should be an attacker node that can conduct the Yo-Yo attack. The detection mechanism will be located at the load balancer or behind the load balancer, but in front of the container service.

## 3.1 Tools

This section describes the tools used in the project.

- VirtualBox 6.1
- Ubuntu 16.04 Xenial, x86_64, 4.15.0-45-generic, CPU: Intel Core i7-6700HQ @ 2.60 MHz, 1 core.
- Kali GNU/Linux Rolling, release 2021.1
- Docker 20.10.6, build 370c289
- Caddy web server - a lightweight web server with no dependencies.
- HAProxy reverse proxy load balancer
- httperf - a tool for benchmarking and stresstesting a webserver by sending http requests at a desired rate. With httperf one can specify the number of connections, requests per seconds and timeouts among

other options. This tool will be applicable for both the legitimate users in order to simulate benign traffic, as well as for the attacker in order to simulate attack traffic.

- autobench - a perl wrapper to httperf which is suitable for stresstesting as well as conveniently creating data and graphs.

## 3.2 High Level Design

Figure 3.1 shows the high level design of the simulation environment. It includes the legitimate users illustrated as "internet", an attacker, a reverse proxy that can do load balancing, a detection mechanism and the container service.

## 3.3 Test environment

The test environment is an experimental environment aimed at simulating a web service that suffers a Denial of Service attack. The web service runs on a cluster of Docker containers that runs Caddy. The web server will reveive traffic from the nodes "client" and "attacker". The client will send requests at a low rate that does not impact the load on the web server in such a way that it is causing service downtime, laying the ground for what we can call "normal traffic". The attacker node runs on Kali Linux and is specced with tools. It will be attacking wiht a httperf in order to impact the web server in such a way that it will trigger the auto-scaling mechanism to trigger up and down, which is called an Yo-Yo attack.

#### Web-server

The web-server is both running a HAProxy load balancer and the web-server. Both the proxy and the web-server is running Docker containers. The web-server consists of several idle containers in order to simulate an environment that is able to scale according to the incoming load. The HAProxy is doing the load balancing on the web-servers that are active and can also deliver a lot of statistics on the incoming traffic. This is also the entrypoint for incoming traffic which makes it ideal for reading the load and acting on it. The web-server will likely be the point for the detection mechanism as well. Running Ubuntu 16.04 Xenial version 4.15.0-45-generic with a x86_64 architecture and one Intel Core i7-6700HQ @ 2.60 MHz, 1 core.

#### Client

The client runs Ubuntu 16.04 Xenial version 4.15.0-45-generic with a x86_64 architecture and one Intel Core i7-6700HQ @ 2.60 MHz, 1 core. The client simulates the normal and benign traffic load towards the web-server and will be requesting data at a rate of 100 requests per second. The data will be sent with the tool httperf.

#### Attacker node

The attacker node, named "Elmer Fudd" after the Looney Tunes character, is working as the attacker node. Just as Elmer shoots bursts with his shotgun, so does the Yo-Yo attack. The Yo-Yo attack sends bursts of overload traffic in order to impact the web-service as well as triggering the auto-scaling function. The attacker node runs Kali Linux which is a Debian distro. It's running version 5.10.13-1kali1 with a x86_64 architecture and a Intel Core i7-6700HQ @ 2.60 MHz, 1 core CPU. It will be using httperf and/or slowhttptest to conduct the attack.

## 3.4 Data

What data do I want to find? The data that is important to find is the detection rate; how many percentage of the attacks is the detection mechanism able to detect?

What needs to be measured in order to detect attacks?

- Load - load is a trigger for the auto-scaling function to scale either up or down. Load is measured by CPU usage and the auto-scaling function typically scales up at 80% CPU usage.

- Auto-scaling triggers - whenever the auto-scaler function scales the system up should be noted by the detection mechanism in order to throw alerts. For example if there is a scale-up and scale-down within a short interval it should be noted.

- Normal behaviour - what does the system look like under a non-attack scenario. This is important in order to detect abnormalities situations.

## 3.5 Auto-scaling

Auto-scaling script. To simulate an auto-scaling functionality, a custom script has been created. The script reads statistics from the HAProxy stats web page to read how many active connections per seconds that are currently connected to the frontend. The script will start docker containers accordingly when the number of active connections to the frontend reaches certain active connections per seconds thresholds. The decisions are simple if statements.

## 3.6 How to simulate it in a VirtualBox environment

Since this experiment is built in VirtualBox it means that the hardware is solely based on one single computer and it gets tricky to overload the victim server as this may cause performance issues on the host machine that runs the VMs, as well as it is hard to add extra computing power because there aren't any.

## 3.7 How to detect

In order to detect the attack the detection system will be broken into two. One component keeps track of the auto-scaling and counts how many times it has been triggered recently. This scale_counter will trigger on predefined threshold that will start alerting when the auto-scaler has been triggered up and down to many times within a certain time period. The second component keeps track of IP addresses that visits the web server. The purpose of keeping track of them is to keep an extra eye on addresses that are sending traffic in the scale-up phase and NOT in the scale-down phase. The whole purpose of the Yo-Yo attack is to send data in bursts, and if the attacker is supposed to stop sending traffic in the scale-down phase, then there should be easy to figure out which address that belong to the attacker.

## 3.8 Load test

In order to get a feeling of how much load the webserver/HAProxy can handle it is appropriate with a load test. In this experiment, autobench is used [5]. The test shows that the web server is being saturated at approximately 600-700 requests per second. The reason for the increase in response time as shown in figure 3.2, is most likely due to sufficient resources on either of the VMs' capability to deal with the high load due

CPU utilization. Both the VMs are running the same hardware so the bottleneck may be on either one of them.

