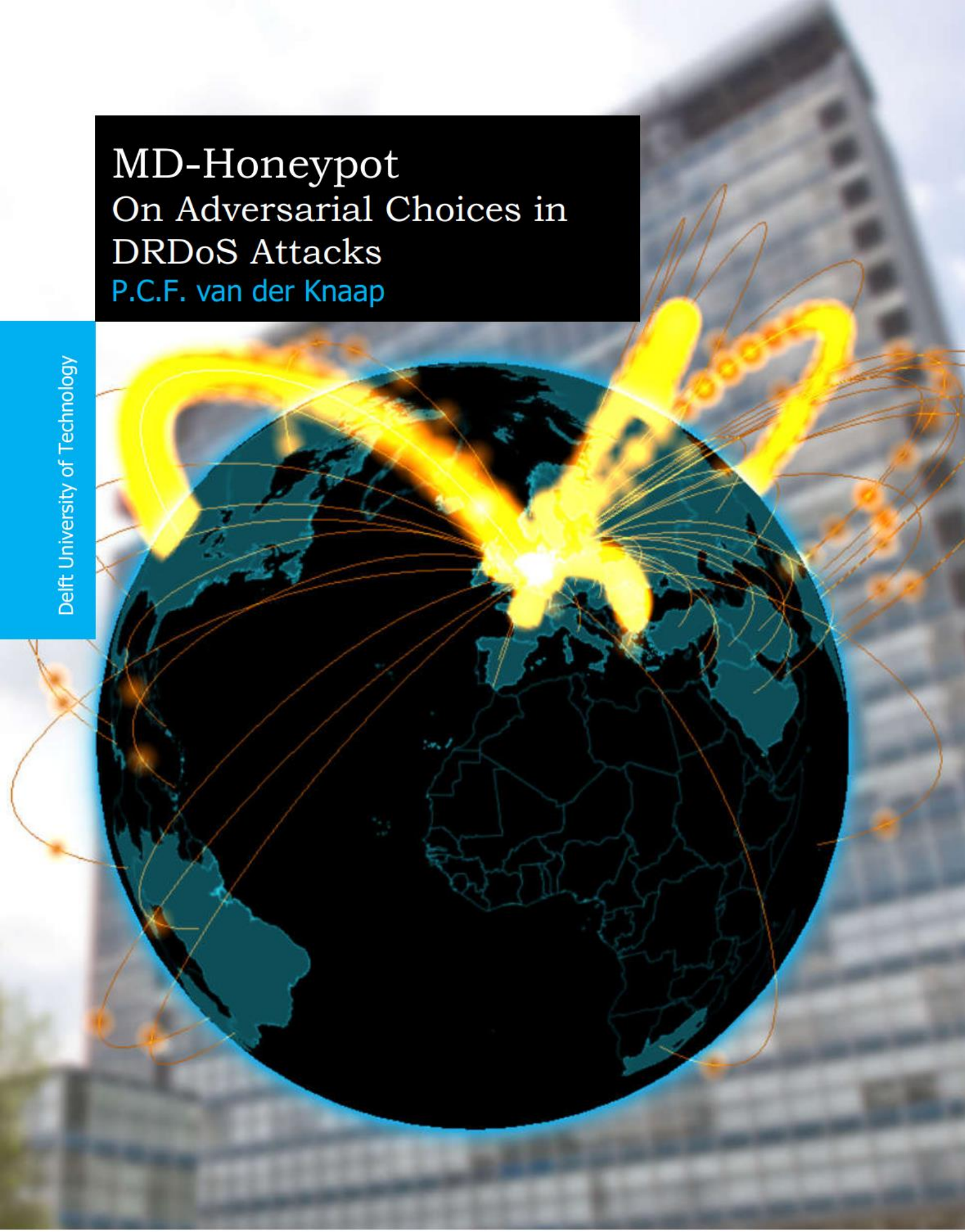
MD-Honeypot
On Adversarial Choices in DRDoS Attacks

P.C.F. van der Knaap

Delft University of Technology



TU Delft Delft University of Technology

Challenge the future

MD-Honeypot

On Adversarial Choices in DRDoS attacks

by

P.C.F. van der Knaap

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on June 23rd, 2020

Project duration:	November 2018 - May 2020	
Thesis committee:	Dr. C. Doerr,	TU Delft, supervisor & committee chair
	Dr. S. Picek	TU Delft, committee member
	Dr. C. Lofi	TU Delft, committee member

TU Delft

Preface and Acknowledgements

This thesis has been written for the partial fulfillment of the Master's thesis Computer Science at the Delft University of Technology. Throughout my thesis I have often thought, "If I were an adversary, what would I do given the knowledge that we have?" This led to interesting points of view and reasoning that has helped me perform this research. In addition, the research process throughout this thesis has enriched me by teaching me new critical ways of thinking.

I want to extend special thanks to my supervisor Christian Doerr and Ph.D. student Harm Griffioen for their support and supervision throughout the thesis. The creation of this thesis would also not have been possible without working closely together with Bart de Jonge and Ahmet Gudek on the developed distributed honeypot network. I especially want to thank all the students and friends that have helped me conduct this experiment by crowd-sourcing the resources required for this research. Without them, this work could not have been completed.

This is also the case for Dereck Bridié, who also deserves a special acknowledgment. Since our bachelor thesis, we have battled our way through difficult courses and projects that felt like they would never end. With our mutual perseverance, the end result was always the same: we made it. Lastly, I would like to thank Tim, Jean, my girlfriend, friends, family, and my fellow students for their continuous encouragement and support whilst working on my thesis. Without them, I would not have made it through the end of the thesis.

Finally, I would like to thank you, the reader, for taking the time to read this thesis report.

P.C.F. van der Knaap
Delft, May 2020

Abstract

The Internet has grown from a few interconnections of trusted parties to an incredibly large network with many different use cases. While the Internet grew, threats emerged as well. Although there are many different threats on the Internet, Distributed Denial of Service (DDoS) attacks are a threat that keeps rising in the threat landscape. The asymmetry between adversaries and defenders is enormous - whereas DDoS attack can be started for less than 5$, DDoS prevention takes up the majority of operational cost in data centers and damages are in the billions.

Thus, it is of vital importance that more effective methods of DDoS prevention are found and implemented to improve defensive effectiveness and to reduce costs. DDoS attacks consist of various types, but the largest share of DDoS attacks are of the subtype Distributed Reflected Denial of Service (DRDoS) attacks.

Adversaries that execute DRDoS attacks use vulnerable servers to create incredibly large attacks and to stay anonymous. Previous work by Rossow showed us which vulnerable services are typically used for these DRDoS attacks, and how well these vulnerable services are exploitable for these attacks. However, we do not know why an adversary uses one vulnerable service but not another. Thus, this work fills that research gap by researching how adversaries react to differently configured vulnerable services, using a large scale experiment.

This work shows that the amplification factor of a honeypot is a primary factor that determines whether an adversary will use a vulnerable server in an attack or not. This work also shows that attackers do not distinguish between regular vulnerable servers and their obvious honeypot counterparts. Furthermore, the response time of a system is of no influence, and some honeypots with packet loss attract fewer adversaries. Additionally, different attacks can be detected while using different service providers and geographical locations for honeypot deployment. Finally, the MD-honeypot framework that was developed for this research may be further developed into fully-fledged DRDoS mitigation software.

Contents

1

Introduction

The Internet has evolved from a small network consisting only of some connected hosts towards a worldwide network connecting billions of devices together. The design and first protocols that the Internet relied on during its construction were not focused on security but merely on (future) functionality [1]. However, due to backward compatibility and lack of maintenance, many of the old protocols and services that were designed decades ago remain in use. For example, in practice, this happens when hardware devices are equipped with firmware and are connected to the Internet, but then receive no further updates or are no longer updated by system administrators.

Meanwhile, the Internet has grown from a few interconnections of trusted parties to an incredibly large network with many different use cases. It is no longer used to simply exchange information between several universities but is now a fundamental part of our lives. Nowadays, there are businesses that only conduct business over the Internet, for example, webshops. In some countries, consumers even shop for groceries online. The amount of time that users spend on the Internet daily has also been on the rise for years [2]. Currently, the average Internet user spends more than six hours online each day [3]. The United Nations has even declared having access to the Internet to be a human right [4]. Already in 2007, Business Roundtable [3,5] stated that an Internet disruption will affect nearly every company in the U.S. directly or indirectly. In conclusion, our reliance on a functional Internet has grown tremendously since the Internet was developed in the early years.

While the Internet grew, threats emerged as well. Adversaries started sending spam, hacking websites, cracking and stealing information, started disinformation campaigns, or try to lure innocent visitors into handing over their personal information. Although these threats are certainly concerning, there is one threat that keeps rising in the threat landscape [6]: Distributed Denial of Service (DDoS) attacks. It is of vital importance that we are able to protect ourselves against these attacks.

In a nutshell, during a DDoS attack, an adversary attempts to disrupt normal traffic of a targeted server, service, or network by overwhelming the target or its surrounding infrastructure with a flood of Internet traffic [7]. While the network is congested, legitimate visitors are no longer able to make use of that service, as the service appears to be offline or unreachable.

There are different types of DDoS attacks, but this thesis will focus on the subtype: Distributed Reflected Denial of Service (DRDoS) attacks. The main difference between this subtype and DDoS attacks is that other (vulnerable) online services are used as a reflector and amplifier of the DDoS traffic. The technical details and terminology of such an attack are discussed in Chapter 2. In 2014, Arbor Networks [8] showed that the majority (65%) of all DDoS attacks were of the DRDoS subtype. Currently, its share is even higher - at 75% [9]. This means that finding solutions and information to better protect against DRDoS attacks is very valuable, as these solutions and information are directly applicable to the majority of DDoS attacks.

Before looking at better ways to defend against DDoS attacks, it is important to understand how DDoS attacks have evolved and how they were turned into the very effective cyber weapon they are today. This information will help us comprehend the magnitude of the problem and select relevant areas to focus on in this thesis.

Evolution of DDoS attacks

The first noted DDoS attack took place as early as 1999 on a very small scale and was targeted at a single university computer [10]. The first published attacks that were covered by the press occurred in February 2000, when CNN, Yahoo, Amazon, and other major websites were hit [11]. Their combined losses were estimated to be in the millions of dollars. The adversary in that case was a juvenile Canadian who reportedly was not even aware he was executing these attacks [12,13].

After these initial incidents, a first countermeasure was already discussed and published under the name 'BCP38'[1]. With BCP38, Internet Providers take an active role in preventing traffic that is used to generate DDoS attacks to exit their network. However, this countermeasure only works if all networks apply the countermeasure. This is because the countermeasure ensures that generated DDoS traffic cannot exit a protected network. However, if there are still unprotected networks, the generated DDoS traffic is still able to exit the network of a provider, whereafter the countermeasure can no longer be applied. Full-scale adoption of BCP38 was slow, and currently, after 20 years, more than 20% of the Internet remains unprotected [14].

Since the first accidental DDoS attacks, DDoS attacks have also been used on purpose, as a means to an end. There are several motivators for adversaries to employ DDoS attacks: political or other ideological differences, feuds, adversaries proving their capabilities, online gaming, diversions, and extortion [6,8,15]. Some examples will be given to illustrate the dangers of these motivators.

Politically motivated attacks have been used as part of cyber warfare for a long period of time. In 2001, the Chinese hacking group "Honker Union" was believed to be behind attacks on U.S. military Internet sites after tensions increased when a U.S. Navy spy plane had to make an emergency landing in Chinese territory [16]. In 2007, Russian actors were to be believed behind attacks on the websites of the Estonian Government and private sector websites. These attacks started after a disagreement about the removal of a Soviet soldier statue among the Estonians and Estonia's partial ethnic Russian population. Reports stated that these attacks continued for several weeks, blocking access to online services of Estonian banks and causing bank machines to become temporarily unavailable [17]. An attack in 2011 targeted South Korean government websites and U.S. military forces in Korea to test their preparedness for cyber attacks. During the Crimea tensions, NATO was attacked, rendering its websites offline [18].

A more recent political-related DDoS attack occurred in 2019. During the Hong Kong anti-extradition protests, massive DDoS attacks hit the messaging app Telegram [19]. Telegram specifically was targeted since protesters used the messaging application to communicate and coordinate their movements. Telegram's CEO accused the Chinese Government of being behind the DDoS attack [20].

The effects of politically motivated attacks can be severe. One of the reasons for this is that attribution of a politically motivated attack can be sensitive. In 2009, North Korea was suspected to be behind attacks on South Korean and U.S. websites [16] - the South Korean intelligence services shared their suspicion through the press. A U.S. lawmaker used this information to call for retaliation on North Korea while having no proof [16] as to whether North Korea was actually behind the attack. When conventional and cyber warfare become intertwined, these attacks can spiral out of control if great care is not taken.

The bar to execute DDoS attacks is much lower than physical war, and as such, ideological differences sometimes escalate into a cyberwar between organizations or groups. Such a cyber feud, or cyberwar, can escalate very quickly as, for example, in 2010, with the emergence of the Wikileaks scandal. In November 2010, Wikileaks published secret documents originating from the U.S. embassy [21]. This incident led to a divided political landscape followed by multiple heavy DDoS attacks [21,22]. A few days later, PayPal suspended its service to Wikileaks under political pressure. This caused the popular hacktivist group Anonymous to start large DDoS attacks on PayPal. Credit card companies MasterCard and VISA followed in suspending service to Wikileaks and following the suspense received the same treatment from the Anonymous group in response. Although the companies never published reports about the damage caused, the costs for the companies related to the attacks is estimated to be in the millions [23].

DDoS attacks may also be used as a diversion tactic. With this type of tactic, a DDoS attack is launched to draw attention away from the fact that the company is being attacked in another form, for example, a hack where company data is being stolen. For example, in 2011, when Sony was not aware that up to 101 million customer records were stolen, because the company was distracted by DDoS attacks [24]. The effects of such a DDoS attack can result in a huge financial loss for these companies. For example, UK telecom company TalkTalk was also targeted with DDoS attacks in 2015 while hundreds of thousands of pieces of customer data were stolen [24]. In the aftermath of the attack, the company's share price dropped by 22%.

The aforementioned DDoS attacks are suspectedly executed by adversarial groups or governments. However, unsophisticated individuals are also able to start DDoS attacks. In 2018, several Dutch banks and websites had a lot of downtime after a 17-year-old executed multiple attacks [25]. His motivation for launching the attacks was because he thought it was a funny thing to do. He was caught not long after the attacks by making simple mistakes [26]. For the attack, the 17-year-old made use of a DDoS service provider, sometimes also called a 'booter' service. A DDoS service provider offers attacks as a product that is usually paid for by anonymous transactions such as Bitcoin. In 2016, a teen UK hacker was arrested, providing such a booter service [27]. He was 15 years old at the time of creation and earned more than $385.000 in a few years with his service. Recently, Ubisoft brought

[1] https://tools.ietf.org/html/rfc2827.html

several juveniles to court that provided DDoS services to players of an online game called Rainbow Six Siege [28]. The platform provided options to DDoS other players out of an online match so that the player initiating the attack would increase his ranks in an online game.

Since DDoS attacks first started occurring, their sizes have been growing tremendously [29], requiring even more resources to defend against them. In 2013, an attack with a size of 300 gigabits per second hit Spamhaus. Spamhaus is a non-profit anti-spam organization that maintains blacklists of abusers that send a lot of spam. In retaliation for placing a large spam organization on the blacklist, the group hit Spamhaus with a massive attack. This was by then the largest known DDoS attack that ever occurred on the Internet. The attack even took the services of Cloudflare offline, the company that assisted Spamhaus in mitigating the attack [30]. The attack is often referred to as 'the DDoS that almost broke the Internet' [31].

In 2016, a DDoS attack on Dyn, a DNS provider, broke all previous records [32], and this attack resulted in many commonly accessed services not being available, including Reddit, Twitter, Netflix, PayPal, and Spotify [33]. An angry gamer was blamed for being responsible soon after the attacks [34]. This incident resulted in an estimated 4% loss of customers for Dyn [35]. A similar attack occurred a year later, in 2017, when DreamHost's DNS services were also hit, knocking 400.000 websites offline [36]

Another year later, in 2018, another high scale attack occurred, breaking all previous records. GitHub, a website for hosting open-source projects, was hit by an attack peaking at 1.35 terabit per second. In 2019, a client of Imperva received a massive DDoS attack with the highest amount of packets per second ever recorded [37]. These attacks are so large that companies that specialize in protection against such attacks have a lot of trouble blocking these attacks this size [38].

Damages caused by DDoS attacks are hard to calculate. For example, it is unknown what the impact is of the loss of customers for Dyn, as well as how many customers would return later. The same applies to DreamHost, as there are no known numbers of how many customers decided to change their host after the incidents. NETSCOUT [29] recently reported that the damage of a DDoS attack greatly varies, with most commonly (23%) a reported damage between $50.000 and $99.000. More interestingly, only 15% of the attacks caused damages less than $10.000. A recent Dutch report by NBIP [39] estimated the total damage of DDoS attacks to Dutch enterprises to exceed a billion euros in a year's time.

To conclude, DDoS attacks can be executed for a multitude of reasons and can be used as a weapon not only by sophisticated adversaries but also by less sophisticated adversaries such as angry gamers. The sizes of DDoS attacks are still growing and the damage caused by the attacks is significant. Ideally, these attacks should be blocked as closest to the source as possible. A solution for this would be the global adoption of BCP38. However, until it is fully adopted, other defense solutions will need to be created and need to be used in order to combat the multitude of daily incoming attacks.

Defending against DDoS attacks

DDoS attacks are not only relatively easy to execute, but they are also easily accessible. Symantec reports that simple attacks may only cost $5-10 [40]. These low costs to execute an attack create a large gap between the cost to execute an attack and the cost to defend against one, as defending against DDoS attacks is costly [41]. Already in 2014, the largest portion of the operational cost of datacenters was attributed to DDoS events [8]. If the number and size of attacks can be reduced, less money would need to be spent on operational DDoS defense. In addition, the reduction of DDoS attacks could reduce the downtime of services. Less downtime of a service would then mean that more consumers are able to access the company's website and thus lead to more revenue. This brings us to the area of focus for this thesis: gathering intelligence to enable new strategic analyses of adversaries and enable new defense solutions to be built.

Currently, the best way to defend against D(R)DoS attacks is to have information available in order to be able to classify traffic as being or attack traffic. With this information, the classified attack traffic can then be blocked, stopping the attack or preventing the attack from congesting the pipeline further down the line. This information could consist of various characteristics that attack traffic has, but regular traffic does not. An example of such a characteristic is a specific packet that only occurs in a high frequency in attack traffic, and only in a low frequency in normal traffic - or not at all. Once a traffic flow with that specific packet in a high frequency is then detected, the flow can be classified as attack traffic. This information that aids classification is also called Tactical Cyber Threat Intelligence and can be shared among defenders.