There might also be an issue with the configuration of either HAProxy or Caddy web server as other popular web servers such as apache and nginx have a max connection limit in order to protect the server from running out of resources. As of the test results it has been configured in the HAProxy configuration file that it is allowed with up to 10000 concurrent connections in order to remove this as a possibility to ruin the test data.

## 3.9 Limitations to this test

Running VMs on a single host means that all VMs run on the same hardware. THis means that this experiment is limited in that it can't provide extra resources to the attacker or the victim, especially extra resources for the victim in case of an attack. A single host can only have a limited number of TCP sockets and file descriptors occupied at a time.

The web server in this experiment struggles to cope with requests at a rate of 600 requests per second (600 r/s) as shown in 3.2.

The diagram is created by the tool autobench which includes a script for converting csv or tsv files to postscript files which are graphical data that can be visualized with the gv program in linux or adobe acrobat reader on other platforms.

## 3.10 CPU utilization

It's hard to determine the CPU utilization on this kind of experiment environment, as all of the hosts eventually are running of the same hardware, therefore it may be necessary to meassure how much CPU the HAProxy container on the web server node is utilizing before a scaling trigger should happen. After stress testing/load testing the web server, it was found that the CPU utilization of the HAProxy container was at 15-16% usage when the traffic input was at 600 request/second. At incoming 600 requests/second, the web server VM gets really sluggish and by running the top command one can see that all the processes approximately make up to 100% CPU utilization.

As seen in 3.3 the total CPU usage is actually more than 100%. By summing the CPU utilization of the active processes listed by the top command, the total CPU usage is 100,9%.

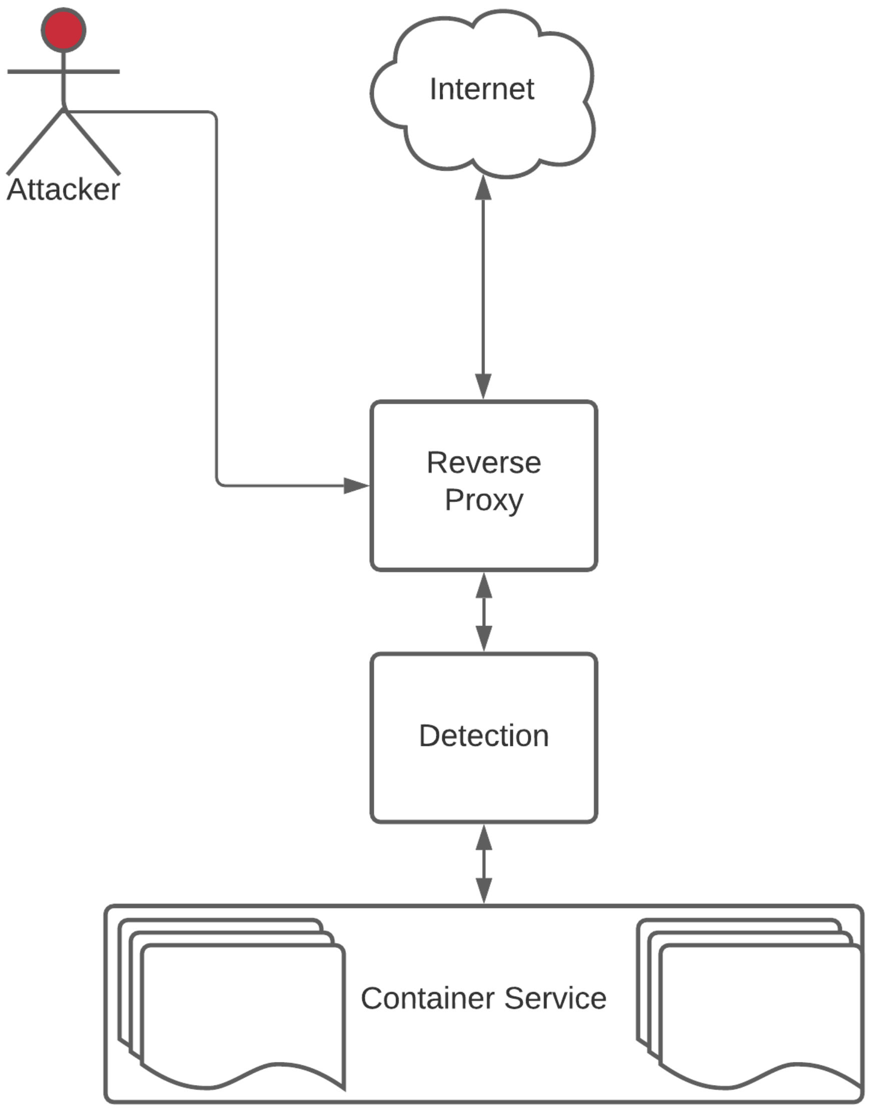


Figure 3.1: High level design of the simulation environment.

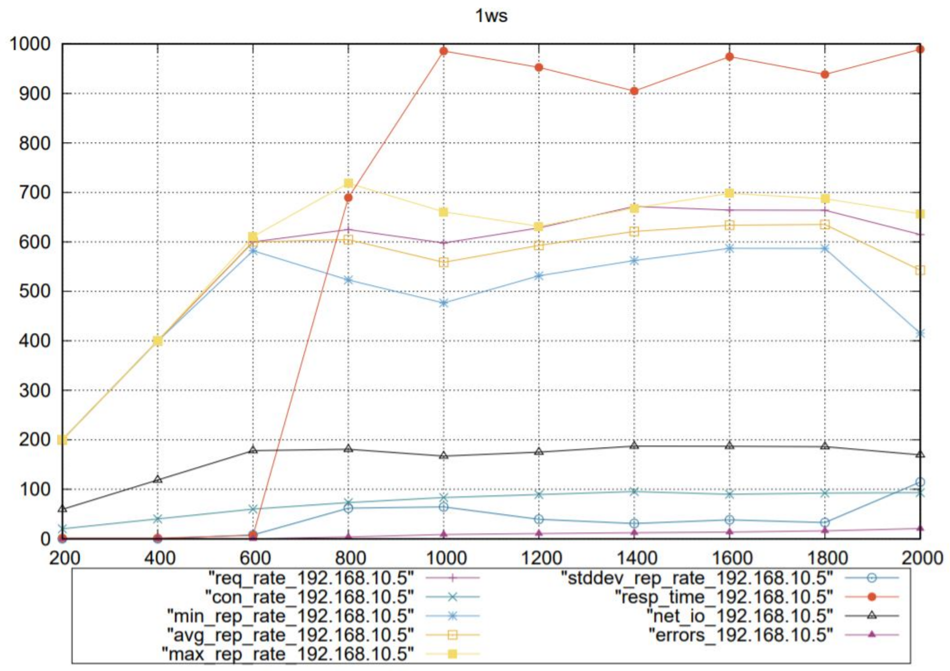
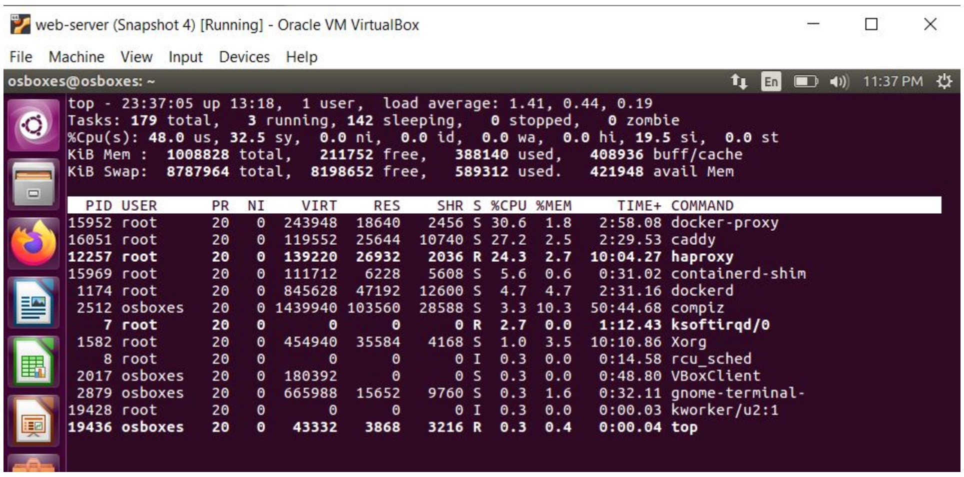


Figure 3.2: Load test of the experimental environment.


web-server (Snapshot 4) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
osboxes@osboxes: ~
En
11:37 PM

```
top - 23:37:05 up 13:18,  1 user,  load average: 1.41, 0.44, 0.19
Tasks: 179 total,   3 running, 142 sleeping,   0 stopped,   0 zombie
%Cpu(s): 48.0 us, 32.5 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi, 19.5 si,  0.0 st
KiB Mem :  1008828 total,   211752 free,   388140 used,   408936 buff/cache
KiB Swap:  8787964 total,  8198652 free,   589312 used.   421948 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
15952 root      20   0  243948  18640   2456 S 30.6  1.8   2:58.08 docker-proxy
16051 root      20   0  119552  25644  10740 S 27.2  2.5   2:29.53 caddy
12257 root      20   0  139220  26932   2036 R 24.3  2.7  10:04.27 haproxy
15969 root      20   0  111712   6228   5608 S  5.6  0.6   0:31.02 containerd-shim
 1174 root      20   0  845628  47192  12600 S  4.7  4.7   2:31.16 dockerd
 2512 osboxes   20   0 1439940 103560  28588 S  3.3 10.3  50:44.68 compiz
    7 root      20   0       0      0      0 R  2.7  0.0   1:12.43 ksoftirqd/0
 1582 root      20   0  454940  35584   4168 S  1.0  3.5  10:10.86 Xorg
    8 root      20   0       0      0      0 I  0.3  0.0   0:14.58 rcu_sched
 2017 osboxes   20   0  180392      0      0 S  0.3  0.0   0:48.80 VBoxClient
 2879 osboxes   20   0  665988  15652   9760 S  0.3  1.6   0:32.11 gnome-terminal-
19428 root      20   0       0      0      0 I  0.3  0.0   0:00.03 kworker/u2:1
19436 osboxes   20   0   43332   3868   3216 R  0.3  0.4   0:00.04 top
```

Figure 3.3: Screen shot when running the top command on the web server in case of the server being overloaded.

# Chapter 4

# Results

## 4.1 Assumptions

### Attacker's knowledge of threshold

In this experiment it is assumed that the attacker does know the scaling point of the target web server because he has done his reconnaissance beforehand. From the attacker's perspective it would be ideal to then send requests at a rate of 700-750 req/s because the attacker then know that some of the requests will time out, thus the web server aren't able to reply to all incoming request within a certain time, here the timeout interval is set to 5 seconds. Figure 4.1 shows how the server responds to a Yo-Yo attack with no scaling.