With DRDoS attacks, an adversary makes use of vulnerable services on the Internet to anonymize and amplify generated DDoS traffic. By anonymizing the traffic, an adversary ensures that the attack cannot easily be traced back to him, and by amplifying the traffic, it needs to use fewer resources to create a bigger attack.

However, before an adversary can execute its DRDoS attacks, it first needs to find these vulnerable services to use in these attacks. A popular method that is used by adversaries to find these services, is scanning the Internet. During this process, an adversary probes all available Internet addresses. When he finds a vulnerable server, he can save the address, and use that address later

when executing an attack. A technical description and examples of the scanning process are included in Chapter 2. A capable adversary is able to scan all IPv4 Internet addresses in less than an hour [42].

Previous research [43–45] has shown which types of services are typically used as a reflector in DRDoS attacks. These services typically provide a response to a query in a stateless manner. An example of such a service could be a simple weather service: when a user requests the weather, the weather service will respond with the weather forecast. The difference between the length of the query and the response results in the amplification factor - how much larger the response is than the query. The typical amplification factor of these services while amplifying the traffic in these attacks is also known from previous work.

On a high-level, this information shows us *which type* of services are selected and also *how much* these services typically amplify. However, not all services are configured exactly the same, and therefore, some services may respond differently than others. Because these services may all show different behavior, an adversary may choose to prefer one configured service over another. Figure 1.1 shows an overview of this process: an adversary may choose to use one configured weather service, but not two others that it also discovered while scanning.

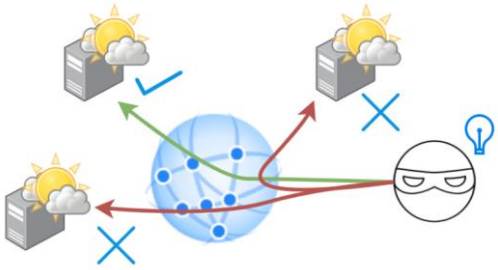


Figure 1.1 - An adversary choosing which services to use to attack a target

In other words, considering that an adversary finds all three services, why would an adversary use one discovered service in an attack but not another? The motivation that drives adversaries to make these choices is currently unknown, and this leaves us with a research gap. Therefore, this thesis will try to fill that gap by answering the following research question:

RQ: Which factors of a vulnerable service influence the abuse by adversaries for Distributed Reflected Denial of Service Attacks?

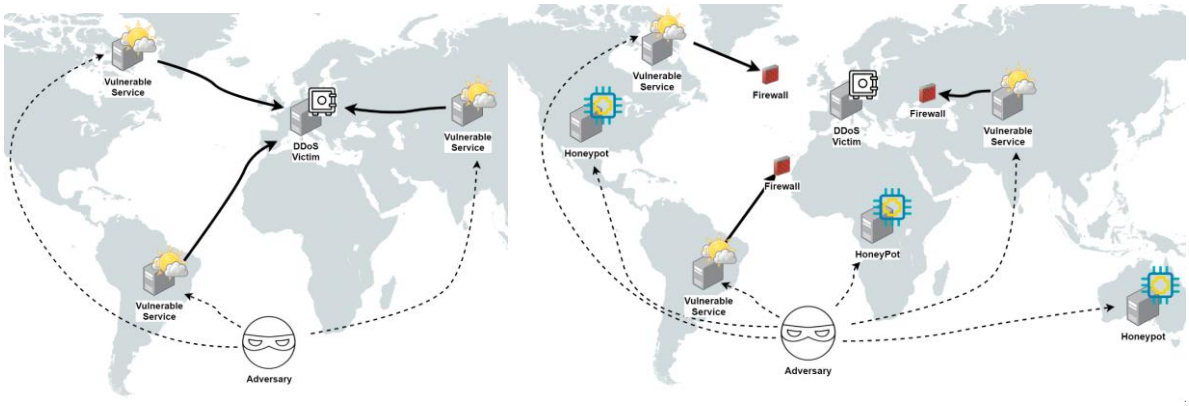
This thesis answers the research question by testing several hypotheses that describe different factors of influence. The main contribution that this research provides is that the amplification factor that a vulnerable service provides is a driving choice for adversaries that determines whether an adversary does or does not use a vulnerable server for his attacks. This work also shows that attackers do not distinguish between regular vulnerable servers and their obvious honeypot counterparts. Furthermore, the response time of a system is of no influence, and some honeypots with packet loss attract fewer adversaries. Additionally, different attacks can be detected while using different service providers and geographical locations for honeypot deployment.

These new findings update the currently available intelligence base of the DDoS adversary landscape and open the path to more effective future research and DDoS solutions. A potential anti-DDoS solution that is now possible due to the gathered knowledge of this thesis is the following:

With real-time information that can be provided by honeypots[2], network traffic could be classified as benign or attack traffic. If this happens in a high-level part of a network, for example, at the level of Internet Service Providers, it provides an early blockade, preventing the traffic from congesting connections further down the line.

Figure 1.2a shows the current situation, where an adversary uses vulnerable services to attack a DDoS victim. Figure 1.2b shows the proposed DDoS defense solution at work. In this case. When an attack is detected (using the honeypots) that targets the victim, traffic that is identical to that of the generated attack traffic may be blocked towards the victim- thus preventing that the line to the systems of the victim will get congested. The system that is engineered during this thesis is called the MD-Honeypot framework and may be extended to be such an effective DDoS defense solution. The answer to the research question of this thesis will help to build attractive honeypots in order to be able to gather the real-time information that is required for the system to work effectively.

[2] Further explained in related work. Honeypots are designed to lure adversaries into using a system or service while recording the behavior of the adversary. By behaving like the system or service that attackers like to use, adversaries are unaware that the system is a honeypot.



(a) (b)

Figure 1.2 - Proposed DDoS defense solution using honeypots

This thesis will first cover the background information that provides more information about DDoS attacks and how these attacks work on a technical level. Chapter 3 covers the related work. Chapter 4 includes the hypotheses and discusses the methodology used to conduct the experiments, the final experiment design, and the test process. Chapter 5 discusses the requirements, design process, and components of the MD-Honeypot framework that was built to test the constructed experiment of this thesis. In Chapter 6, the gathered data is analyzed and evaluated against the hypotheses, and the results are presented. Finally, chapter 7 concludes this thesis and discusses limitations and suggestions for future work.

2

Background

The previous chapter discussed the rise of DDoS attacks from their first occurrence to becoming a significant threat factor in the current Internet. Before diving deeper into DDoS attacks, it is essential to understand how they work on a technical level. This helps to understand how adversaries operate and how we can design experiments to gather intelligence. The first section of this chapter will explain the basic framework and foundations of the Internet. These are the layers that make a DDoS possible on a technical level. If you are familiar with IPv4 and UDP, you may skip the reading of this first subsection. The section thereafter thoroughly explains what DRDoS and DDoS attack are- and what the difference between the two is.

2.1. The Internet

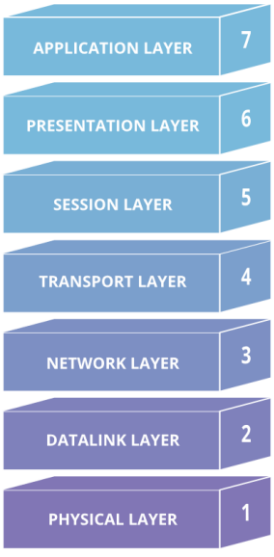
The Internet is often defined as the largest global internet: a large network of smaller computer networks that are all linked together, communicating with the same set of communication protocols. Its large number of interconnections facilitates the ability for anyone or any device to communicate with others and other devices around the world without barriers. However, having many people and devices connected also poses risks, including DRDoS attacks. Before explaining what such an attack is, how it works, and how it is used, this chapter first explains how the Internet works.

OSI Model and TCP/IP

The Open System Interconnection model (OSI Model) is a conceptual model consisting of different layers that help to standardize communication systems between different systems. It was published in 1984 when different networks and devices all used their own protocols to handle and set-up connections. The idea of the model is that each layer relies on the layer below for errorless communication while providing an error-free communication link for the layer above. The original model consists of seven different layers:

1. Physical layer: it ensures bits are transferred from one physical point to another. The copper cables, glass fiber, wireless signals are the protocols to transmit between them are all part of this layer.
2. Data link layer: it ensures that frames of data can be transmitted between two connected nodes.
3. Network layer: it ensures that packets of data can be transmitted between two connected nodes. These nodes have addresses so that they can be reached over a network. This layer also ensures that packets are automatically fragmented and send in parts if packets are too large.
4. Transport layer: it provides the functionality to transfer a block of variable length data from the source to the destination host. It can consist of different classes of features such as error correction, flow control, and segmenting data.
5. Session layer: it establishes, manages, and terminates connections between two hosts.
6. Presentation layer: it translates data between a networking service and an application; including character encoding, data compression, and encryption/decryption
7. Application layer: High-level APIs, including resource sharing, remote file access

Figure 2.1 - OSI Model



The Internet Protocol Suite (IP) is an implementation of the conceptual OSI model. It is currently the most widely used protocol to exchange information through the Internet. The Internet Protocol suite consists of many protocols operating in the different levels of the OSI model, although it is not strictly following the separation of all the model layers. The IP Suite is split into four different layers: the link layer which is similar to the data link layer in the OSI model, the internet layer which is similar to the

network layer of the OSI model, the transport layer which is similar to the transport layer of the OSI model and finally the application layer which is similar to a combination of the last three layers of the OSI model. The two protocols of interest of this suite are the Internet Protocol version 4 (IPv4), operating in the network layer and the User Datagram Protocol (UDP), operating in the transport layer.

Internet Protocol version 4 (IPv4)

Internet Protocol version 4 (IPv4)[3] is one of the core protocols of the Internet currently in use. IPv4 is a connectionless protocol that is used to route packets over the Internet. The idea is that each node has a different unique address that is routable through the Internet. Without these unique addresses, systems on the Internet are unable to find each other. Each address contains four numbers between 0 and 255, separated by a dot. For example, the address of user A could be 208.67.222.222 and 8.8.4.4 could the address of user B. Now user A can send a packet to the address of user B which is routed over the internet until it arrives at user B. However, the delivery of a packet is not guaranteed, a packet could be dropped along the way.

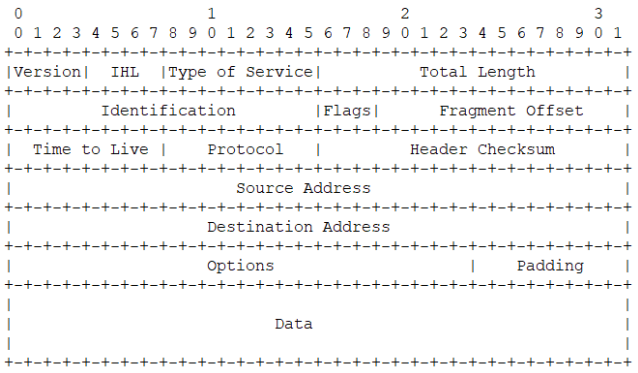


Figure 2.2 - IPv4 Header

IPv4 was first deployed in 1983, and at the time of the creation, an address size of 32 bits (four bytes) was deemed more than enough. This limits the total number of available addresses to 2^{32}, or roughly 4.3 billion addresses. Some addresses are reserved for special use, leaving approximately 4 billion addresses available for use. At the time, this was enough to provide almost all inhabitants of planet Earth with a unique address. However, in 2011 [46] the last blocks of IPv4 addresses were reserved, and there are now no new blocks available for use.

The IP header is shown in figure 2.2. The header consists of a fixed 20-byte length, with additional options trailing the header if specified. After the header and options, the payload of the packet is included. The header specification is as follows: the first four bits indicate the IP version. For IPv4, this field will always be four. The Identification field is primarily used for identifying individual fragments of datagrams. It can also be used to trace packets.

In the third row, there is the Time to Live (TTL) that specifies the maximum time the packet may be in the Internet system. At every hop, this value decreases by one. This is to prevent undeliverable packets from being in the system forever. Next to the TTL is the protocol specifier that specifies the protocol of the payload. The most commonly used protocols to transfer data between applications are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP).

The IPv4 header also included the source address of the packet and the destination address of the packet. These addresses are used to route packets from one host on the internet to another. The receiver can then also send back a response to the sender by using the source address.

User Datagram Protocol

Applications mainly use two different protocols to transmit data between applications over the Internet: UDP and TCP. Before giving a detailed description of UDP, a small description and comparison are given between UDP and TCP. TCP provides a reliable data stream in which packets will always arrive, arrive in order, and arrive without errors. It also provides additional features such as flow control to ensure no data is sent faster than the recipient can receive. To do this, TCP first negotiates a connection between two hosts and only starts transmitting data when a connection is established. Upon completion of the full session, the connection is closed.

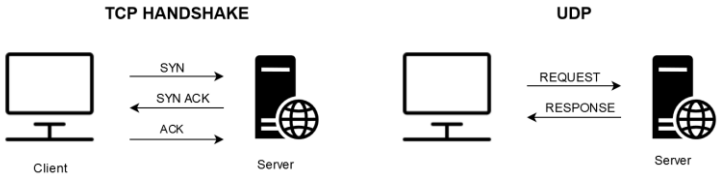


Figure 2.3 - TCP vs UDP communication

[3] https://tools.ietf.org/html/rfc791

The User Datagram Protocol (UDP) does not make use of connections and transmits the data directly to the target without setting up a connection first. Packets may get lost, or packets may arrive in a different order than in which they were sent. Because of the connectionless nature and the fact that there is no waiting period to transmit potentially dropped packets, UDP generally has a much lower overhead and latency. Nowadays, UDP is mostly used for applications that are time-sensitive and where packets lost in transmission are acceptable. Voice calling is an example of such a service, where you are willing to sacrifice not hearing every tiny single fragment of speech to obtain low latency communication.

As such, UDP is a very simple protocol, and its header reflects this. The header format can be seen in figure 2.4. The length of the header is exactly 8 bytes. The first two bytes define the source port of the packet. The next two bytes define the destination port of the packet. Byte 5 and 6 define the total packet length of the UDP packet, and finally, there is a checksum that is used to check for any transmission errors.

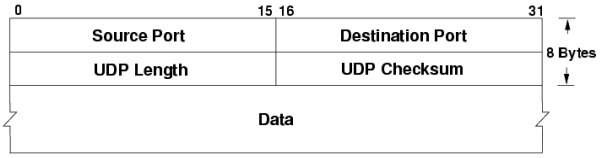


Figure 2.4 - UDP Header format

Internet Control Message Protocol

ICMP packets, or Internet Control Message Protocol packets, are encapsulated in IPv4 packets, similar to UDP. The protocol is described in RFC 792[4] in 1981 and was expanded upon in RFC 1122[5] and RFC 1812[6]. ICMP messages are used for diagnostic or control purposes. They can be generated in response to a transmitted IPv4 packet and are sent to the source address of the original packet.

The message contains a type as well as a code that provides information about the purpose of the packet. The messages also include the original transmitted packet and the destination towards which the packet was sent. This allows tracing back the control message to a particular attack that occurred. During the research in this thesis, there is one particular type of ICMP message that is worthwhile to investigate further: ICMP messages with 3 as type. Type 3 is defined as a response that informs the transmitter that the destination is unreachable. When such an ICMP is received during an attack, this may provide more insight into what is happening with the victims during a DDoS attack. These destination unreachable messages also include a code that further specifies why the destination was unreachable. The different codes and their meaning are discussed during analysis.

2.2. D(R)DoS attacks

The previous section explained that the Internet could be conceptualized in different layers. An implementation of this model is the Internet Protocol Suite. In our case, the focus is on two protocols which are included in the IP suite: the IPv4 protocol and the UDP protocol.

This section will explain step by step what a DRDoS attack is by first describing what a Denial of Service (DoS) attack is and then discussing what the difference with a DDoS attack is. To understand this difference, more background information is provided. Thereafter, the technical details behind a DRDoS attack are explained.

DoS Attack

In a denial-of-service attack (DoS), an adversary tries to render a target application, machine, or other resources unavailable to its intended users. The adversary can do this by employing several techniques. Common techniques include crashing a server with a specific command that the service is vulnerable to, or quickly opening and closing connections so that the server's resources get exhausted.

Another technique that is used, is an attack where an adversary floods a target with data to saturate the available bandwidth and thereby preventing legitimate users from still being able to access the service. The saturation attack can be compared to a physical situation. For example, imagine a village with a restaurant. The restaurant can only be reached by car, and there is only a single road going to and from the restaurant. On a typical day, this road is barely used and only visitors to the restaurant use the road.

[4] https://tools.ietf.org/html/rfc792
[5] https://tools.ietf.org/html/rfc1122
[6] https://tools.ietf.org/html/rfc1812

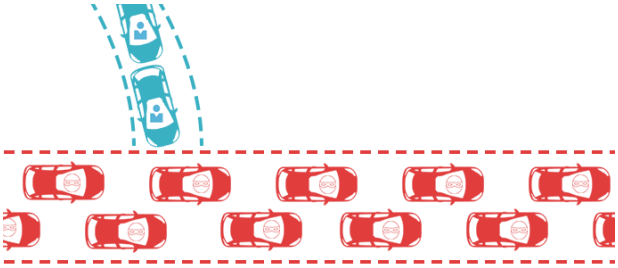


Figure 2.5 - DDoS on the road

The adversary has ill intentions and wants to prevent the restaurant from gaining revenue. The adversary buys a lot of self-driving cars and ensures that they all drive down the single road leading towards the restaurant. The road is now congested. The intended visitors of the restaurant can no longer enter or leave the restaurant and the adversary prevented the restaurant from serving customers. Figure 2.5 illustrates this process. The regular customer's cars can no longer reach the restaurant because the main road to the restaurant is full of adversarial cars.

On the Internet, each packet, or in the example of the restaurant, a car, has a destination address and a source address. A potential solution to prevent this type of attack would be to have an authority, such as the police, to remove all cars - or packets - with the source address of the adversary from the road. Therefore, this form of attack is seldom used. The next subsection presents a modified version of the attack that is harder to block.

DDoS Attack

A DoS attack that uses flooding could be stopped by blocking all cars - or Internet packets - with the source address of the adversary. However, that attack can be modified to circumvent this simple filter, leading to an attack that is harder to block.

In a DoS attack, all cars were labeled with the source address of the adversary. However, the adversary could have cars originate from different locations. In that case, all cars would have a different source address on them. Because all the source addresses differ, it is no longer possible to filter out the malicious cars that were coming from a single source. Because of the distributed nature of this attack, this form of attack is called a Distributed Denial of Service attack - or a DDoS attack.