The CPU utilization of the HAProxy container runs at approximately 80% of total CPU power at the same time as the response time of incoming requests are going over 100 ms. This is shown in figure4.2. Here, the CPU usage is measured with the top command at the web server VM which is serving the HAProxy container. There is some noise in the output here, but as previously shown in 3.3 the highest CPU usage the HAProxy container will get is approximately 25%. If we say that 25% is 100% then 20% CPU usage for the HAProxy is equal to 80%. The web server VM is laggy and sluggish when we're at this scenario.

### Scaling point

The web server is assumed to scale the web service if a threshold of load is met. The load threshold has to be met for a certain amount of time in case of input spikes, meaning that random events like misconfigurations or user behaviour could cause spikes that exceeds the threshold for a second

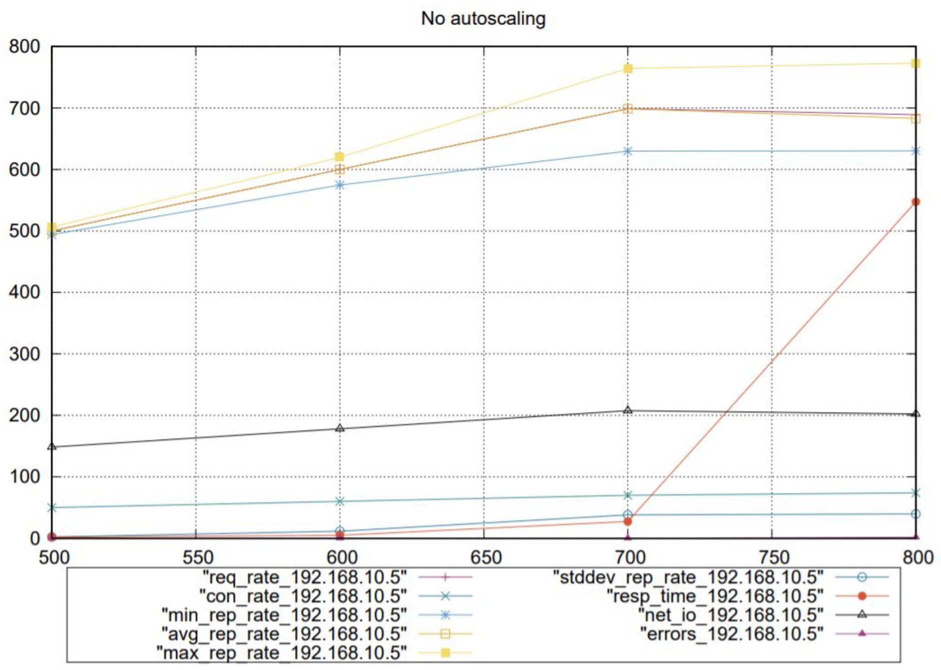


Figure 4.1: Request rate and reply rate under a no scale scenario.

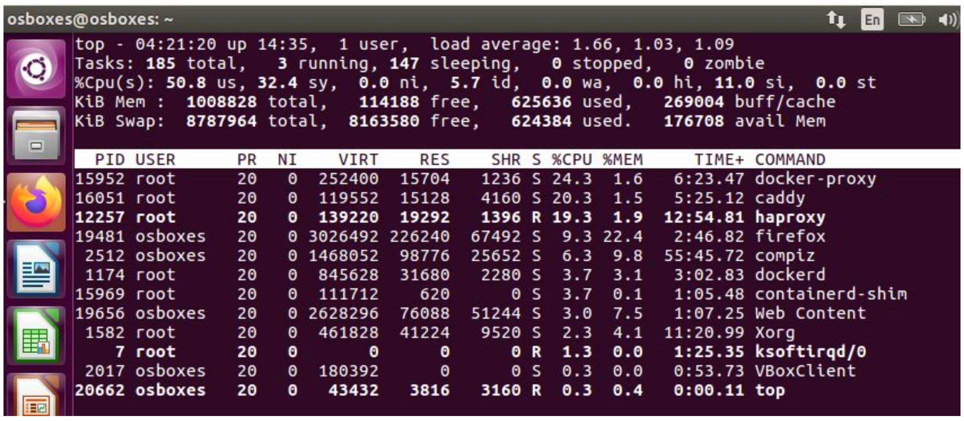
or two, and wont cause enough performance degradation for the need to scale up the web service as the load will decrease quickly.

By these two assumptions, the attacker has to start an attack and increase the rate in order to detect when the web server starts to suffer from the load. The attacker will detect this by response time from the web server.

## 4.2 Attack with auto-scaling

Since this project isn't able to summon more compute resources this is an assumption of how the Yo-Yo attack would have made an impact on this setup. Assuming that expanding the web server by one container that could lift the bandwidth of the web server by the double up to 1400 req/s, then the graph will look a little more linear. The reply rate will follow the request rate, and the response time will stay below 100 ms. There will probably be a minor spike in response time, but the instantiating of one additional container is done so quickly that the attacker won't notice it. This is shown in figure 4.3.

The reason that the assumption is valid is that in the demonstration of the Yo-Yo attack by Bremler-barr et al. they set the start for the off-attack phase when the response time of the web site is at 1000 ms. The reason they get such a high response time may be due to other many factors such



```
osboxes@osboxes: ~
top - 04:21:20 up 14:35,  1 user,  load average: 1.66, 1.03, 1.09
Tasks: 185 total,   3 running, 147 sleeping,   0 stopped,   0 zombie
%Cpu(s): 50.8 us, 32.4 sy,  0.0 ni,  5.7 id,  0.0 wa,  0.0 hi, 11.0 si,  0.0 st
KiB Mem :  1008828 total,   114188 free,   625636 used,   269004 buff/cache
KiB Swap:  8787964 total,  8163580 free,   624384 used.   176708 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
15952 root      20   0  252400  15704   1236 S 24.3  1.6   6:23.47 docker-proxy
16051 root      20   0  119552  15128   4160 S 20.3  1.5   5:25.12 caddy
12257 root      20   0  139220  19292   1396 R 19.3  1.9  12:54.81 haproxy
19481 osboxes   20   0 3026492 226240  67492 S  9.3 22.4   2:46.82 firefox
 2512 osboxes   20   0 1468052  98776  25652 S  6.3  9.8  55:45.72 compiz
 1174 root      20   0  845628  31680   2280 S  3.7  3.1   3:02.83 dockerd
15969 root      20   0  111712    620      0 S  3.7  0.1   1:05.48 containerd-shim
19656 osboxes   20   0 2628296  76088  51244 S  3.0  7.5   1:07.25 Web Content
 1582 root      20   0  461828  41224   9520 S  2.3  4.1  11:20.99 Xorg
    7 root      20   0       0      0      0 R  1.3  0.0   1:25.35 ksoftirqd/0
 2017 osboxes   20   0  180392      0      0 S  0.3  0.0   0:53.73 VBoxClient
20662 osboxes   20   0   43432   3816   3160 R  0.3  0.4   0:00.11 top
```

Figure 4.2: CPU utilization of HAProxy under an attack with no auto-scaling in use. The top command shows approximately 20% which is transitioned into 80% usage.

as travel time, but the main factor is that the experimental evironment in their resarch used VMs which we have learned has a much higher boot time than containers. Bremler-Barr et al.[9] mentions that the boot time for their VMs are 1 minute plus 2 minutes for warming up the VM, and there you have a 3 minute window for the attacker to meassure the response time.

In our case of a container-based web service the containers hardly use any time at all to instantiate. The boot time can hardly be called a boot time, because the containers have very few dependencies and do not have to run their own OS and kernel, thus making the overhead of scaling containers low, as explained in "An Introduction to Docker and Analysis of its Performance" by Rad et al.[7].

## 4.3 Detecting the adversarial requests

The main goal of this article is to detect the malicious user attacking the web server with a Yo-Yo Denial-of-Service attack. In order to do so it is convenient to examine the logs that HAProxy produce. Since the HAProxy is running in a container, the logs have to be accessed by forwarding logs to Docker with stdout or by sending them to rsyslog. Haproxy.log offer access log lines of each request which makes it possible to count how many occurrences of an IP there is. If we compare the number of occurrences of one IP address related to the scale up and scale down phases, it should be possible to detect an attack.

For transparency, this is the pseudocode for the detection script shown in 4.4.