Amplification

Before explaining how a DRDoS attack differs from a DDoS attack, a vulnerability of using UDP services is demonstrated that enables the possibility of launching DRDoS attacks. To illustrate the vulnerability, a toy protocol running on UDP is used. Since UDP is a connectionless protocol, an adversary can send a packet towards a service running on UDP, where once the adversary sends a valid command, the server will send a response back to the source address. There is no further interaction required.

For the toy protocol, a theoretical weather service is created. The weather service works as follows: the client sends a request to obtain the current weather status, and it does so by sending a packet containing the text "Weather of the next few days?". The weather service receives the packet and processes the command as it detects it as a valid request: it is able to handle the request and can generate a response that the user expects. The weather service replies with a packet containing a weather report. Figure 2.6 shows an overview of this interaction.

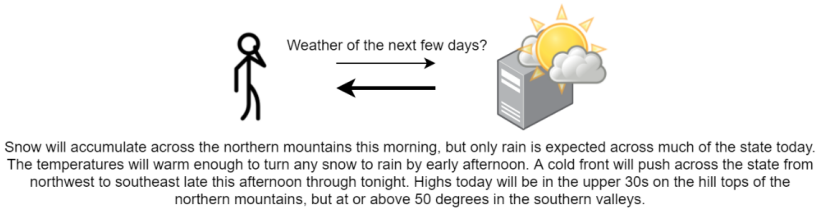


Figure 2.6 - A person requesting the weather forecast

This service is seemingly innocent, but the response the server sends back contains a lot more characters than the request the client sent. There were 29 characters sent, and 439 characters received. This means that the server transmitted 15 times more data as the client did. This factor is called the amplification factor. This factor may vary with each response, as different weather messages may consist of a varying number of characters.

Spoofing and the DRDoS attack

The IPv4 header contains a source and a destination address[7]. The receiver uses the source address in order to send a response back to the source. Now imagine a service that sends a response that contains exactly what is sent to the server. Such as service can be useful for debugging purposes, for example, because the sender would know that the server he sent the response to is still active.

But the server cannot verify whether a valid source address was used. To understand why this is not possible, consider the case where someone receives a letter by mail. You would not know from whom the letter is, all you know is that the postman delivered it. The sender can include his address on the letter so the receiver would know to whom to return a response. However, if the sender would include another address that is not his, you have no way to assume that that address is not from the actual sender. This is a non-trivial problem.

The same applies to the digitally transmitted packets. The adversary could use an address that is not his in the IPv4 header. In this case, the server would send a reply to the address the adversary provided. The adversary cannot see the response to any of his requests in this case. The only thing that occurs is that the replies to his packets arrive at the provided address. This leaves the question: why would an adversary include an address that is not his?

Imagine the weather service again. When the adversary sends the 29 bytes long request containing “Weather of the next few days?” but puts another address as the source, that address will receive 439 bytes of data. Sending packets with a falsified source address is called IP address spoofing. Figure 2.7 illustrates what happens in such a situation. In figure 2.6, an adversary sends a small request and uses the source address of the bank in the packet. The weather service sends its much longer response to the server of a bank, instead of back to the adversary.

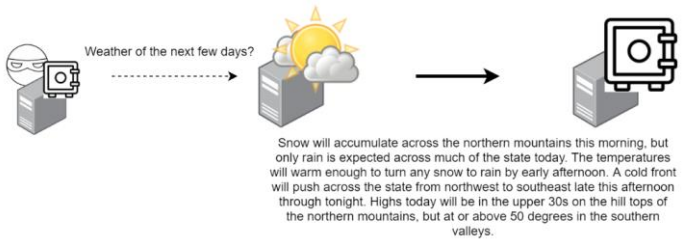


Figure 2.7 - Spoofing the source address of a packet

This results in different benefits for the adversary: the victim cannot see the source address of the adversary, but instead sees a source address of the vulnerable server. This makes attributing an attack much more difficult, as the apparent source of the attack is also a victim. Secondly, the adversary will need fewer resources to create a much more massive attack. Consider the weather service example: to generate an attack of 1 Gbit/s, the adversary only needs to generate 15 times fewer data - 67 Mbit/s - to create an attack of equal size. Finally, an adversary can easily create a distributed attack, while only making use of a single source. It will only need a single server that allows for sending out packets with falsified source addresses.

Making use of spoofing and vulnerable amplification services is also called ‘Reflection’. If this is combined with the distributed nature of the attacks, together, this forms a Distributed Reflected Denial of Service Attack, or a DRDoS attack. The attack is often categorized as a volumetric attack, as the purpose of the attack is to overflow a victim. It is sometimes also named an Amplification Based Distributed Denial of Service Attack (ADDoS). Figure 2.8 shows an example of a DRDoS, where an adversary makes use of several vulnerable weather services to reflect and amplify traffic to the server of a victim, in this example a server of a bank.

[7] technical details are included in section 2.1

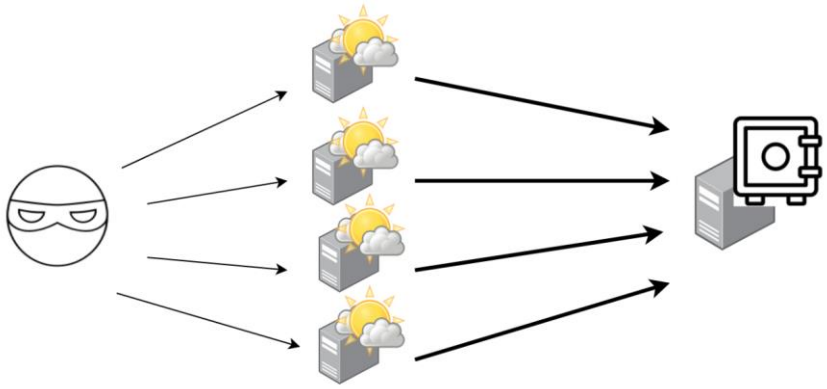


Figure 2.8 - DRDoS attack using vulnerable services to attack a server of a bank.

Scanning

Before an adversary can execute a DDoS attack, he needs to find vulnerable servers that it can use for these attacks. These servers are scattered around the Internet, but an adversary does not know where all these services are. A method commonly used by adversaries to find these services, is scanning. With scanning, an adversary sends packets towards a large number of Internet addresses that are targeted specifically on selected ports. These selected ports are chosen by an adversary that is looking for specific services to use. Since these services have commonly used ports, this is an effective method to find many of these services.

There are several different outcomes of a scan:

1. No response is returned, and the address is not usable.
2. A response does not match something that the adversary was expected to find, for example, because the service is not vulnerable.
3. The content is as expected, for example, because the response indicates that the service is vulnerable.

With the third outcome, the discovered Internet address and port combination can be stored. The stored addresses can then later be used to start reflection attacks. Figure 2.9 shows a schematic overview of the different scanning steps. In step 1, an adversary wants to find services on the Internet, in step 2, the adversary performs a scan. Step 3 shows the three different outcomes of a scan: (1, in red) no responses, (2, in black) servers that were not vulnerable, and (3, in green) vulnerable weather services. Scanning all IPv4 Internet addresses is feasible, it can even be done in a few minutes [42].

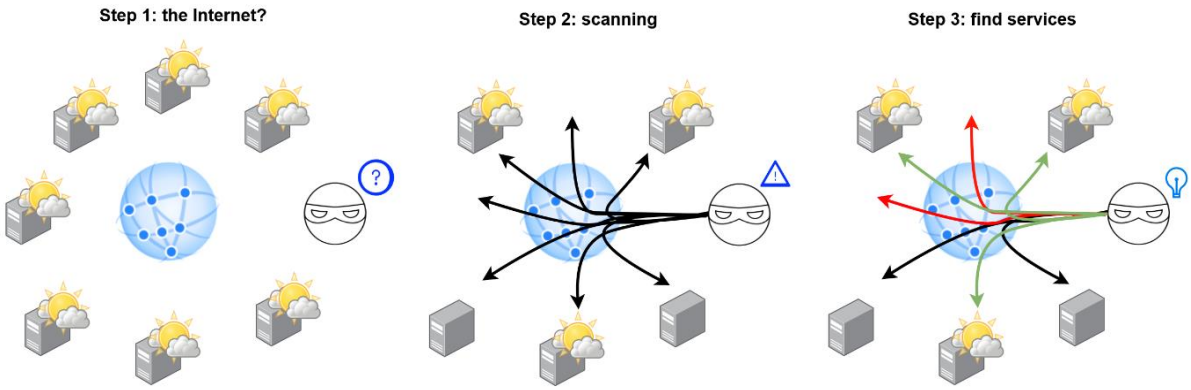


Figure 2.9 - overview of service discovery by adversaries

DRDoS Attack Summary

In a Distributed Reflected Denial of Service Attack, adversaries sent packets with spoofed addresses to vulnerable services that amplify the generated traffic. The vulnerable services then sent their amplified response to the spoofed address: the victim. The victim is then flooded with traffic appearing to originate from the many vulnerable services that are used. An adversary scans the Internet in order to find these vulnerable services.

3

Related work

To understand how an attacker executes a DDoS attack, first, a model is introduced that describes the steps an attacker needs to take in order to execute attacks. The section thereafter discusses the related options that are available to gather intelligence about adversaries and the landscape in which they operate. In the section thereafter, vulnerable services that are commonly used in DRDoS attacks are discussed.

3.1. The modeling of DDoS attacks

Executing a DDoS attack requires multiple stages [47]. While an adversary needs to succeed in all attack stages, a defender only needs to block the adversary on its path to the final stage. Using a model to analyze adversaries helps to standardize the different stages an adversary has to take, which in turn helps to focus collective defense efforts on a single stage.

A frequently used model while studying the behavior of adversaries, is the cyber kill chain model. The cyber kill chain model was introduced by Hutchins et al. [48] and published by Lockheed Martin. The kill chain model provides a universal framework with different stages. However, several stages of the kill chain model are missing with DDoS attacks, and some stages do not properly fit the description. This means that the kill chain model cannot be directly applied to model DDoS attacks. Harris, Konikoff & Petersen [47] published work which includes a mapping of DDoS attacks into the kill chain model in 2013. This mapping and its definitions aimed at DDoS attacks aid the discussion to analyze DDoS attacks and to construct counter-measures in each of the stages. The stages are defined as the following:

- Reconnaissance: in this first phase, an attacker is trying to find vulnerable targets. In this case, that would be finding various UDP hosts and services which allow being used to reflect data and provide a sufficient amplification factor. Adversaries often employ a technique called scanning to find such services. Countermeasures in this stage include Intrusion Detection Systems (IDS), which will detect ongoing scans and block those scans. This prevents adversaries from further gathering intelligence.

- Weaponization: in the weaponization phase, an attacker designs an attack with the resources it possesses. There are various types of DoS attacks that can be employed, but this thesis focuses only on DRDoS attacks. Countermeasures in this stage include modifying existing systems or upgrading vulnerable protocols. Nowadays, new protocols are designed such that known vulnerabilities are avoided.

- Command & Control: To execute an attack, an adversary uses communication channels towards a botnet or otherwise controlled systems to execute the attack. Control messages may include targets, durations, bandwidth configurations, and other settings. Countermeasures on this stage focus on detecting the control flow used by adversaries and disrupting or blocking them.

- Actions on objective: During this stage, a D(R)DoS attack is ongoing. Countermeasures may vary. For example, services may be distributed among several networks in a way that an attacker will need to attack multiple networks to create an outage. Other strategies may be used to detect D(R)DoS flows and dropping these malicious packets in a place where bandwidth is still sufficient so that the line or server further down the line will not get congested.

3.2. Gathering Intelligence

The previous subsection introduced a model to study DDoS attacks. As a defender, it is only of importance that the adversary is stopped in one of the stages. During this thesis, the primary area of focus will be on the final stage: actions on objective. However, the stages are closely related, and the reconnaissance stage is a precursor for the final stage. Thus, related work from both stages is discussed to aid the direction of our research and to identify the research gap.

Both stages have active and passive means of analysis, and so both passive and active analysis methods are discussed for the reconnaissance stage and the actions on objective stage. The first two subsections discuss active and passive scanning analysis. Thereafter, the passive and active attack analysis are discussed.

Passive Scanning Analysis

The first stage of executing DDoS attacks consists of finding vulnerable servers on the Internet that can be used to reflect and amplify data. To find these vulnerable servers, adversaries scan all the addresses that are available on the Internet. Monitoring the scanning activity can help to gather data and to detect patterns and trends. Different adversaries may employ different scanning techniques. One adversary might find importance in not being detected whereas another adversary perhaps wants to scan the whole Internet as frequently as possible to gather the majority of open servers and would not mind that some networks would block its scans. Griffioen [49] discovered several complex strategies by analyzing scanning data retrieved from the TU Delft network telescope. Some of these discovered strategies are: trying to randomize scanning IPs, only execute scans at an extremely slow rate, doing only partial network scans, randomizing packet fields, and scanning at random time intervals.

Work by Kührer et al. [50] shows that roughly half of the vulnerable amplifiers vanish after a week. This means that adversaries will have to rescan vulnerable services in order to ensure that they are still vulnerable and available. Newly deployed vulnerable services, including those deployed for research, will thus likely be found within several days to a week.

Analysis of results that Bohte [43] published shows that the two honeypots that he used in his work detected different scans. His two honeypots were deployed in different geographical regions. One honeypot detected more QOTD scans than the other honeypot. Furthermore, his work from 2018 shows that there are still active scans that are scanning for services running QOTD, CHARGEN, DNS, NTP, RIP, MSSQL, and SSDP.

Active Scanning Analysis

The method of scanning is not only used by adversaries. Researchers employ similar methods to make estimations of the vulnerable landscape. In a way, these research projects take the view of an adversary to remove the information asymmetry between adversaries and researchers. These results are then used to estimate the total available attack factor [43] or to run campaigns to reduce the vulnerable landscape [50].

There are also research projects that publish scanning results openly for anyone to access. One such a research project is Shadowserver[8]. When Shadowserver finds a vulnerable server, it informs the registered owner of the IP network that a vulnerable service was found on their network. Shadowserver also provides statistics on publicly accessible vulnerable servers. These statistics are discussed for the relevant services in section 3.3.

Research projects like Shadowserver usually use reverse DNS records on the IP addresses where they are performing the scans from. The description of that reverse DNS record provides an indication that the scans are for non-malicious practices. In this thesis, we will also make use of those reverse records to distinguish scans used for research and by adversaries.

Passive Attack Analysis

Information about attacks can be gathered by analysis of internet traffic. There are projects that do this type of analysis and that monitor D(R)DoS attacks in real-time. One such monitor is the Digital Attack Map[9], which is a map of the world with current attacks taking place, including current attack flows. The DDoSMon Insight [9] provides historical information about attack counts, sizes, durations, and attack compositions. As such, passive attack analysis can provide a global view of the attack landscape. Such an overview also helps to provide direction for research. The DDoS Monitor insight shows that 75% [9] of DDoS attacks are of the reflection type that this thesis focuses on.

[8] https://www.shadowserver.org/who-we-are/
[9] http://digitalattackmap.com/

Active Attack Analysis

Honeypots can be used to do active attack analysis to gather information. Honeypots are designed to lure adversaries into using a system or service while recording the behavior of the adversary. By behaving like the system or service that attackers like to use, adversaries are unaware that the system is a honeypot. With the recorded data, behavioral patterns can be established. This provides more information about the techniques and tools an adversary uses. It also helps to understand the goals of such an adversary. A honeypot could, for example, monitor and store which passwords an adversary uses when trying to login to a specific system.

The decade's old concept was first used and described in Spitzner's [51] work in 2002 and by Provos in 2003 [52]. Since then, honeypots have been deployed for various research purposes before, for example, for collecting and analyzing malware [53] or for collecting the behavior of adversaries [54].

Usually, when honeypots are designed, the goal is to build a honeypot that acts similar to a real system. If the created honeypot is similar enough to a real system, the adversary will not notice that the system is a honeypot. Figure 3.1 illustrates such a system. Ideally, the adversary will believe that he is communicating with a vulnerable service, such as the weather service example. In reality, the adversary is communicating with a honeypot.

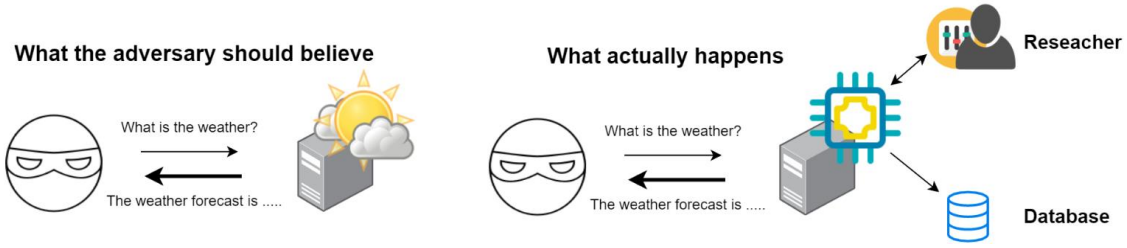


Figure 3.1 - The behavior of a honeypot

Although the concept of honeypots is not new, the first use of honeypots while doing research on DDoS attacks is more recent. Krämert et al. [44] created the AmpPot honeypot framework in 2015. With this framework, they managed 21 different honeypots running in various modes of operation. The modes of operation they used are the following:

- 'Emulated': In the emulated mode, the honeypot has a parser for the protocol built-in. It will parse the incoming request and generate a response according to the specification of the protocol.
- 'Proxy': In the proxy mode, the honeypot will forward the request to an internal server running the vulnerable protocol. The response of that internal server is then sent back to the client.
- 'Agnostic': In the agnostic mode, the server will completely neglect the validity of the incoming request and will not parse any of the data. It will only send back a large block of data. This mode assumes that the attacker does not care about the (validity) of the responses, but only about the amount of data it will give back.

This thesis will also make use of these different modes of operation, which are further discussed in the upcoming two chapters. Aside from the modes of operation, the research of Krämert et al. [44] also leaves some additional insights:

- 90% of scanners only scan one protocol. Only a few scanners scanned multiple ports, mostly in combination with DNS. This could mean that the number of running services per honeypot will not (noticeably) affect results.
- All deployed honeypots were abused within 24 hours after deployment. As more days passed, the number of attacks started to increase further. After 5 days, the number of attacks stabilized.
- 96% of the observed attacks appeared to originate from a single source, indicating that the source is a booter service[10] and was not a large DDoS botnet.

The AmpPot project also introduced a concept to measure honeypot effectiveness. In summary, the idea is to compare the number of unique attacks that consecutive honeypots detect. For example: assume honeypot A. Then calculate how many attacks honeypot B detects that were not seen by honeypot A. Then repeat for honeypot C, detecting which attacks were not seen by honeypot A and B. This concept is used and further explained during analysis.