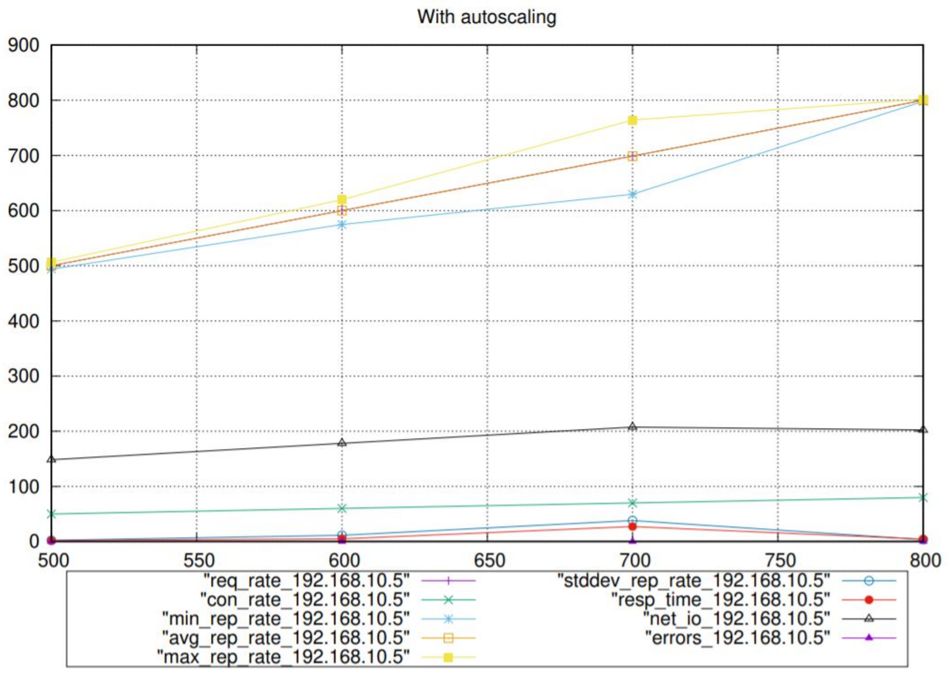
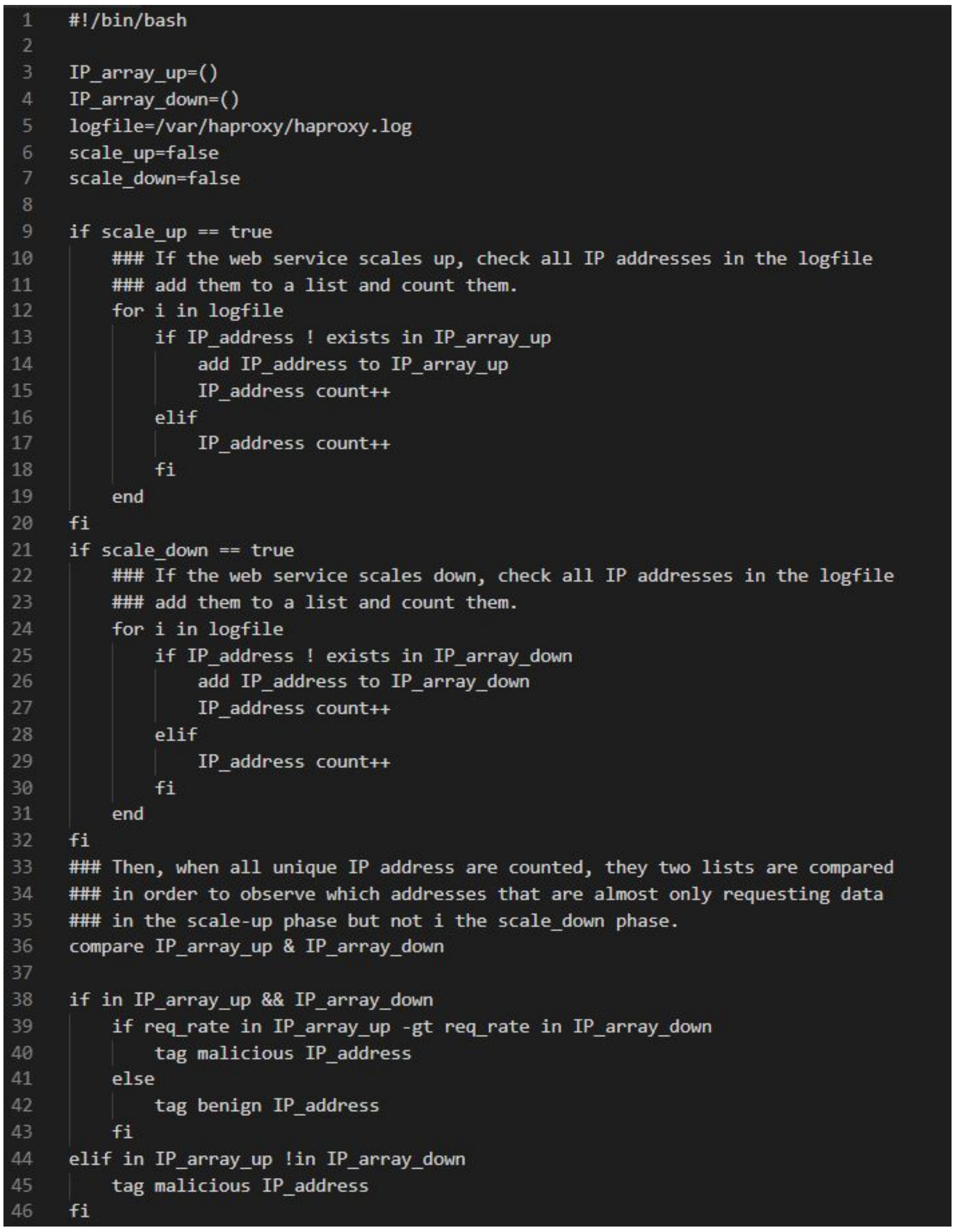


Figure 4.3: The assumed load of web service with auto-scaling. The response time stay below 100 ms and is hardly noticeable.



```
#!/bin/bash

IP_array_up=()
IP_array_down=()
logfile=/var/haproxy/haproxy.log
scale_up=false
scale_down=false

if scale_up == true
    ### If the web service scales up, check all IP addresses in the logfile
    ### add them to a list and count them.
    for i in logfile
        if IP_address ! exists in IP_array_up
            add IP_address to IP_array_up
            IP_address count++
        elif
            IP_address count++
        fi
    end
fi
if scale_down == true
    ### If the web service scales down, check all IP addresses in the logfile
    ### add them to a list and count them.
    for i in logfile
        if IP_address ! exists in IP_array_down
            add IP_address to IP_array_down
            IP_address count++
        elif
            IP_address count++
        fi
    end
fi
### Then, when all unique IP address are counted, they two lists are compared
### in order to observe which addresses that are almost only requesting data
### in the scale-up phase but not i the scale_down phase.
compare IP_array_up & IP_array_down

if in IP_array_up && IP_array_down
    if req_rate in IP_array_up -gt req_rate in IP_array_down
        tag malicious IP_address
    else
        tag benign IP_address
    fi
elif in IP_array_up !in IP_array_down
    tag malicious IP_address
fi
```

Figure 4.4: Pseudoscript for the detection mechanism.

# Chapter 5

# Discussion

## 5.1 Container vs. Virtual Machine

One thing to think about is whether or not containers are a better choice than Virtual Machines in regard of auto-scaling. In practice, the boot-time/warm-up time of a container is significantly shorter compared to virtual machines. Containers can boot in a couple of seconds, whereas virtual machines could take several minutes to boot. How does this impact the Yo-Yo attack and is this a benefit or a drawback? Hypothetically, an Yo-Yo attack against a container-based web-service could cause even a bigger economic penalty, as the scaling will occur much more frequently. In regards of the performance penalty, it may not be a problem at all. The Yo-Yo attack impacts the performance of a VM-based web service because the it takes time to boot up additional VMs. The performance cost that is demonstrated in Bremler-Barr et al. [9] shows that the Yo-Yo attack is highly efficient at detecting warm-up and cool-down phases of the web service, and the short moment the web service is suffering from a too high load, the service becomes unavailable or "slugghish" for the users. If the provider of the web service where to use containers in stead of VMs, it may not be a performance issue at all since additional containeres would be started almot instantly [7].