Kamoen [55] did a study on adversarial behavior using honeypots and showed that the metadata a server sends has a significant influence on the behavior which adversaries would employ next. In his research, two groups were used in A/B testing to test whether a banner indicating the server was a honeypot had any affect. The results showed that only 4% of adversaries tried to login after it was clear the server was a honeypot, compared to 86% with other banners. The absolute amount of logins differed

[10] see glossary for definition

by more than 140 million. This indicates that adversaries are careful with the services that they abuse. This research was, however, focused on a service running on the TCP protocol, where address spoofing is not possible. For UDP, an actual DRDoS attack only contains spoofed addresses.

Honeypots also provide means to attribute attacks to adversaries. The source address of the incoming traffic is actually the address of the victim. Thus it is not straightforward to distinguish the origin between an adversary and the actual targets. Several attempts have been made into attack attribution with honeypots. Krupp, Backes, and Rossow [56] used a framework that only responded to a scan with a selective set of honeypots. This unique set of used honeypots acted as a fingerprint. When exactly that set of honeypots was used in an attack - the attack could be linked to an adversary. Work by Krupp et al. [57] used a subscription to several booter services and executed DDoS attacks on their own targets. They then used machine learning to attribute attacks to booter services.

3.3. Vulnerable Services

Any service that amplifies data and uses UDP can potentially be used in a DRDoS attack. In 2014, Rossow [45] published a comprehensive list of vulnerable UDP protocols. In 2018, Bohte [43] published a follow up to that work and thereby updated the current state of intelligence for amplification-based DDoS attacks. The different protocols are often compared by using their potential amplification factor. This section first explains how the amplification factor is defined and then discusses vulnerable protocols that are selected for the research of this thesis.

The Amplification Factor

Rossow [45] published work where the amplification factor was defined. Subsequently, this definition was then also used by Krämert et al. [44], Kührer et al. [50], and Bohte [43]. This definition of the amplification factor was the following:

Bandwidth Amplification Factor = len (UDP payload) amplifier to victim / len (UDP payload) attacker to amplifier

In other words, it is the ratio of the transmitted data divided by the received data. However, as section 2.1 explained, more than just the data is transmitted. The Internet uses protocols and headers to ensure that data is delivered from source to destination. A reason for this exclusion is given by Rossow [45]: "*We chose not to include Ethernet, IP, or UDP headers in this calculation so that our measurements remain valid even if the upper protocol layers change in the future, such as during the migration from IPv4 to IPv6." [45].* Although this is a valid reason, it does not reflect well on reality. This definition can have a major influence on results, even skew them so that it appears that simple protocols offer an extremely high amplification factor, while in reality, the factor is much lower. This thesis will use a modified version of this definition, which is further discussed in section 4.2.1 of the methodology chapter.

Vulnerable Services

The protocols that are used for this thesis are those that were most recently used in DDoS research, by Bohte [43] in 2018. These protocols were selected because there is recent information available and they are still being actively used and scanned for. Bohte [43] discussed seven different protocols, one of which is MSSQL. However, since any emulation for this protocol is much more complex than the other six services, and it was the service with a relatively low amplification factor compared to the other services, this service was not selected.

Thus, the selected six services are the following: QOTD, CHARGEN, DNS, NTP, RIPv1, and SSDP. Table 3.1 shows an overview of the key characteristics of these six services, their use cases, and what the typical amplification factor is that is reported by Bohte [43]. More in-depth information about the six different services is provided in the glossary.

Table 3.1 - Service overview

Service	UDP Port	Complexity	Status	Use Case	Typical Amplification Factor
RIP	520	Simple	Deprecated	Routing Protocol	131
CHARGEN	19	Simple	Legacy	Diagnostics / Entertainment	358
QOTD	17	Simple	Legacy	Diagnostics / Entertainment	140
SSDP	1900	Medium	Active / in use	Service Discovery in Networks	30
NTP	123	High	Active / in use	Time Synchronization	557
DNS	53	High	Active / in use	"Phonebook" of the Internet	28

Table 3.2 shows the number of open servers of each of the different services running the six different protocols in the last five years. These numbers are based on reports by Shadowserver. For RIP, Shadowserver had no such information available.

The legacy protocols QOTD, CHARGEN, and the deprecated mode 7 functionality of NTP have many fewer open vulnerable servers compared to the still actively used protocols DNS and SSDP. However, although there are significantly fewer open vulnerable servers, large attacks can still be generated with them. If each of the servers is only targeted with one megabit, this would total to an attack size of 4.18 Tbit/s for QOTD, 12.10 Gbit/s for CHARGEN, and 23.02 Tbit/s for NTP based on numbers on January 2019. Thus, even though there have been advancements made in bringing down the number of available vulnerable servers, the servers that remain still pose a threat.

Bohte [43] reported similar insights on the number of open servers for QOTD, CHARGEN, and SSDP. However, the number of open DNS servers found by Bohte was twice as much as reported by Shadowserver in 2018. The amount of NTP servers was much higher than reported here - even when combining NTP Mode 7 and NTP Mode 6 servers, Bohte reported ~4 million more servers.

Table 3.2 - Number of open servers according to Shadowserver

	2015	2016	2017	2018	2019
QOTD[11]	~29.700	~21.600	~23.000	~29.900	~34.400
CHARGEN[12]	~60.500	~26.700	~35.700	~27.200	~33.800
DNS[13]	~7.109.000~	~5.045.609	~3.836.300	~3.813.500	~3.144.900
NTP Mode 7[14]	~87.400	~57.900	~34.500	~38.100	~41.400
SSDP[15]	~15.074.800	~11.065.900	~6.733.900	~5.310.675	~3.319.000

[11] https://scan.shadowserver.org/qotd/stats/
[12] https://chargenscan.shadowserver.org/stats/
[13] https://dnsscan.shadowserver.org/stats/
[14] https://scan.shadowserver.org/ntpmonitor/stats/
[15] https://ssdpscan.shadowserver.org/stats/

4

Hypotheses & Methodology

This chapter first develops the hypotheses that this thesis tries to validate. The section thereafter maps these developed hypotheses that are used in A/B testing[16]. The third section discusses the different test settings that are used for the constructed variables. The fourth section discusses how the influence of the different settings is compared. The fifth section discusses the test phase where more information was gathered to aid the construction of the settings. The two sections thereafter provide more detailed information about the settings that are used for the different experiments and the test groups that are used.

4.1. Hypotheses - Anticipated Adversary Behavioral Traits

In previous work, honeypots were used to gather and analyze adversarial behavior related to DDoS attacks. With these and other techniques, it is known which services are vulnerable and what their typical amplification factor is. However, these vulnerable services and the honeypots that emulate them are generally attractive vulnerable targets that can be exploited in perfect conditions.

Estimates on the total amount of vulnerable servers are also either the amount of services that responded to a query or are interpolated based on a partial scan of the entire IPv4 address space. However, it relies on an easily made assumption: it assumes that the available, usable resources of adversaries are equal to the amount of available vulnerable servers. This assumption may be incorrect: many of the services might not be vulnerable at all or have other influencing configuration settings so that they can actually not be used as a good reflector in a DRDoS attack.

Adversaries, like any other company or project, will also have a limited amount of resources available. An adversary may only have a certain amount of compromised servers available, with a certain maximum amount of bandwidth. As such, an adversary may also be trying to use the resources it possesses in the most efficient manner possible.

This raises several questions: are adversaries sufficiently sophisticated to take differences between configured systems into account? And if they do, which characteristics do they use to make distinctions between one system and another? This thesis tries to answer those questions with the following research question:

RQ: Which factors of a vulnerable service influence the abuse by adversaries for Distributed Reflected Denial of Service Attacks?

One of the primary factors in determining the effectiveness of a DRDoS attack is the total bandwidth of the attack. This means that a vulnerable service that amplifies more data will have a higher impact than a service which amplifies fewer data. Therefore, the primary hypothesis of this research is:

Hypothesis 1: "There will be a difference in abuse frequency between groups that provide a high amplification response and groups that provide a lower amplification response."

[16] With A/B testing, a variable with different settings is used, so that the effect of that variable can be measured.

However, several other properties could influence the frequency of attacks that the services will experience. A high-quality connection may see more activity than a low-quality network. There are several properties that describe network quality. These include the connection speed, the amount of packet loss, and the response time. It is expected that these factors, similar to the amplification factor, will have an influence on the abuse frequency. This leads to the following three additional hypotheses:

Hypothesis 2a: "There will be a difference in abuse frequency between groups which have a fast network connection and a network which has a slow network connection."

Hypothesis 2b: "There will be a difference in abuse frequency between groups which have almost zero packet loss and a network which has a high amount of packet loss."

Hypothesis 2c: "There will be a difference in abuse frequency between groups which have a fast response time and a network which has a slow response time."

In the experiments Krämert et al. [44] conducted, many services were hosted at the same IP address. As the researchers themselves noted, this might influence the results as attackers may exclude these honeypots from their attacks. However, it also showed that 90% of scanners only scan a single protocol. This thesis tries to test this hypothesis to see if a significant difference could be detected.

Hypothesis 3: "There is no difference in abuse frequency between groups in a network where each IP contains multiple vulnerable services compared to a network which only runs one vulnerable service per IP".

Research by Kamoen showed that honeypots that specifically indicated that they are a honeypot were less often abused. However, this research focused on services running on the TCP protocol. This meant that a connection had to be established first and that the source IP could not be spoofed. With DRDoS attacks, the traffic is spoofed. Thus, with DRDoS attacks, one could argue that adversaries are indifferent to the content of the response. A real service, an emulated service, or even a response, which indicates that the used system is a honeypot: they may all be abused equally. This leads to the following hypotheses:

Hypothesis 4a: "There is no difference in abuse frequency between real responses and responses that indicate that the system is a honeypot."

Hypothesis 4b: "There is no difference in abuse frequency between emulated services and actual implementations running in a virtualized container."

Bohte [43] had only two honeypots but there were placed in different geographical locations and at another provider. These two honeypots showed different results. Krämert et al. [44] also placed their honeypots in different geographical locations but it was unclear if this had any influence on the results. Thus, several questions arise: are these differences caused by selecting different providers, or are these differences caused by hosting honeypots at various geographical locations? Or perhaps even both? This leads to the following two hypotheses:

Hypothesis 5a: "There is a difference between attacks that are detected among honeypots that are located at different geographical regions."

Hypothesis 5b: "There is a difference between attacks that are detected among honeypots that are located at different providers."

4.2. Mapping Hypotheses to Variables

This section discusses the mapping of the hypotheses to variables. Table 4.1 shows an overview of the different hypotheses, the purpose of the variable in testing that hypothesis, and finally, the variable that is used to test the hypothesis. For each of the variables, the goal is to test how the different settings of that variable influence how often the honeypot that runs that setting is used in an attack.

For hypothesis *H1*, the goal is to test what the impact is of the amplification factor. In this case, if the hypothesis is correct, it is expected that honeypots that run with a setting that provides a higher amplification factor, are used in attacks more often than the honeypots that return a response that provides a lower amplification factor.

For hypotheses *H2.a, H2.b, and H2.c*, the goal is to test the influence of perceived network quality properties. These include the connection speed, packet loss, and the response time of the system. These influences are tested by using rate-limiting, artificial

packet loss, and artificial packet delay respectively. For each of the hypotheses, a variable is introduced with different settings. The details of these three factors are included in the next section.

For hypothesis *H3*, the goal is to test the impact of running multiple honeypots on a single public IP address compared to running only one honeypot on a single IP address. Due to the nature of this experiment, this test is included in a group with experiments that are called 'deployment differences', as it refers to *how* a honeypot is deployed.

Four of the six services that are used in this research are protocols that were of simple or medium complexity and can be implemented in the test framework directly. These four services are CHARGEN, QOTD, SSDP, and RIPv1. They are used to test hypothesis *H4.a*. For these services, the variable influences which content the services return upon a request.

Hypothesis *H4.b* is used for DNS and NTP. These services are more complex than the other four services and have implementations that are commonly used on the Internet. For these two services, the variable impacts the mode of operation: one mode uses an external service running on the system to generate the traffic, whereas the other mode may use a (partial) emulator. Hypotheses *H4.a* and *H4.b* are both included in an experiment group that is called 'fake services'.

Hypotheses *H5.a* and *H5.b* are also both part of the 'deployment differences' experiment group - they determine *where* a honeypot is deployed. The goal of hypothesis *H5.a* is to test what the impact is of running a honeypot at a specific geographical location. e.g. is there a difference if a honeypot is deployed in Europe or Asia? The goal of hypothesis *H5.b* is to test what the impact is of running a honeypot at a specific provider: is there a difference if a honeypot is deployed at provider X or Y?

Table 4.1 - Overview of hypotheses and their corresponding variables.

Hypothesis	A/B - Testing The Influence Of The	Variable Used
H1	The Amplification Factor	Amplification Factor
H2.a	Network Connection Speed	Rate-limit
H2.b	Packet Loss	Packet Loss
H2.c	Response Time	Packet Delay
H3	Multiple services per IP	Number of services running on 1 IP
H4.a	Response Type (Real or Honeypot)	Difference in Response Content
H4.b	Response Type (Emulated or Real service)	Difference in Operation
H5.a	Geographical Location used for the honeypot	Geographical Location
H5.b	Provider used for the honeypot	VPS Provider

4.3. Test Settings for the Variables and Experiment Construction

The variables that are defined to confirm or reject hypotheses each require to define different settings that can be compared against each other. The first five subsections discuss the different settings that are selected for the variables to test the hypotheses. Thereafter, the experiment design is discussed that follows from the constructed settings.

The settings for some of the variables are based on a preliminary test phase that was used to aid the construction of the settings. For these settings, there was no known information that could be used to relate the settings to. The details for the test phase are included in section 4.5.

4.3.1. Variables for H1: The Amplification Factor

Before the settings for the amplification factor can be discussed, the amplification factor must be defined. In section 3.3, the amplification factor was defined as the ratio of the length of the data that was sent to the victim compared to the length of the response. This definition did not include any headers and can have a major influence on results. With the definition, some protocols can appear to have a high amplification factor, while in reality, the factor is much lower.

For example, consider an example case with the QOTD protocol. If the size of only the UDP payload is compared to the size of the UDP payload including the size of the headers, there is a significant difference. These headers include the layer 2 (MAC Header), layer 3 (IP Header), and layer 4 (UDP Header). The total header is 42 bytes in total. Table 4.2 shows the differences between the

definitions. If only the payload size is considered, the amplification factor for QOTD would be 1400, whereas it would be roughly 34 if the headers are also taken into account.

Table 4.2 - Comparison of amplification factor calculation

	Request Size (bytes)	Response Size (bytes)	Amplification Factor
UDP Payload size	1	1400	1400
UDP Payload + headers	43	1442	33.53

Although the amplification may vary when the underlying protocols are changed, such as IPv6, many UDP servers run with IPv4 connected through Ethernet. Moreover, both IPv6 or a wireless link layer would only increase the header size - resulting in even lower amplification factors.

While performing an attack, the headers are always included in the used bandwidth. Thus, this work will take the header size into account while analyzing the results. The definition without headers is further abbreviated as BAF (Bandwidth Amplification Factor). The definition that takes into account all transmitted bytes is abbreviated as NBAF (Network Bandwidth Amplification Factor).

Amplification Factor Setting Groups

Hypothesis *H1* is the primary hypothesis for this thesis, and for that reason, two sets of settings were selected for the amplification factor variable. The primary set of settings for the amplification factor consists of two settings: one setting that ensures the honeypot returns a low amplification response and one setting that ensures the honeypot returns a high amplification response. The exact settings for this variable differ between the different services.

An additional set of settings that uses five data points instead of two was constructed to gather more data to validate or reject hypothesis *H1.* The set with five data points is used because this not only allows us to distinguish between a low and high amplification preference but also allows to more narrowly find a boundary of the amplification factor where different behavior is detected. This set of settings is referred to as the squeeze settings because the goal is to find this boundary.

To determine the size of the responses that should be used for both sets of settings, data from the test phase was used. During the test phase, arbitrary values were selected. This information can be found in section 4.5. These preliminary findings of the test phase suggested that after a certain amplification factor was reached, honeypots that used an even higher amplification factor were not used more. On the other hand, a very low amplification factor ensured that almost no attacks were attracted.

For the primary test set there are only two settings: a response that provides a low amplification factor and a response that provides a high amplification factor. Because an NBAF below 2 did not attract any attacks in the test phase, the minimum chosen value had at least an NBAF of 2. For the second setting, a larger value was chosen. The exact settings vary between the services and are discussed in section 4.6.

For the squeeze set, the settings are also based on the preliminary test phase. The preliminary data shows that there was less than a 0.5% difference in the number of attacks for QOTD between the settings that used 350 bytes and 1600 bytes as a response size. Therefore, the 350-byte value was selected and was rounded up to 400 bytes as a setting. A 600-byte setting is used in order to be able to measure if, for some protocols, an even higher amplification elicits more attacks.

The preliminary data also showed that for CHARGEN, there were fewer attacks that used the 94-byte honeypots than the CHARGEN honeypots that provided a larger response size. The test also showed that no 42-byte QOTD honeypots were used. Based on these findings, the lower two groups were selected: one setting that returns 90 bytes and one setting that returns only 1 byte. Finally, a value between 90 and 400 bytes was used for the final group: 200 bytes. The exact settings vary between the services for the squeeze settings. These different settings are discussed in section 4.7.

4.3.2. Variables for H2: The Network Quality Factors

The variables that are used to test hypotheses *H2.a*, *H2.b,* and *H2.c* are all related to the perceived quality of a network connection. The amount of available resources for this experiment is limited. This puts a limit on the number of settings that can be used for each sub experiment to ensure that there are enough data points to do meaningful analysis. For this reason, the number of different settings used for these variables is 3.

Hypothesis *H2.a* tests the influence of rate-limiting on the abuse frequency by adversaries. With rate-limiting, the maximum amount of bytes or packets that receive a reply is limited. The settings for this sub experiment were determined by analyzing the data during the test phase of the system. The goal of the rate-limiting experiment is to enforce a rate limit that is lower than what almost all of the attacks use - so it is evident that rate-limiting is applied for an adversary that would verify. Both selected values are below the 99th percentile of values that were found in the flows of the test phase.

- Setting 1: no additional limits (control group).
- Setting 2: 'light' rate-limiting: a maximum of 5 packets per second and 250 Bytes/sec.
- Setting 3: 'strong' rate-limiting: a maximum of 2 packets per second and 125 Bytes/sec.

Hypothesis *H2.b* tests the influence of packet loss on the abuse frequency by adversaries. In a network that is running optimal, there is no packet loss - so the setting of the packet loss variable would be 0%. In a network that loses all packets - no packet is ever returned to a response - the value of the packet loss variable would be 100%. Using the maximum value of 100% packet loss will not yield any results, as adversaries will be unaware that the honeypot exists. Thus, a control group and two settings that are spread between the minimum and maximum value are used. The three different settings that are used for this sub experiment are the following:

- Setting 1 has a packet loss of 0% (control group).
- Setting 2 has a packet loss of 33% on all packets after the first received packet.
- Setting 3 has a packet loss of 66% on all packets after the first received packet.

Even if a honeypot is configured to use packet loss, the first packet of a flow is always served with a response. Otherwise, adversaries would not be aware that there is a vulnerable service running.

Hypothesis *H2.c* tests the influence of packet delay on the abuse frequency by adversaries. Packet delays occur naturally on the Internet, as it takes time to transmit a packet from its source to destination. Furthermore, upon receiving a packet, a system should still process a packet before it can be further processed by an application that runs on that system. A slow response time may indicate to an adversary that the system is far away from its victim, or far away from the system that generates the DDoS traffic. It could also be an indicator for an adversary that the quality of the network is less than the quality of other networks. An adversary may thus prefer a honeypot with a fast response time.

A packet traveling from one side of the world to the other side of the world and back typically takes around 250 milliseconds. This was used as upper bound for the first delay setting, and double that value was selected for the second value. This leads to the following three different settings that are used for this sub experiment:

- Setting 1 has no artificial delay added (control group).
- Setting 2 has 250ms of artificial delay added on all packets.
- Setting 3 has 500ms of artificial delay added on all packets.

4.3.3. Variables for H3: Number of services running on 1 IP

Hypothesis *H3* states that there is no difference in the detected number of attacks between hosts that run honeypots for multiple services and hosts that only run a honeypot for a single service. To run only a single honeypot on a host is considerably more resource-intensive than if multiple honeypots can run on the same host. For this reason, the experiment primarily makes use of systems that run honeypot for each of the six services. To test *H3*, separate machines are selected that only run a single honeypot on them. The number of attacks they attract is then compared during analysis for each service.

4.3.4. Variables for H4: The Response Type

To test *H4.a*, honeypots for the following four services are used: CHARGEN, QOTD, SSDP, and RIP. The first setting provides a response that is according to the specification. The second setting is a response that presents the service as being a honeypot by including certain keywords in the response of the honeypot, for example, the words 'honeypot' and 'research'. To test *H4.b*, the two other services are used: DNS and NTP. For these services, the groups are based on redirection to an actual DNS or NTP service, or an emulated implementation depending on the setting used.

4.3.5. Variables for H5: Deployment Differences

Hypothesis *H5.a* and *H5.b* both apply to *where* the honeypots are deployed: at different providers and in different geographical locations. To test hypothesis *H5.a*, multiple locations should be picked, but they should be hosted at the same provider - otherwise, the influence of the provider might be measured, instead of the influence of the geographical location only. To test hypothesis *H5.b*, at least multiple providers should be selected, preferably hosted in the same geographical location.

4.3.6. Variables and the Experiment Design

The different variables and settings that were discussed can be combined into several groups. The amplification factor *(H1)* and the type of response *(H4)* determine *what* the type of response is of a honeypot. The different network quality factors that are related to *H2* apply to the *operational* settings of a system. Finally, *H3* and *H5* include settings that determine *where* and *how* a honeypot is deployed.

These different groups aid the construction of the experiment design. At the core of the experiment are the two variables that determine *what* the response is. These two variables - the amplification factor and the type of response - are used to create four test groups and are called the main experiment. The squeeze settings also determine *what* the response of a honeypot is, but this experiment requires much more data points. For this reason, a separate experiment is used for the squeeze settings.

The operational variables each included three different settings, one of which is a setting that is a normal functioning of a system. This means that the main experiment already provides the data for this control setting of these operational variables. To reduce the number of resources that are required to run this experiment, the main experiment is thus used as a control group for the network quality factor tests. As such, the network quality factor experiments can be modeled as being similar to the main experiment, but with an additional operational setting applied.

For hypothesis *H3*, the difference between the two settings is the number of honeypots that are deployed on the same public IP address. For the main experiment, a honeypot is deployed for each service on the same host, sharing the public IP address. This is done because of the limited amount of resources. This means that the main experiment provides for one of the two settings for this variable and hypothesis. The other variable uses the same groups and settings form the main experiment, but is deployed differently. This allows us to compare the influence of the deployment variable on the different groups that are used for the main experiment.

A similar reasoning is used for hypotheses *H5.a* and *H5.b*. The main experiment is already deployed at certain providers and geographical locations, so this already presents a setting - or multiple settings when the experiment is deployed at multiple providers and locations. As such, no separate experiment is used for these hypotheses. If the same experiment is reused, this allows to answer both hypotheses *H5.a* and *H5.b.* while simultaneously increasing the number of data points that are available for the main experiment and analysis.

Figure 4.1 shows a schematic overview of the discussed four different experiments, how they are related, and which hypotheses can be answered with the different experiments. In summary, the groups in the main experiment are constructed with the settings of *H1* and *H4*. This main experiment acts as a control group for both the tested network quality factors and the differences in deployment. Finally, the squeeze experiment is run separately to be able to do a deeper analysis for the primary hypothesis *H1*.

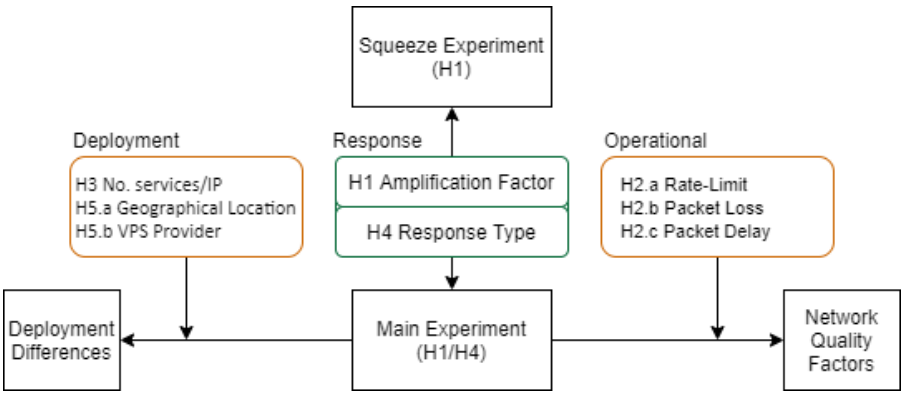


Figure 4.1 - Experiment Overview

4.4. Comparing the Influence of Settings

A method is required to measure the effect of the different constructed settings on adversarial behavior. The primary method that is used, is an analysis of the honeypots that are used in attacks. A single attack consists of one or more honeypots that are used simultaneously to attack a victim. Figure 4.2 shows a schematic overview of this process. If adversaries only select and use honeypots that run with a configuration called A to attack a target and no honeypots that run with a configuration called B, that suggests that adversaries prefer to use honeypots that run with a configuration A above honeypots that run with configuration B.

During analysis, the terminology of flows and attacks is used. A flow describes traffic between a single source and a single destination IP address. Thus, in the example of figure 4.2, there are two flows. During the processing stage, if multiple flows originate from the same source address during the same timeframe, they are combined into a single attack. Thus, in this example, there is one attack, consisting of two flows. Furthermore, for this attack, two honeypots were used in that attack.

Not all detected flows are considered attack flows. Only flows that have more than 10 packets in total and have at least one outgoing packet are included. This threshold is used to exclude both passive flows in the dataset as well as to exclude some services which may send out a burst of packets while operating under normal circumstances.

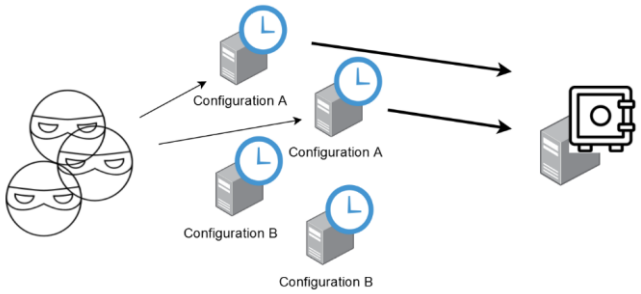


Figure 4.2 - Honeypot selection in a DDoS attack with NTP honeypots

Global Rate-limiting

By deploying a lot of vulnerable servers, we may be participating in large scale DDoS attacks. This means that there are ethical considerations to ensure that our research will only minimally contribute to DDoS attacks. For this reason, global rate-limiting is used that is applied to all the experiments. These limits are notably higher than the limits that are used in the rate-limit experiment. Since these limits are only related to ethical considerations, they are discussed in section 7.2 of the final chapter.

4.5. Test Phase

To the best of our knowledge, there was no previous work that would aid the selection of the different settings for the variables, most importantly for the amplification factor settings. For this reason, a small scale test experiment was run in order to gather some data to tune the settings of the experiment.

The primary goal of the test phase was to test the feasibility of the experiment. To determine the feasibility, the following questions were used:

- Do adversaries find and use the deployed servers in their attacks? If this would not be the case, it would not be possible to use the designed methodology and thus the research would not be feasible.
- How long does it take before adversaries start using the honeypots to execute their DDoS attacks? If the time between deployment and abuse would be too long, that would have implications for the planned allocated time for the experiment.
- Does the framework[17] keep running and provide data reliability during the DDoS attacks? If the designed framework would not reliably perform under the load in a real-world situation, that would mean that the data that would be collected during the active experiment phase could also not be used for analysis. Thus it is of importance that the framework continues to operate under load without interference or a loss of data.

During the test phase, initially, only the services QOTD and CHARGEN were used. For both protocols, the input could be discarded. This meant that implementation was less likely to introduce bugs and that the focus of the test phase did not have to be on the correct implementation of the protocols.

Since there was no baseline information available on the amplification factor and how adversaries react to these differences, the testing values were arbitrarily selected. Table 4.3 shows the number of flows that had more than 10 packets and the response size of both protocols during the testing phase. These results showed that a very low amplification factor (~1) seemingly does not attract adversaries, whereas a value of ~2 attracts some adversaries, and furthermore, that a higher amplification factor attracted the most adversaries.

[17] The framework refers to MD-Honeypot, which was specifically engineered for and during this thesis. The framework is discussed in the next chapter.

Table 4.3 - Test Phase QOTD & CHARGEN Response size and number of flows

	QOTD Response Size	QOTD NBAF Factor	QOTD #	CHARGEN Response Size	CHARGEN NBAF Factor	CHARGEN #
A	~42 bytes	~1	0	94 bytes	~2.5	89
B	~1600 bytes	>= 38	2386	8929 bytes	~212	10318
C	~350 bytes	~8	2390	94 bytes	~2.5	23

This testing phase also provided insight into the time between the deployment and the first attack that occurred. The QOTD honeypots were deployed on the 12th and 14th of July, 2019. All honeypots that attracted at least one attack, detected their first attack on the 19th of July. For CHARGEN, all attacks to the higher amplification honeypots also occurred within the first week. This was useful information because it would mean that in the full-scale experiment there was also a high chance that it would take at most a week before the first attacks could be detected.

4.6. Test Group Construction for the Main- and Sub Experiments

This section includes the configuration settings of the groups for the six different services that are used in the main experiment, deployment differences experiment, and network quality factor experiments. These experiments were discussed in section 4.3.6, experiment design. Recall that the groups for the services are similar: two variables with each two settings are used. The first variable was the amplification factor with a small and a large response, and the second variable was the type of reply with an expected response for the given service and a response that clearly indicates that the system is fake or is a honeypot. Table 4.4 shows an overview of the groups that are constructed from these two variables. These groups are used for honeypots with QOTD, CHARGEN, RIPv1, and SSDP.

Table 4.4 - Overview of two variables and the resulting groups

		Type of Response	
		Real Response	**Fake Response**
Amplification Factor	**Small Size**	A (RS)	C (FS)
	Large Size	B (RL)	D (FL)

The abbreviations RS, RL, FS, and FL will be used further to improve readability and reference to the amplification factor and the type of response. This section further discusses one example of how these groups translate into the specific configuration settings, this is done for the service Quote of the Day (QOTD). The constructed groups for all of the six services are included in Appendix C. Note that both DNS and NTP use a slightly different group order and settings, as both these services include emulated servers. For all groups and services, both the bandwidth amplification factor (BAF) and the network bandwidth amplification factor (NBAF) are mentioned.

Example: Configuration Settings for Quote of the Day Groups

The QOTD protocol discards any input it receives and sends a random quote back. Thus, the response can be triggered by any received packet but is usually one byte - the minimum packet size. Any returned content is a valid response according to the protocol specification.

Group A (RS) has a small random quote with an average size of around 45-50 bytes. The BAF varies between 44-50 and the NBAF between 2.0-2.1. An example quote is the following:

```
"To infinity .... and beyond! - Toy Story"
```

Group B (RL) contains large quotes with an average size of 1450 bytes. This results in a BAF of 1470 and a NBAF of 34.7. Quotes in this group are a selection of Lorem Ipsum[18] texts. For example:

> "Quisque et arcu vitae mi congue sodales. Nullam varius felis turpis, eget semper lectus sollicitudin id. Suspendisse condimentum auctor convallis. Praesent accumsan maximus nisi ac hendrerit. Nam fringilla leo sed lectus commodo laoreet…. Mauris efficitur vehicula erat at maximus. Vestibulum suscipit quam leo, at placerat risus aliquet non. In vel orci sed. - Lorum Ipsum"

Group C (FS) contains a single quote with a size of 53 bytes, indicating clearly that the system is a honeypot. The BAF and NBAF are similar to group A: 53 and 2.2 respectively.

> "This is a honeypot attack detector - Researchers"

Group D (FL) contains a single quote with a size of 1437 bytes, indicating clearly that the system is a honeypot. The BAF and NBAF are similar to group C: 1437 and 34.4 respectively.

> "This is a honeypot system! This is not a real QOTD server! You should not be using this QOTD server! Your IP internet address is logged in a database. We use this server to investigate who connects to it and what happens with these quotes! So it is really best you do not use this server! And if you do, please leave a cool message, since we log all the packets anyway ;) We will publish any misuse of this service on the internet and detect your IP internet address. (Repeated 3x) - Project Honeypot Research"

4.7. Test Group Construction for the Squeeze Experiment

For the services QOTD, CHARGEN, NTP, and DNS, the same groups are used. Each of the groups contain a specific length, and the response is a plain packet with a static string of random characters. Table 4.5 shows the overview of the five groups and the size of the generated static string. The random characters only contain ASCII characters and were generated once before the start of the experiment. For example, the string of squeeze group 2 was the following:

```
WRVBkmUuCyfWgvmSsBHZX5eECujvxarV2365AAuBTmPrAsmPpFwar4V3re3RSMrEgSuxxCfvY9E8ZXrrsKVfyt
wk38
```

Table 4.5 - Squeeze groups and their size

Squeeze Group	Group 1	Group 2	Group 3	Group 4	Group 5
Size	1 byte	90 bytes	200 bytes	400 bytes	600 bytes

For the RIPv1 service, five different routing table responses were used: each group included the RIPv1 routing table exchange header and a number of entries. Squeeze Group 1 only consisted of the header itself and was 4 bytes. Group 2 consists of 4 addresses, with 84 bytes in total. Group 3 has 10 addresses, with 204 bytes in total. Group 4 has 20 addresses, with 408 bytes in total. Finally, group 5 has 30 addresses with 604 bytes in total.

For SSDP, group 1 returns the text 'SSDP' (4 bytes). Group 2 consists of 84 bytes and has the following response:

```
NOTIFY * HTTP/1.1
CACHE-CONTROL: max-age = 1800
SERVER: UPnP/1.0
NTS: ssdp:alive
```

Group 3 is identical to the one of the main experiment in group A (small real), with 272 bytes in total. Group 4 is identical to group B (large real) of the main experiment, with 430 bytes in total. Finally, group 5 is 600 bytes in total and provides a response similar to that of group B, but includes the following data at the end:

```
OTHER-DATA:
CJMbkkZ9NR5hJuGuKR9zcGxrxAPeFBXd5efwjPKphmKs8WjaZWxC7TTHHWAD8CbEE8Mx7gnhvH7gzvjJRjk7rX
ZzhezpDEGzuhGTnVZGBx6ZbPh9CgsjcR86vJRXj49bYsdG5uwUbDj2JcrEZxy8jsfYZr
```

[18] https://lipsum.com/

5

MD-Honeypot Framework Design

Before differences between the constructed settings and groups can be analyzed, first, data needs to be gathered. In previous work, Krämert et al. [44] used AmpPot and Kamoen [55] used Honeytrap for data collection. However, AmpPot was not available for use because the source code was not available, and Honeytrap used different agent sensors that were all redirected towards a central agent. This posed a problem as, in that case, DDoS traffic would be forwarded back and forth from the honeypots to the central agent. With the large scale of testing that is used for this thesis, this would not be a feasible option. The use of bandwidth would overload the central agent and with that, there would be a risk of the system collapsing under its own load. To the best of our knowledge, there are no other frameworks available that fulfill the requirements needed to conduct this experiment. Therefore, for this research, a new framework is created: the MD-honeypot framework.

Together with my colleagues Bart de Jonge and Ahmet Gudek, we developed a new versatile honeypot framework that can facilitate A/B testing with hundreds of honeypots. The newly built framework supports both the TCP and UDP protocol, but since this thesis is only focused on UDP, the TCP parts of the system are only briefly mentioned but not further explained. For additional information regarding the TCP elements of the system, the work by de Jonge and Gudek can be consulted.

The core parts of the system were jointly developed. On a high-level, the following items were mostly developed by the researcher of this work: all UDP related functionality, including the UDP agent dashboard, the UDP rate limiter and dashboards, UDP protocol stack/handler implementation, the handlers for all UDP services, most of the Docker/virtualized integration, the largest share of the dashboard server, the largest share of the Python to JAVA rewrite, a share of the data logging system and the data retrievers for the logging server.

This chapter discusses the process and final product that was required to conduct the experiment of this research. The MD-honeypot framework was constructed in several phases: first, the requirements were set, which is discussed in the first section. Then, the high-level framework was designed and prototyped, which is discussed in section two. Detailed information about the components and their features is discussed in Appendix A, and the available configuration settings of the framework can be found in Appendix B.

5.1. Requirements

The requirements were split up into two parts: functional and non-functional requirements. These requirements were created with the methodology in mind, as well as maintainability and future use. For the requirements, the following list was created:

Functional requirements:

- The honeypot should support at least the following modes of operation:
  - Emulated (for certain protocols): implement a protocol in the system itself so the system can handle a request of a particular protocol directly without external calls of forwarding.
  - Virtualized: The system should support running software in virtualized containers, so it is trivial to encapsulate existing software to run on different machines without the need to distribute different versions of the system. It also provides security benefits as the software is contained in its own container.
  - Tunneling (for TCP): allow incoming packets to be encapsulated and be forwarded to another honeypot so the packet can be handled there. This is useful as it shields the system where the connection comes in from potential vulnerabilities in the set-up. This is especially useful for SSH or telnet honeypots, where an adversary can execute commands on remote systems. A honeypot using this mode is called a forwarder.