## 5.2 DoS vs. DDoS

There are some scenarios where the solution that is propesed in this paper won't be sufficient to detect an Yo-Yo attack. One such scenario is in a DDoS attack where the attacker may send bursts from many different machines. If the detection system is solely based on keeping an extra eye out for addresses that sends overload traffic in the scale-up phase, that same address may not appear again in the next scale-up phase.

## 5.3 Other variants of the attack

Randomized attack patterns - e.g. by also sending traffic in the off-phase of the attack in order to camouflage the IP addresses.

# Chapter 6

# Conclusion

The work done in this article is a simple but novel project as there hasn't been conducted a research for a detection mechanism on Yo-Yo attack on a container-based environment. In fact, there hasn't been done much research on the Yo-Yo attack since Bremler-Barr et al.[9] demonstrated the attack for the first time in 2016. Though, it is highly interesting since this is an attack that exploits a functionality that first and foremost is a counter mechanism to high loads and fluctuating loads, which in turn is a nice tool in order to adjust to Denial-of-Service attacks.

The detection mechanism proposed in this research is simple but accurate in detecting an Yo-Yo attack in this particular setup. The experimental setup is somewhat artificial as it all runs on the same hardware, whereas a cloud environment would have a surplus of hardware just waiting to be paid for.

The containers seems to be a benefit in countering the Yo-Yo attack as the attacker not necessarily is able to detect when the scaling happens. This is because the scaling process happens in a few seconds, so if the attacker should catch when the switch has been flipped, then he must send probing packets often, or be able to act upon response time quickly. Acting upon response times may be hard since response times always are relatively fluctuating due to the whole network chain.

# Bibliography

[1] URL: https://trends.google.com/trends/explore?date=2007-02-04%202021-03-04&q=cloud%20computing. (accessed: 04.03.2021).

[2] URL: https://res.cloudinary.com/practicaldev/image/fetch/s--jmQCnO87--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https://dev-to-uploads.s3.amazonaws.com/i/tt6b2sau1mdcg73qt4iz.png. (accessed: 19.03.2021).

[3] URL: https://imengine.public.prod.agp.infomaker.io/?uuid=460fd77e-8ded-51cf-a9de-6b03cfb47226&function=fit&type=preview&source=false&q=75&maxsize=1500&scaleup=1. (accessed: 09.04.2021).

[4] URL: https://www.coindesk.com/price/bitcoin. (accessed: 16.04.2021).

[5] URL: https://github.com/menavaur/Autobench. (accessed: 14.04.2021).

[6] Abhishek Agarwal et al. 'Detection and mitigation of fraudulent resource consumption attacks in cloud using deep learning approach'. In: *Journal of Information Security and Applications* 56 (2021), p. 102672. ISSN: 2214-2126. DOI: https://doi.org/10.1016/j.jisa.2020.102672. URL: https://www.sciencedirect.com/science/article/pii/S2214212620308243.

[7] Babak Bashari Rad, Harrison Bhatti and Mohammad Ahmadi. 'An Introduction to Docker and Analysis of its Performance'. In: *IJCSNS International Journal of Computer Science and Network Security* 173 (Mar. 2017), p. 8.

[8] BBC. *Amazon 'thwarts largest ever DDoS cyber-attack'*. 2020. URL: https://www.bbc.com/news/technology-53093611. (accessed: 25.02.2021).

[9] A. Bremler-Barr, E. Brosh and M. Sides. 'DDoS attack on cloud auto-scaling mechanisms'. In: *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. 2017, pp. 1–9. DOI: 10.1109/INFOCOM.2017.8057010.

[10] CERT/CC. *[linux-security] CERT Advisory CA-96.21 - TCP SYN Flooding and IP Spoofing Attacks*. 1996. URL: https://listman.redhat.com/archives/linux-security/1996-December/msg00006.html. (accessed: 22.04.2021).

[11] Catalin Cimpanu. *Google says it mitigated a 2.54 Tbps DDoS attack in 2017, largest known to date*. 2020. URL: https://www.zdnet.com/article/google-says-it-mitigated-a-2-54-tbps-ddos-attack-in-2017-largest-known-to-date/. (accessed: 19.04.2021).

[12] Cloudflare. *Application Layer DDoS Attack*. URL: https://www.cloudflare.com/learning/ddos/application-layer-ddos-attack/. (accessed: 23.03.2021).

[13] Cloudflare. *Famous DDoS attacks | The largest DDoS attacks of all time*. URL: https://www.cloudflare.com/learning/ddos/famous-ddos-attacks/. (accessed: 19.04.2021).

[14] Cloudflare. *How Do Layer 3 DDoS Attacks Work? | L3 DDoS*. URL: https://www.cloudflare.com/learning/ddos/layer-3-ddos-attacks/. (accessed: 19.03.2021).

[15] Cloudflare. *HTTP Flood Attack*. URL: https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/. (accessed: 23.03.2021).

[16] Cloudflare. *NTP Amplification DDoS Attack*. URL: https://www.cloudflare.com/learning/ddos/ntp-amplification-ddos-attack/. (accessed: 25.03.2021).

[17] Cloudflare. *Smurf DDoS Attack*. URL: https://www.cloudflare.com/learning/ddos/smurf-ddos-attack/. (accessed: 19.03.2021).

[18] Cloudflare. *SYN Flood Attack*. URL: https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/. (accessed: 19.03.2021).

[19] Cloudflare. *UDP Flood Attack*. URL: https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/. (accessed: 23.03.2021).