- The honeypot should be configurable in different modes for different protocols simultaneously. For example, it could run a UDP service in emulated mode while forwarding (tunneling) TCP SSH traffic towards another honeypot.
- Additionally, it should be possible to run different (UDP) honeypot configurations on different ports simultaneously.
- The system should run on different platforms. It should run at least on Linux and Windows.
- There should be a central logging server to collect (aggregated) information about incoming packets and connections.
- There should be a central configuration server, so configurations can be (re)loaded dynamically without requiring a redistribution of the software or its configurations.
- There should be a simple dashboard to view statistics, so participants that run our software can also view information about the data they contribute to the project.
- Components: the different components of the system should run independently of each other so the components can be deployed at different systems and their resources can be scaled up when required.

Additional function requirements for UDP:

- Rate limiting: to prevent our honeypots from amplifying too much data, there should be a rate-limiting feature that can limit the number of packets and transmitted bytes for each connection.
- Global rate limiting: Different honeypots may be targeting the same targets. Therefore, there needs to be a form of communication between the different honeypots to ensure that there is also a global rate limit towards any particular target that should not be exceeded.
- Packet delay: it should be possible to introduce an artificial delay between receiving and responding to a packet.
- Packet loss: it should be possible to introduce an artificial amount of packet loss.
- Performance: the maximum throughput of a honeypot should be at least 100 megabits of data per second.

Non-functional requirements:

- Choice of programming language: since the development time is relatively short for the thesis project, the programming language should be a language in which we have already worked with extensively, to prevent delays while trying to master a new programming language.
- Choice of frameworks: if possible, choosing proven technology and frameworks is preferred over using experimental software and frameworks.

5.2. Designing the framework

The design process that led to the final system consisted of different steps. First, a system design was created, and languages and frameworks were chosen. This was followed by prototyping and (performance) testing. The prototype was incorporated to run as an assignment during the Network Security Course at the TU Delft. This led to lessons learned that are discussed later in this section.

5.2.1. System overview

With all the requirements set-out, the first step of the design process was to incorporate all the requirements into a single component design overview. After a few iterations, this led to the design, which is illustrated in figure 5.1. The figure shows dependent components with a regular line and optional dependencies with a dashed line. The design choices are now explained. More in-depth details about all the components can be found in Appendix A.

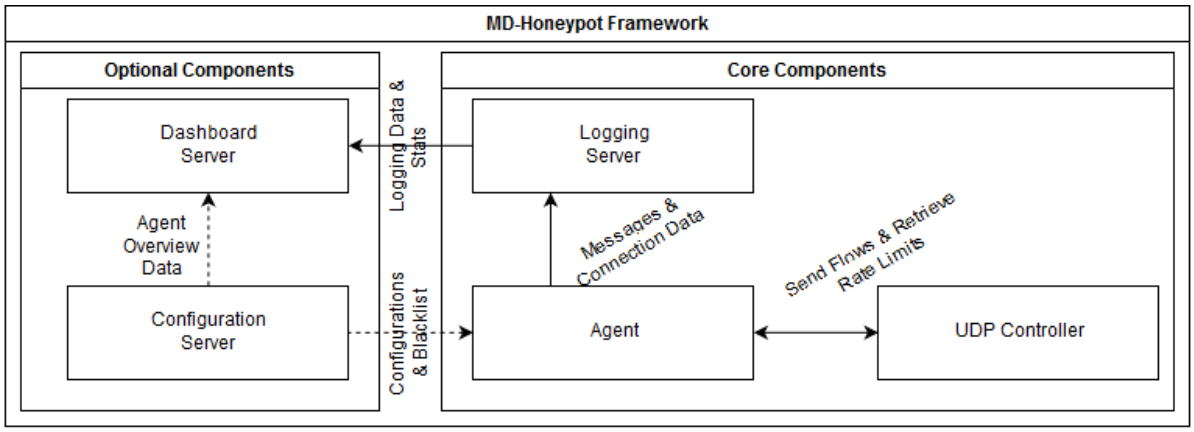


Figure 5.1 - Schematic MD-Honeypot Framework overview

The final design consists of five different components:

- The logging server accepts data from an agent and stores the database in its database. The dashboard also queries the logging server to display statistics and logging data.
- The configuration server can be queried by agents to retrieve a configuration for the agent to run. This component is optional. Agents can also run with a local configuration file. The configuration server also provides a dynamic blacklist.
- De dashboard server is optional but provides a visual overview of current (aggregated) statistics.
- The agent runs in one or more of the previously discussed modes of operation. It can optionally retrieve a configuration file at startup. It always logs its data to the logging server. While this can be disabled, the agent does not have any local logging capabilities. This would violate the principle of separate dependencies. However, it is possible to run a logging server locally on the same system as the agent. Finally, if one of the running operation modes is tunneling, it will forward all the traffic for the specific service to another agent to handle. The agent can, in this case, be seen as a proxy server between an incoming connection and the actual agent handling that connection.
- UDP Rate Controller: this controller maintains an overview of all current flows at all the different agents. If the sum of the flows exceeds a certain value, it will make sure that all the agents will apply a much stricter rate limit towards the target so that the aggregate flow towards the target will still stay below specified limits.

5.2.2. Choice of language and frameworks

Among our group, both Python and JAVA were languages that we had sufficient experience with to execute this engineering project. As I already created a simplistic version of a honeypot system used to forward connections towards a virtualized container during another master course, the preference was to continue to use Python to be able to reuse parts of this system. In addition, Python allowed for the handling of raw sockets, whereas JAVA could not.

On the other hand, JAVA is much more performant and undoubtedly fast enough for our needs. However, Python is a lot quicker to develop with. Eventually, the choice was made to use Python as the primary programming language for this framework. If, after prototyping, the agent would not provide the required performance, only the agent could be rewritten in JAVA while the other components could still work independently in any of the other languages.

As backend for the logging server, the software MariaDB (a MySQL fork) was selected. This database engine has decades of research and optimizations and is already known to be highly performant and scalable. It has connectors available to virtually any programming language and has excellent documentation. We also have all worked with this database and the type of queries before. As such, it was concluded to be the best choice.

For the communication between the different parts of the systems JSON over HTTP(S) was picked as an exchange format. This form of communication is also the most widely used stateless communication protocol to communicate between services to date. Again, we all had experience in working with it too. These protocols are also supported in many programming languages.

5.2.3. Prototyping

To test the first prototype, the prototype was deployed at home devices at student homes during a master course at the TU Delft. As part of the subject of internet scans, students had to deploy and run our software for 72 hours and then analyze the data and write a small report about the gathered data. To not expose anyone's home network to unwanted DDoS risks, replies for the UDP protocol were disabled. For TCP, all traffic was forwarded to an agent on an external network.

This first test phase provided valuable information and lessons learned. For example, in most cases, automatic port forwarding did not work correctly. In addition, the deployment and system compatibility turned out to be a much bigger issue than initially expected. Although Python has an automatic package manager installation system, many compatibility issues arose between different minor versions of Python and their related libraries. These issues also influenced the decision to rewrite the agent part of the system in JAVA. This will further be discussed in the next subsection.

5.2.4. Performance tests

While the prototype was deployed and running to gather test data, the performance of the system was further tested. These tests were run with the now completed protocol implementations using requests and responses that are also used in the wild to execute attacks. The initial tests were executed on a cloud with limited resources[19]. The first explored test case was an NTP server running in a container, with the agent in virtualization mode. After optimizations, the maximum achieved total throughput was around 8 megabits per second. At that point, the CPU was fully utilized.

[19] The CloudVPS was hosted at Digital Ocean and had 1GB of ram and one Intel(R) Xeon(R) CPU E5-2630L v2 @ 2.40GHz.

This led to a lengthy process of different tests to find the root cause of the limited performance. The first test tested a barebone implementation of sending and receiving packets in Python. In this implementation, the application would receive a packet, forward it to the virtualized container, and then sends back the response to the source - without any further processing. Even then, the maximum throughput was not higher than 10 to 15 Mbit/s. The problem was not necessarily all the extra functionality that we created besides sending and receiving packets. This raised a suspicion that the unexpected low performance was related to the programming language Python. The next test consisted of the same barebone implementation in JAVA. Although getting a higher throughput - around 20 Mbit/s - the results still surprised us. The throughput was still nowhere near the line rate, which was the expected and required throughput.

The same tests were conducted on barebone machines equipped with multiple[20] cores to ensure the test system was not the culprit. The throughput was a lot higher - around 25 Mbit/s for Python and around 80 Mbit/s for Java. Still, these machines were supposed to be a lot faster, and anything under a few hundred Mbit/s was not in line with expectations. The next conducted test used iPerf[21] to ensure that the used programming language was not the problem. iPerf is used as the de-facto tool for network performance testing in the network industry sector.

iPerf was configured to run in a bidirectional mode while sending UDP packets. During the test, the goal was to maximize the bandwidth throughput while having less than 1% packet loss. With iPerf, the line rate of around 1000 Mbit/s was quickly reached. However, this was with a default packet size of 1470 bytes. While re-running the tests with different packet sizes, radically different results came out. Although the throughput was still higher than on the cloud system, the throughput dropped significantly towards around 275 Mbit/s. When taking into account that the packets skip one hop in this test, the comparable throughput would only be half, around 140 Mbit/s.

This was odd, but during the tests, one system indicator provided a lead. The CPU utilization on a system is split into different categories. Figure 5.2 shows our system utilization during the test. The red bars indicate that a large share of CPU resources was spent in the kernel and not in the application. During the tests, the amount of sent and received packages were also recorded. This data showed clearly that the throughput in terms of bandwidth was not the culprit, but that instead, it was the number of packets which could be processed each second.

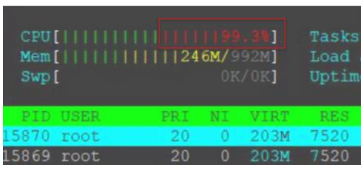


Figure 5.2 - CPU load overview

These hints led to a post written by Majkowski at Cloudflare [58], which discussed this niche problem. In summary, the only feasible solution for our use case was to use multiple threads. At this point, Python was no longer an option. While Python does support multithreading, it effectively does not run multiple threads simultaneously. As a design feature, only one thread can be active at the same time. This system in Python, ensuring this safety feature is called the Global Interpreter Lock (GIL). Unfortunately, this cannot be disabled. This meant that the agent had to be rewritten in another language that was not Python.

Finally, this led to the conclusion to rewrite the prototype of the agent in JAVA. The other components remain in the Python codebase. Switching to JAVA for the agent also solved the deployment and dependency issues - as we could distribute a single binary file with JAVA.

[20] The systems had 24GB of RAM with two Xeon X5650 @ 2.67Ghz-3.0Ghz processors. In total this is 12 cores and 24 threads.
[21] https://iperf.fr/

6

Data Analysis & Results

After the initial test runs, the final data collection phase started on August 29th, 2019. During the first week of the data collection phase, all honeypots were configured to listen passively for incoming traffic and to not respond. This allowed us to test the system on a large scale to prevent scaling issues while starting the active phase of the experiment. After the first week, all honeypots were started with their corresponding configuration. The active phase started on September 6th and ran until September 27th - allowing three weeks for the configured honeypots to be used by adversaries.

As mentioned in the methodology chapter, each honeypot is running on a virtual machine (VM). All machines that were not used for the single service experiment, ran honeypots for all six services. Furthermore, note that all reported dates and times in this thesis are in the UTC time zone.

This chapter starts with a section containing an overview of the experiment and its duration. This is followed by five sections corresponding to the five primary hypotheses: the influence of the amplification factor (H1), the influence of network quality factors (H2), running multi vs single services (H3), running fake services (H4) and finally the influence of the honeypot placement (H5). Thereafter, two special attack types are discussed: multi-vector and subnet attacks. The effectiveness of attacks is discussed in section 6.8. The retention period - the period where attacks continue while the honeypots no longer provide a response - is discussed with its implications in section 6.9. Finally, further developments and findings are discussed in section 6.10.

6.1. Experiment Overview

The experiment overview starts with a subsection that describes the size and duration of the experiment. The subsection thereafter discusses which addresses were excluded to receive a response. The third subsection shows an overview and timeline of the number of attacks that occurred during the data collection phase. Thereafter, the time between deployment and attacks are discussed. Finally, the honeypot convergence is estimated, providing an estimation of how the gathered data represents the attack landscape.

6.1.1. Experiment Size and Duration

This subsection presents an overview of the data to get this size and cover of the experiment established. Hundreds of public IP addresses were required so that there were enough data points available for all the different configuration types. A combination of self-sourced machines together with crowd-sourced virtual machines were used. Each virtual machine (VM) had only one public IP address - which allowed it to run a honeypot for the six different services.

There were 549 servers available to run honeypots on for the experiments. 171 of those machines ran on the Amazon AWS platform located in North America - the only available location for the accounts that provided the machines. 24 machines ran on the Microsoft AZURE platform located in Europe. 166 machines ran on Digital Ocean located in Europe, and 124 machines ran in the Google Cloud in Europe. Finally, there were 24 OVH servers running in Europe, 20 OVH servers running in Sydney, Australia, and 20 servers running in Singapore, Asia.

All machines ran CentOS version 7.4, except for the machines that ran at AWS. The AWS machines ran on Ubuntu 18.04. For all machines, both the firewall of the provider as well the firewall on the machines were configured to allow any incoming traffic. Finally, to ensure that the data coming from several locations can be easily analyzed, all the agents and machines were configured to use the UTC time zone.

Figure 6.1 shows the number of machines running over the course of the experiment. Due to an unforeseen error, nothing was logged for the majority of honeypots between September 11th 12:00 and September 13th 18:00. This affected only the data

logging process, but did not affect the response queries - adversaries targeting the servers, fortunately, could not notice a difference. After this incident, the number of reporting machines was back up to 549.

Due to resource limitations, not all machines were able to run until the completion of the experiment. The first machines that ran out of resources were 80 of the 124 servers that were running in the Google Cloud, on September 15th. 70% of these servers were running the experiment for *H3* (amount of services running on 1 IP). Therefore, this reduction in the number of available honeypots for the experiment primarily impacted only one sub experiment. In total, 27 servers that ran on Amazon's AWS services also ran out of resources before the experiment concluded.

Thus, not all honeypots have an equal duration in which they operated. This could influence the comparison between the groups and experiments because the honeypots did not all have the same amount of time to record data. To account for this difference, results are normalized with the amount of exposure time a server has had. The exposure time is the time during which the server was active and running, and during which time data is logged.

Figure 6.1 - Number of running VMs during the experiment

The machines were distributed with different settings and groups, each targeted to gather data for a specific hypothesis. There were at least 12 VMs for each of the five squeeze groups. All of the VMs that ran the squeeze honeypots have been running from the beginning until the end of the experiment. The other four groups saw a decline in the number of available honeypots after September 15th. 165 VMs were used for the main experiment. Each of the network quality factor experiments (Rate Limit, Packet Delay, Packet Loss) used 60 different VMs. 24 VMs were used for each of the six different services in the single service experiment.

Tables 6.1 and 6.2 show the overall and aggregate statistics of all the incoming connections, flows, and attacks. The scan flow count includes passive traffic as well as scanners that scanned for open services. The 20,259 uniquely detected attacks consist of ~2 million flows. Note that although the scan flow count is much higher than the attack flow count, a scan is often a single packet whereas an attack flow has many more packets and also has a much longer duration. In section 4.4, it was discussed that an attack consists of one or more flows. This explains why the total attack count is much lower than the attack flow count.

In total, there were 11,392 different unique targets. 2462 targets were attacked at least twice. The attack that used the most unique honeypots is 390. This means that we likely detected some large attacks. The total incoming traffic was 934 GB, this denotes unamplified traffic. During this time, more than 15 billion packets were received.

Finally, the sum of the duration of all flows that contributed to attacks is roughly 14.25 years. The sum of the duration of all attacks was roughly 61.38 days. The sum of these durations is significantly longer than the runtime of the experiment. Overall, it appears that there is enough data gathered to analyze the research question and to answer the hypotheses of this research.

Table 6.1 - Overall Statistics

General Statistics	
Scan Flow Count	3,006,956
Attack Flow Count	1,966,512
Attack Count	20,259
Unique Targets	11,392
Most Unique Honeypots Used in Attacks	390
Total Unique Packet Count	16,900

Table 6.2 - Aggregate Statistics

Aggregate Statistics	
Total Incoming Bytes	933.8 GB
Total Incoming Packets	15,222,814,223
Total Attack Flow Time	14.15 years
Total Attack Time	61.38 days

6.1.2. Blacklisting Research Projects

During the passive data collection phase[22] of the experiment, the reverse DNS records for each incoming connection were automatically recorded. The reverse DNS records help to detect whether certain addresses should be excluded from receiving a response in a later stage as they often describe a target or source. For example, the address 145.94.***.*** points towards scanner2.scanning.****.net. By manual inspection of such a domain[23] we notice that this is an Internet scanning project - and as such, we want to prevent such addresses from receiving a reply to prevent our experiment from accidentally influencing other cybersecurity research projects. By inspecting more of these reverse DNS records, more research projects or companies were found that included the word 'scan' or 'research' in the reverse DNS name. To ensure that our experiment would not interfere with these projects, they were excluded and added to the blacklist.

6.1.3. Attacks Overview

This subsection presents an overview of the attacks that were detected during the experiment. The first attack was detected on September 6th at 19:01, 23 minutes after deployment. The last attack was detected and ongoing at the moment the honeypots were taken offline on September 27th, around 15:00. Figure 6.2 shows the number of attacks during each 12-hour interval of the experiment on a logarithmic scale. For the deprecated and legacy protocols CHARGEN, QOTD, and RIP, the number of detected attacks appear to be low in comparison to the other protocols. During several time windows, no attacks were observed for these deprecated and legacy protocols. For the other protocols, there were no time windows where no attacks were detected.

NTP, in particular, saw an uptick in the number of attacks since September 13th. An even larger increase in the number of NTP attacks occurred at the end of the experiment, but the experiment had to be shut down at that time, so it is unclear if this trend continued. Section 6.9 provides further analysis to make an estimation of how the number of attacks continued after September 27th.

Some attacks also used honeypots from different services simultaneously during an attack. In total, 278 of the 20,259 attacks (1.37%) can be categorized as such an attack, which is called a multi-vector attack. These attacks are separately analyzed in the multi-vector section 6.8.1.

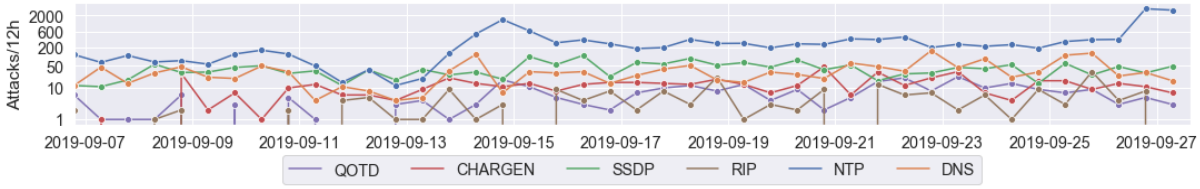


Figure 6.2 - Number of attacks per 12 hours during the experiment

6.1.4. Time from Deployment to First Attack

When the honeypots changed their configuration from the passive listening configuration to the active experiment configuration, the honeypots started to send responses to all the incoming queries. From that moment on, adversaries could detect the honeypot and then could have decided to use the detected honeypot for their attacks.

Figure 6.3 shows the number of hours that passed between the first detected scan and the first detected attack for honeypots that ran the main experiment. There are observable differences between the different services. For QOTD, almost all servers are already used in attacks less than a day after deployment. NTP and CHARGEN see a gradual increase over time, whereas both SSDP and RIP have hardly any honeypots that are used in attacks on the first day, but then see a sudden increase. The honeypots that run DNS are exploited much later than the honeypots of the other services.

In Rossow's work with AmpPot [44], all honeypots were already used in attacks within 24 hours after deployment. Thereafter, the honeypots saw a steady increase in the five days after, whereafter the number of attacks remained at a steady level. The results in this work are different from these observations, as not all honeypots are already used within the first 24 hours after deployment. Only QOTD shows that almost all honeypots were used within 24 hours.

[22] between August 29th and September 6th 2019

[23] for privacy reasons, the actual name and IP of the example is blinded with asterisks

However, this work also used many more honeypots - 549 in comparison to 21. These new findings suggest that the time-until-attack greatly varies for each service, but also that it may take longer than a day before all honeypots are used in attacks. This helps to correct expectations for future work.

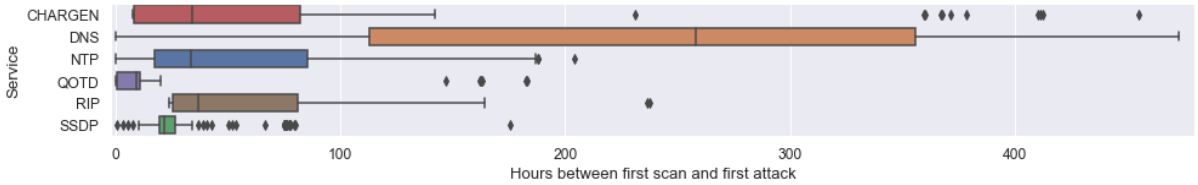


Figure 6.3 - Boxplot - Number of hours that passed between the first scan of a honeypot, and the first detected attack of that honeypot.

6.1.5. Honeypot Convergence

To ensure that the reported findings and analyses apply to the entire attack space, we need to estimate the representativeness of the gathered data. Krämert et al. [44] used the convergence measure method in their work for this assessment. The convergence measure method works as the following: a random honeypot and the attacks it has detected is selected. Then, repeatedly, another random honeypot is selected, and the number of attacks that the newly selected honeypot has detected that were not yet previously detected by the other honeypots is used to measure the convergence. This shows how many more attacks have been detected with each additional honeypot.

This method can be used to make several estimations. First, the overall convergence is analyzed. This will answer the question of the representativeness of the gathered data, and how many honeypots are required to detect most attacks. A similar analysis is also used in section 6.6 to test hypotheses *H5.a* and *H5.b.* These hypotheses state that there is a difference between different geographical locations and providers and the attacks that are detected.

For the analysis of the convergence, each of the six services is analyzed separately. The described method of calculating the convergence is repeated 50 times. During each sample, the honeypots are ordered randomly. There are two factors that influence the calculation of the convergence. The first factor of influence are subnet attacks[24], that comprise a large share of the NTP attacks. Thus, these attacks are combined so that an attack targeting a subnet is considered to be a single attack. Secondly, the honeypots that provide high and low amplification are analyzed separately[25]. To also exclude other external variables, only the honeypots in the main experiment are used for the convergence calculation. These are the honeypots that ran with no additional settings. Finally, this analysis generalizes the attacks to the victims that were targeted. In other words, the victims that other honeypots did not detect are compared. This generalization reduces the influence of a select group of targets that were attacked much more than targets that were only attacked once or a few times.

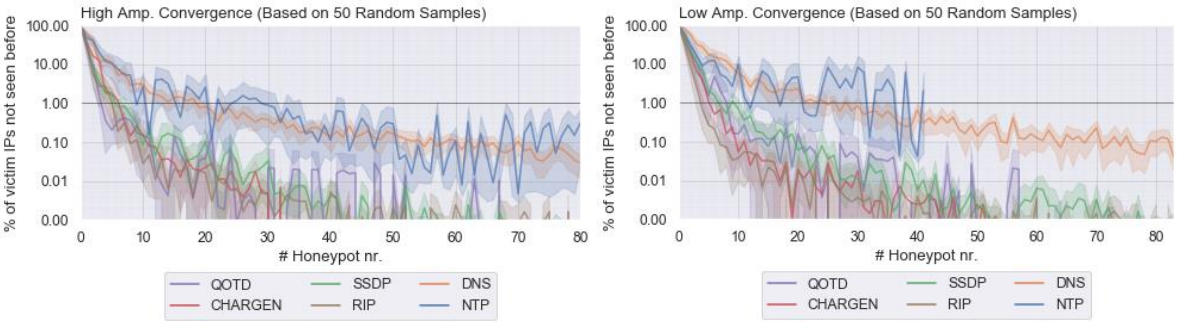


Figure 6.4 - Honeypot convergence

The convergence of the high amplification honeypots on the left, and the convergence of the low amplification honeypots on the right. The thicker colored lines are the averages of the 50 samples, and the colored bands around the lines show the 95% confidence interval of the samples.

Figure 6.4 shows the convergence of the honeypots. There is a noticeable difference between NTP/DNS and other protocols. For RIP, CHARGEN, QOTD, and SSDP, each honeypot after the 10th honeypots do not detect more than 1% new victims that are attacked compared to the first 10 honeypots that are used. In fact, the share of new victims that are detected drops even

[24] See the glossary, subnet attacks are further analyzed in section 6.7.2.

[25] section 6.2 analyzes the influence of the amplification factor

further thereafter. With more than 50 honeypots, the additional victims that are detected drops further below 0.01% of all victims. Thus, it appears that scaling up with many honeypots does not necessarily lead to many more attacks detected. This means that even smaller experiments that do not have the ability to run hundreds of honeypots, will still be able to gather information that captures a majority of the attack landscape.

The difference between NTP and DNS can be explained by comparing the typical number of honeypots that were used in the attacks for the differences servers. For NTP and DNS, this number was much lower than with the other protocols. Because the typical number of honeypots was lower, the attacks were detected at a smaller number of honeypots, thus the chance that a honeypot already detected an attack was lower. For a similar reason, the convergence for high amplification NTP honeypots is better than for the low amplification NTP honeypots. The average number of honeypots that were used in NTP attacks that used low amplification honeypots was lower.

Krämert et al. [44] measured a convergence where more than 11 honeypots added less than 5% of new attacks and concluded that that was enough honeypots to cover most attacks. However, these numbers present an average of different services. As shown in this analysis, the results vary between the services. For some services, the number of required honeypots is lower than for others to get the same cover.

In conclusion, the results of this work show a similar (NTP/DNS) or better (RIP, CHARGEN, QOTD, SSDP) convergence result than previous work. This suggests that the data that is available to analyze the hypothesis is sufficient to make a general conclusion. However, these findings indicate that it is important to analyze the convergence for different protocols separately, and that it is also important to analyze the average number of honeypots used in these attacks to put the convergence into perspective.

6.2. H1 - Amplification Factor

Hypothesis *H1* states that there will be a difference in the abuse frequency between the honeypots that provide a higher amplification factor than their low amplification factor counterparts. Thus, we try to disprove the null hypothesis that states that there is no significant difference in the abuse frequency between the honeypots that provide different amounts of amplification. To test this hypothesis, different sizes of responses were constructed in the methodology section: for the primary experiment, honeypots are configured to provide a low amplification or a high amplification response. In the squeeze experiment, five different settings are used ranging from a very low amplification to a high amplification. This section will analyze the differences between these groups to answer hypothesis *H1*.

This section starts with an explanation and reasoning for the method of analysis. This is followed by two subsections that compare the different groups in the main experiment and the squeeze experiment respectively. Thereafter, several observations that come forward during the amplification analysis are analyzed in more detail. These observations include an analysis of the (number) of honeypots that were used in an attack, the preference for the type of honeypots, and finally, the content of the received packets.

6.2.1. Analysis Methodology Amplification Factor

In methodology section 4.6 two variables were discussed that were used to create four groups. The two variables were the amplification factor and the type of response. This led to four groups, and resulted in groups A and C with a small response size, and groups B and D with a large response size. This means that to compare the amplification factor, two sets of groups can be compared. Although groups C and D also had an additional variable applied (the type of response), they are compared in addition to comparing groups A and B. For the analysis of the influence of the amplification factor, data is only used from the primary experiment that had no further operational settings applied to it.

The four mentioned groups are used for QOTD, CHARGEN, RIPv1, and SSDP - the details about their content were included in Appendix C. For NTP and DNS, the construction of groups was different because these services are also used to analyze the difference between emulated and non-emulated services. Still, the different groups of NTP and DNS also differed in their response size, which allows a similar analysis as with the first four mentioned services. For NTP, group A provided the lowest amplification factor, then group B and then group C. Group D did not provide amplification. For DNS, the amplification factor of the groups is similar to the first four mentioned services: groups A and C provide a lower amplification than groups B and D.

The primary experiment only provides data to do an analysis between two type groups: a low and a high amplification group. This allows only to conclude that one group is preferred to be used more in attacks than the other group. To extend upon this experiment, the squeeze experiment was constructed that had five different amplification groups ranging from a low to a high amplification factor. This allows gathering a more fine-grained insight into the differences between honeypots that provide higher and lower amplification.

To disprove the null hypothesis for *H1*, and to prove that there is a statistically significant difference in the abuse rate between the honeypots that provide a different amplification factor, the average attack rate per honeypot per day is used for each of the different services. Thus, each honeypot contributes to a single data point. As mentioned in section 6.1.1, not every honeypot has run its configuration for the same amount of time - for example, the missing logs early in September or the machines that ran out of resources before the end of the experiment caused that some honeypots had less data logged than others.

To account for these differences, each honeypot is normalized by its exposure time. The exposure time is calculated by using 8-hour dayparts. If any data is logged during this time window, the server is considered to have been active during that time without disruptions. Finally, the total number of attacks is divided by the aggregated number of the day parts of the exposure time, leading to a daily average number of attacks for each honeypot.

These honeypots are grouped by the different experiments and groups that were used. Then, the distributions of one group of data points are compared against another group. To determine a fitting statistical test to compare the distributions, the first question to answer was whether the data was distributed normally. In order to answer this question, the Shapiro-Wilk normality test is used (with $p < 0.05$), which works well even with a low number of samples [59]. 78% of distributions in the main experiment are not normally distributed, and for the other 22%, there is no definite conclusion. For the squeeze experiment, 38% of the distributions are not normally distributed and for 62% it could not be determined if the data is normally distributed. Because the majority of data is not normally distributed, or there is no certainty about the normality of the data, a non-parametric statistical test is used. Furthermore, only distributions between the two groups are tested.

This leads to the choice of using the Mann-Whitney U test to test whether two independent samples were sampled from the same distribution.

6.2.2. Main Experiment

For the analysis of the main experiment, the six services are analyzed separately. Each service includes a figure that shows the distributions of the number of daily attacks each honeypot running in the four groups received. If hypothesis *H1* is correct for the service, it is expected that the distribution of the lower amplification honeypots are centered around lower values than the distribution of the higher amplification honeypots. Furthermore, it is then also expected that the distributions between the lower and higher amplification groups are significantly different.

To test for significant differences between the two tested distributions, the Mann-Whitney U test is used. The chosen alpha level is at most 0.05, otherwise, the distributions are not considered to be significantly different. The number of data points varies between 30 and 43. Finally, note that for this analysis, only single-vector attacks are considered. Multi-vector attacks are analyzed separately in section 6.7.1.

Each of the figures will show a boxplot of the distribution of the daily attack rate per honeypot, grouped by the configuration that they were running. If hypothesis *H1* is assumed to be correct, it is expected that the distributions of the larger amplification honeypots are centered around higher values than the lower amplification honeypots.

RIP

Figure 6.2.1 shows that the boxplots of the lower and higher amplification groups overlap. RIP does not show any significant ($p < 0.05$) differences in the distributions of the daily attack rates. The median of the daily attack rates is between 5.00 and 6.57 attacks a day. For RIP, we cannot conclude that there is any significant difference between higher and lower amplification honeypots.

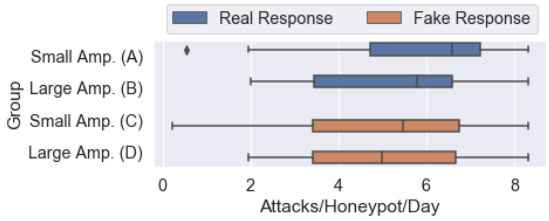


Figure 6.2.1 - RIP Attacks per Honeypot per day

CHARGEN

Figure 6.2.2 shows that the boxplots of the lower and higher amplification groups overlap. The distribution of the average attack rates for CHARGEN are all similar, and there is no significant ($p < 0.05$) difference between the groups A and B or the groups C and D. The median of the daily attack rates vary between 8.71 and 10.79. For CHARGEN, we cannot conclude that there is any significant difference between higher and lower amplification honeypots.

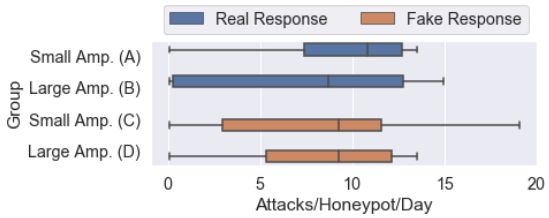


Figure 6.2.2 - CHARGEN Attacks per Honeypot per day

QOTD

Figure 6.2.3 shows that the distributions of the lower amplification honeypots are centered around much lower values than the distributions of the higher amplification honeypots. The distributions of the daily number of attacks are also significantly different ($p < 0.01$) between the groups A and B, and significant ($p < 0.01$) between the groups C and D. The median of daily attacks is 1.63 for group A and 9.21 for group B. For group C, the median is 1.93 and for group D it is 9.18.

Thus, both the small response groups attract fewer attacks than their higher amplification counterparts.

We can conclude that for QOTD, a higher amplification honeypot attracts significantly more attacks than lower amplification honeypots.

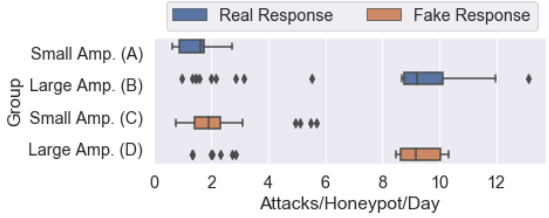


Figure 6.2.3 - QOTD Attacks per Honeypot per day

SSDP

Figure 6.2.4 shows that the distributions of the lower amplification honeypots are centered around lower values than the distributions of the higher amplification honeypots. The distributions of the daily number of attacks are also significantly different ($p < 0.01$) between both group A and B and between the groups C and D.

The median daily attack rate is 17.15 for group A, and 38.06 for group B. The median daily attack rate for group C is 30.49 and 37.69 for group D.

We can conclude that for SSDP, a higher amplification honeypot attracts significantly more attacks than lower amplification honeypots.

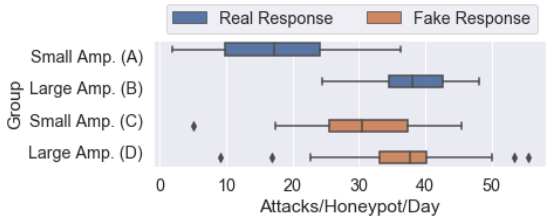


Figure 6.2.4 - SSDP Attacks per Honeypot per day

NTP

The honeypots for NTP that ran configuration D did not attract any attacks at all. This is not necessarily surprising because group D consisted of patched servers and did not provide an amplified response. Figure 6.2.5. shows the distribution of the number of daily attacks for the other three groups, group A, B, and C.

The smaller amplification honeypots group attracted significantly fewer daily attacks than the higher amplification honeypots B and C ($p < 0.01$).

The median of the attack rate for group A was 0.71 attacks/day whereas the medians of groups B and C were 21.89 and 2.16 respectively.

We can conclude that for NTP, a higher amplification honeypot attracts significantly more attacks than lower amplification honeypots.

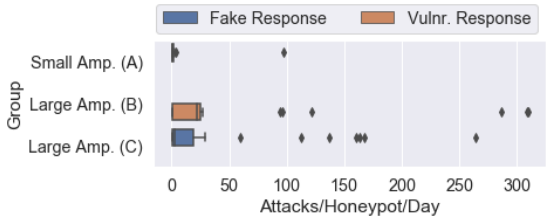


Figure 6.2.5 - NTP Attacks per Honeypot per day

DNS

Figure 6.2.4 shows that there is a difference between low amplification DNS honeypots and high amplification DNS honeypots.

The closed resolver (group A) that provides a low amplification is significantly ($p < 0.01$) less intensively used than the open resolver (group B). The mean of daily attacks of the closed resolver was 0.16 whereas for the open resolver it was 0.33.

The emulated response using 244 bytes (group C) is also used significantly ($p < 0.01$) less than the emulated version giving 351 bytes (group D). The mean of daily attacks on the emulated response was 0.20 whereas the mean for the larger amplification was 0.61.

Although with this last difference, care needs to be given to the fact that this could also be due to the fact that one provided a DNS-like response, whereas the other did not. This will be discussed in section 6.5.

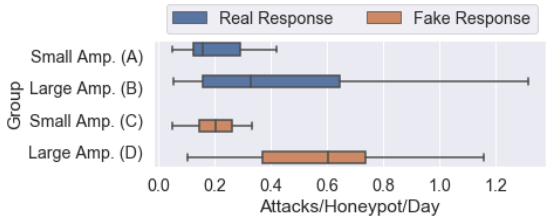


Figure 6.2.6 - DNS average attack rate

Overview

This subsection has shown that there are significant differences in the rate of daily attacks that the honeypots receive. However, these differences are not significantly present among all the services. Table 6.2.1 provides an overview of the acceptance hypothesis *H1*. For RIP and CHARGEN, there is no evidence that suggests that honeypots of either service are attacked more or less depending on its amplification factor. Among the other services, SSDP, DNS, NTP, and QOTD, there are significant differences, but it depends on the service how substantial these differences are. Thus, hypothesis *H1* is accepted, but the results suggest that not all adversaries are operating with similar strategies.