[20] Cloudflare. *What is a Denial-of-Service (DoS) Attack?* URL: https://www.cloudflare.com/learning/ddos/glossary/denial-of-service/. (accessed: 25.02.2021).

[21] Cloudflare. *What is a low and slow attack?* URL: https://www.cloudflare.com/learning/ddos/ddos-low-and-slow-attack/. (accessed: 25.03.2021).

[22] Sam Cook. *DDoS attack statistics and facts for 2018-2021*. 2021. URL: https://www.comparitech.com/blog/information-security/ddos-statistics-facts/. (accessed: 25.02.2021).

[23] Shane Huntley. *How we're tackling evolving online threats*. 2020. URL: https://blog.google/threat-analysis-group/how-were-tackling-evolving-online-threats. (accessed: 19.04.2021).

[24] Imperva. *Smurf DDoS attack*. URL: https://www.imperva.com/learn/ddos/smurf-attack-ddos/. (accessed: 19.03.2021).

[25] Kaspersky. *About us*. URL: https://www.kaspersky.com/about. (accessed: 15.04.2021).

[26] Kaspersky. *DDoS attacks in Q1 2020*. 2020. URL: https://securelist.com/ddos-attacks-in-q1-2020/96837/. (accessed: 15.04.2021).

[27] Kaspersky. *DDoS attacks in Q2 2020*. 2020. URL: https://securelist.com/ddos-attacks-in-q2-2020/98077/. (accessed: 15.04.2021).

[28] Kaspersky. *DDoS attacks in Q3 2020*. 2020. URL: https://securelist.com/ddos-attacks-in-q3-2020/99171/. (accessed: 19.03.2021).

[29] Kaspersky. *DDoS attacks in Q4 2016*. 2017. URL: https://securelist.com/ddos-attacks-in-q4-2016/77412/. (accessed: 16.04.2021).

[30] Kaspersky. *DDoS attacks in Q4 2017*. 2018. URL: https://securelist.com/ddos-attacks-in-q4-2017/83729/. (accessed: 16.04.2021).

[31] Kaspersky. *DDoS attacks in Q4 2018*. 2019. URL: https://securelist.com/ddos-attacks-in-q4-2018/89565/. (accessed: 15.04.2021).

[32] Kaspersky. *DDoS attacks in Q4 2019*. 2020. URL: https://securelist.com/ddos-report-q4-2019/96154/. (accessed: 15.04.2021).

[33] Kaspersky. *DDoS attacks in Q4 2020*. 2021. URL: https://securelist.com/ddos-attacks-in-q4-2020/100650/. (accessed: 13.04.2021).

[34] Kaspersky. *What is a Botnet?* URL: https://usa.kaspersky.com/resource-center/threats/botnet-attacks. (accessed: 23.03.2021).

[35] Z. Li et al. 'Exploring New Opportunities to Defeat Low-Rate DDoS Attack in Container-Based Cloud Environment'. In: *IEEE Transactions on Parallel and Distributed Systems* 31.3 (2020), pp. 695–706. DOI: 10.1109/TPDS.2019.2942591.

[36] M. Mao and M. Humphrey. 'A Performance Study on the VM Startup Time in the Cloud'. In: *2012 IEEE Fifth International Conference on Cloud Computing*. 2012, pp. 423–430. DOI: 10.1109/CLOUD.2012.103.

[37] Paul Nicholson. *Five Most Famous DDoS Attacks and Then Some*. 2020. URL: https://www.a10networks.com/blog/5-most-famous-ddos-attacks/. (accessed: 19.04.2021).

[38] Fabian Skalleberg Nilsen. *Telenor presset for millionbeløp etter dataangrep*. 2020. URL: https://e24.no/teknologi/i/9OvRPd/telenor-presset-for-millionbeloep-etter-dataangrep. (accessed: 09.04.2021).

[39] Parmy Olson. *The Largest Cyber Attack In History Has Been Hitting Hong Kong Sites*. 2014. URL: https://www.forbes.com/sites/parmyolson/2014/11/20/the-largest-cyber-attack-in-history-has-been-hitting-hong-kong-sites/?sh=bc9b4ab38f6e. (accessed: 20.04.2021).

[40] Matthew Prince. *Deep Inside a DNS Amplification DDoS Attack*. 2012. URL: https://blog.cloudflare.com/deep-inside-a-dns-amplification-ddos-attack/. (accessed: 25.03.2021).

[41] Iain Thomson. *Conversation with Eric Schmidt hosted by Danny Sullivan*. 2006. URL: https://www.google.com/press/podium/ses2006.html. (accessed: 05.03.2021).

[42] Iain Thomson. *World's biggest DDoS attack record broken after just five days*. 2018. URL: https://www.theregister.com/2018/03/05/worlds_biggest_ddos_attack_record_broken_after_just_five_days/. (accessed: 25.02.2021).

[43] Xiaoqiong Xu et al. 'Towards Yo-Yo attack mitigation in cloud auto-scaling mechanism'. In: *Digital Communications and Networks* 6.3 (2020), pp. 369–376. ISSN: 2352-8648. DOI: https://doi.org/10.1016/j.dcan.2019.07.002. URL: https://www.sciencedirect.com/science/article/pii/S2352864819301440.

[44] Kejiang Ye et al. ‘Fault Injection and Detection for Artificial Intelligence Applications in Container-Based Clouds’. In: *Cloud Computing – CLOUD 2018*. Ed. by Min Luo and Liang-Jie Zhang. Cham: Springer International Publishing, 2018, pp. 112–127. ISBN: 978-3-319-94295-7.