Table 6.2.1 - H1 Acceptance Overview - Honeypots that provide a higher amplification factor will be used *more* in attacks than their low amplification factor counterparts

Service	RIP	CHARGEN	QOTD	SSDP	NTP	DNS
***H1* Accepted**	No	No	Yes	Yes	Yes	Yes

6.2.3. Squeeze Experiment

The main experiment consisted of two amplification factor sizes: a low and a higher value. The squeeze experiment uses five different amplification values. The values differ from service to service and were discussed in section 4.6. Squeeze group 1 provided the lowest amplification factor, whereas squeeze group 5 provided the highest amplification factor. All groups for all services have exactly 12 data points. The only exception is RIP, that only had 10 data points for the first squeeze group.

The previous subsection showed that there were significant differences between low and high amplification for four of the six services, but not for RIP and CHARGEN. For the squeeze experiment, similar results are to be expected if the hypothesis is true, and it is expected that there is an order from the lowest to the highest amplification server. In other words: a higher squeeze group number is always expected to have a higher average attack rate than a lower squeeze group number. The analysis in this subsection follows that of the previous subsection on the main experiment. This means that a higher squeeze group is expected to be centered more to the right in the figures, compared to the lower squeeze groups. The six services are analyzed separately in the same order.

The interpretation of the included figures is similar to the figures of the previous subsection. The figures show boxplots of the distributions of the daily attack rates, where each honeypot contributed to a data point. If hypothesis *H1* is assumed to be correct, the distributions of the higher squeeze groups are expected to be centered around higher values than the distributions of the lower squeeze groups.

RIP

For the main experiment, there were no significant differences between any of the groups and median of the daily the rates varied between 5.00 and 6.57. The squeeze experiment shows similar behavior for the groups 2 to 5. The median of these groups lies between 3.42 and 5.86. Figure 6.2.7 shows the distribution of daily attack rates for the honeypots in the different squeeze groups.

Contrary to the other groups, squeeze group one shows a significant difference ($p < 0.01$) in comparison to the other 4 groups, with a median of daily attack rates of only 0.11.

The smallest response only included a partial RIP-header instead of only the smallest response. As such, we cannot be sure if the difference is due to the amplification factor or because the response was not a valid RIP response. Either way, the difference is most certainly due to one or the other.

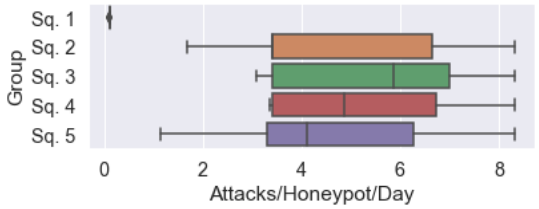


Figure 6.2.7 - RIP Attacks per Honeypot per day

CHARGEN

Similar to the main experiment, for the squeeze experiment, CHARGEN also does not show any significant ($p < 0.05$) differences in attack rates. Figure 6.2.8 shows the distribution of the number of attacks per day per honeypot. The medians of the distributions vary between 9.21 and 9.55.

This is interesting because the first squeeze group has a Network Bandwidth Amplification Factor (NBAF) of less than one. Thus this suggests that adversaries that use the CHARGEN servers for attacks are ignoring the size of the response.

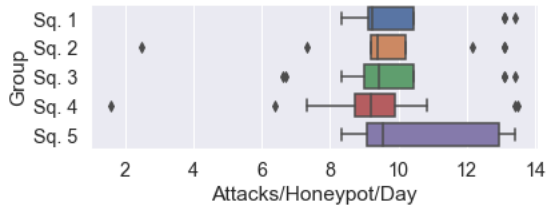


Figure 6.2.8 - CHARGEN Attacks per Honeypot per day

QOTD

Figure 6.2.9 shows the distribution of the daily attack rates per honeypot for QOTD. The figure shows that there are differences for QOTD, where a higher amplification group attracted more attacks on average than the lower amplification groups.

There is a significant ($p < 0.01$) difference in the orders Squeeze 1 < Squeeze 2 < Squeeze (3, 4 & 5). Squeeze 1 attracted the least amount of attacks: the median was 1.56 attacks/day on average. Squeeze 2 attracted more attacks with a median of 2.32, and finally, the median of the last three groups attracted was between 9.19 and 9.61.

For QOTD, the amplification factor that attracts the most attacks is thus somewhere between 90 and 200 bytes. Note that for group 1, the data points seem to be especially concentrated in comparison to the other groups.

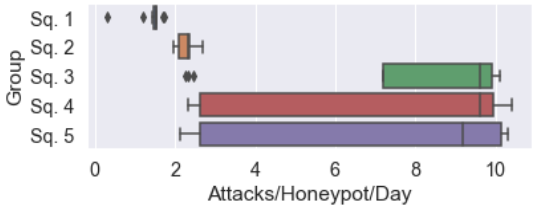


Figure 6.2.9 - QOTD Attacks per Honeypot per day

SSDP

Figure 6.2.10 shows the distribution of the daily attack rates per honeypot for SSDP. Here, a different pattern is visible.

SSDP shows three distinct significant differences ($p < 0.01$). Both groups Squeeze 1 & 2 elicit the lowest attack rates: the medians of the number of daily attacks per honeypot are 3.74 and 4.29. Squeeze group 3 has a mean of 18.18, and finally, Squeeze 4 & 5 elicit the highest number of daily attacks per honeypot per day: the means of these groups are 38.41 and 36.51.

Since there is no significant difference between groups 4 and 5, the amplification factor tipping point is probably somewhere between groups 3 and 4, this is between 272 and 430 bytes.

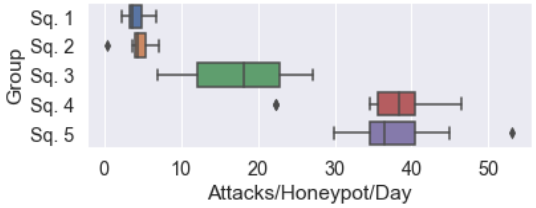


Figure 6.2.10 - SSDP Attacks per Honeypot per day

NTP

Figure 6.2.11 shows the distribution of the daily attack rates per honeypot for NTP. The difference between the first three groups and the last two are multiple factors apart.

NTP shows three significant differences between the groups: the median of squeeze group 1 is 0.37, the median of squeeze group 2 is 0.71, and the median of squeeze group 3 is at 1.05. Squeeze 4 and 5 have medians of 223.36 and 258.08. There is a significant ordering between Squeeze 1 < Squeeze 2 ($p < 0.01$), Squeeze 2 < Squeeze 3 ($p < 0.05$) and Squeeze 3 < Squeeze 4 & 5 ($p < 0.01$).

Since there is no significant difference between groups 4 and 5, the amplification factor tipping point is probably somewhere between groups 3 and 4. This is a response size between 200 and 400 bytes.

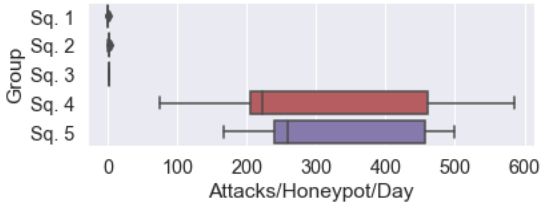


Figure 6.2.11 - NTP Attacks per Honeypot per day

DNS

Figure 6.2.12 shows the distribution of the daily attack rates per honeypot for DNS. By grouping together Squeeze 1, 2, and 3 - a significant difference ($p \leq 0.01$) can be detected between the combined Squeeze 1-3 group, and group 5.

The median of the combined group is 0.37 attacks a day, whereas group 5 has a median of 0.71. Note that the honeypots for DNS in the squeeze experiment only return a static random string of data and not a DNS-like response.

Since groups 4 and 5 do not show completely similar behavior, it is not clear if the boundary was already reached, or that an even higher amplification factor would elicit even more attacks.

However, since there are significant differences, it can be concluded that the number of attacks that a honeypot will detect is influenced by the amplification factor.

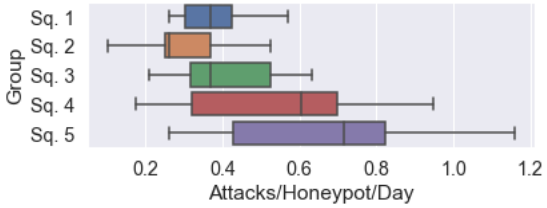


Figure 6.2.12 - DNS Attacks per Honeypot per day

Overview

This subsection has expanded upon the findings of the previous subsection. For RIP and CHARGEN, there is no conclusive evidence that the amplification factor of the honeypots influences the number of attacks it is used in. For the other services, there is a significant ordering between the groups. For the services NTP, SSDP, and QOTD there appears to be a tipping point, whereafter the rate of attacks does not further increase. Table 6.2.2 shows an overview of these three services, their tipping point, and what their lowest and highest bandwidth amplification factor (BAF) and network bandwidth amplification factor (NBAF) would be. Although the difference between the BAFs of the different services is enormous, the difference between the NBAFs is much smaller. This could mean that the established theory about the adversarial strategies on the amplification factor is correct: adversaries take into account the overhead of the network to determine whether a server is amplifying enough data. This is not surprising because this is the effective amplification of network traffic that occurs. The results further support the acceptance of *H1* as there are significant differences between groups that had different amplification factors.

Table 6.2.2 - Tipping point for honeypot selection in attacks.

Service	Tipping Point	Request Size	BAF (low/high)	NBAF (low/high)
QOTD	90 - 200 bytes	1 byte	90.0/200.0	3.1 / 5.6
SSDP	272 - 430 bytes	117 bytes	2.3 / 3.7	2.0 / 3.0
NTP	200-400 bytes	8 bytes	25.0 / 50.0	4.8 / 8.8

6.2.4. Honeypot Selection Preference

In the previous two subsections, we showed that for NTP, SSDP, DNS, and QOTD, honeypots that provide a higher amplification are used more in attacks. For RIP and CHARGEN, no such evidence could be found. The implication of the differences between the higher and lower amplification honeypots is that some honeypots are used less for attacks than others because otherwise they would have been used just as much in attacks. This could be because an attack uses fewer servers of one type than another, or because some attacks use all servers, whereas other attacks do not use low amplification honeypots. This subsection further analyzes the number of low and high amplification honeypots that are used in attacks. The preference towards one type of honeypot or another is further referred to as honeypot preference or preference in short.

First, the method of calculating the preference is discussed, and the results are discussed for each of these services. Thereafter, a deeper analysis follows comparing the different groups of honeypots that were selected during attacks. Finally, a short conclusion is presented.

Preference Calculation

The honeypot preference is calculated separately for each of the six services. The idea of the preference analysis is to have a better indication of how many attacks preferred higher amplification honeypots over the lower amplification honeypots. In order to make this estimation, the preference of each individual target is calculated. For each of the unique targets, all honeypots that were running the main experiment configuration are considered. For each target, the share of high amplification honeypots that were used is determined, which results in a percentage between 0 to 100%. For example, if

one target was attacked by using 25 low amplification honeypots, and 75 high amplification honeypots, the preference of that attack is 75%, meaning there is a preference towards high amplification honeypots. Similarly, if the share is lower than 50%, this would mean that there is a preference towards low amplification honeypots - because more low than high amplification honeypots were used in the attack. With this method, each unique target contributes to a single data point.

Figure 6.2.13 shows a boxplot of all these data points, for each service. For both CHARGEN and RIP, there is no evidence that there is a preference for high amplification honeypots. For DNS, there is a preference visible - but only minimal. This is also in line with earlier results that showed minimal differences in the daily attack rates with DNS. For QOTD and SSDP, there is also a preference visible, but it is stronger than with DNS. Again, this is in line with the previous analysis that showed that the higher amplification groups for these services attract significantly more attacks. NTP showed the highest number of attacks with the higher amplification honeypots compared to the other services, and the preference in this figure also confirms that: there are many attacks with a very strong preference towards high amplification servers only.

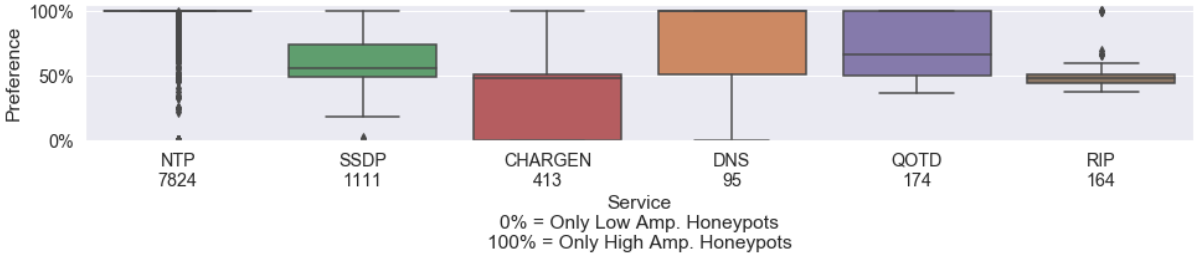


Figure 6.2.13 - Preference for high or low amplification honeypots

Each of the boxplots shows the preference for high amplification honeypots of adversaries that use honeypots of the given service. The number below the service denotes the number of data points in the shows distributions. A value of 50% means that there is no preference for either lower or higher amplification honeypots as they are used both as much in an attack.

Honeypot Selection

Analysis of the main experiment and the squeeze experiment showed that low amplification honeypots attract fewer attacks than high amplification honeypots. The preference calculation showed that the differences can be explained by analyzing the preference towards higher/lower amplification honeypots in attacks. The question that remains is whether this preference for high or low amplification honeypots is because all adversaries have a certain ratio of low and high amplification honeypots that they use in attacks, or that there are different adversaries that use different selection strategies. For example, one adversary could select all honeypots to use in his attack, whereas another could only select high amplification honeypots. Therefore, this subsection further analyzes the attacks to answer that question.

In section 4.3 we explained that an attack consists of flows from individual honeypots, and these individual honeypots had different settings applied to them. Multiple honeypots with the same settings belonged to the same group. Thus, an attack could use 50 honeypots of Group A and 50 honeypots of Group B. In that case, there would be no preference for either group. However, if an attack used 10 honeypots of Group A and 50 honeypots of Group B, there would be a strong preference towards honeypots of Group B. During the analysis of honeypot selection, these preferences are plotted in a scatterplot that allows the number of honeypots that are used in one group compared to another group.

Before analyzing the honeypot selection preference of the six services, one example analysis is given. Figure 6.2.14 shows attacks that were detected with the honeypots in groups A and B, and how many honeypots from one group were used compared to another group. For example, this plot shows on the Y-axis the number of honeypots in group A for an attack and the number of honeypots in group B on the X-axis. Each dot represents a victim and shows the number of honeypots that were in attacks towards that victim from honeypots in one group compared to honeypots in another group. Each dot is plotted with a transparency value. When the color is more saturated, this means that there are data points that are overlapping.

The diagonal line shows where there is an equal number of honeypots for the group on the X and Y-axis. Attacks located around this line have no apparent preference towards either one group or another. In this example, there are three observations:

- There are attacks that use honeypots from both groups equally. These attacks thus have no selection preference to low or high amplification honeypots.
- There are also attacks that use a lot of high amplification honeypots, and that only use a few low amplification honeypots. These attacks do appear to have a strong selection preference to high amplification honeypots but do not exclude lower amplification servers.
- There are attacks that only use honeypots from the high amplification group and none from the low amplification group. These attacks have the strongest selection preference towards high amplification honeypots.

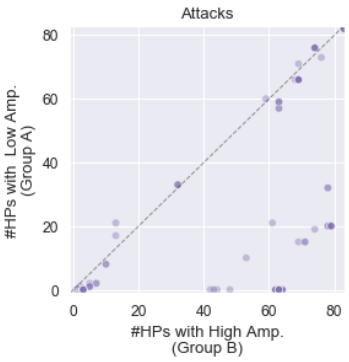
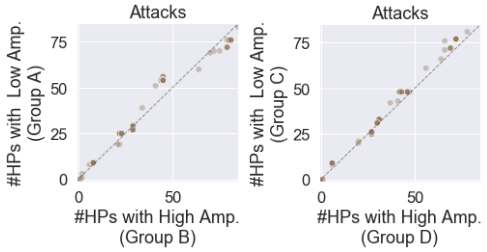


Figure 6.2.14 - Example of honeypot selection of groups A and B

The six different services are now analyzed separately in the same order as the previous analysis. For each service, a similar figure is included as figure 6.2.14 - with details for that specific service.

RIP

For RIP, all attacks are located close to the main diagonal. There is no apparent preference that shows that adversaries only select honeypots of one group over another group. This is in line with the earlier findings that showed that adversaries that use RIP for attacks appear to have no preference for high or lower amplification honeypots.

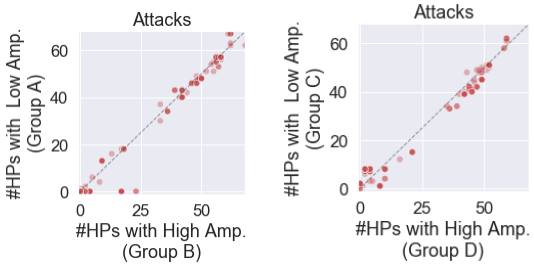


(a) **(b)**

Figure 6.2.15 - Honeypot selection in RIP attacks

CHARGEN

For CHARGEN, many attacks are located close to the main diagonal. There are a few attacks that only used honeypots in Group B over honeypots in Group A, but these cases seem to be limited, and these attacks seem to be only small attacks. In general, there is no apparent preference that shows that adversaries only select honeypots of one group over another group. This is in line with the earlier findings that showed that adversaries that use CHARGEN for attacks appear to have no to only a very minimal preference for high or lower amplification honeypots.

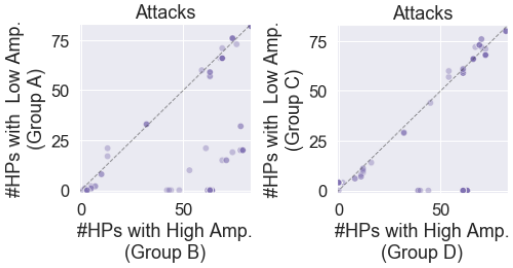


(a) **(b)**

Figure 6.2.16 - Honeypot selection in CHARGEN attacks

QOTD

QOTD shows different results than both RIP and CHARGEN. Although there are attacks that are located on the diagonal - attacks that show no preference - there are many attacks that do show a preference. Figure 6.4.17.a shows that there is a group of attacks that do show a strong bias to honeypots in group B compared to group A. Similarly, figure 6.4.17.b shows that there are attacks where no honeypots of group C are used, but many honeypots of group D are used. Another interesting observation is that the attacks that do appear to have a bias, also use a lot of honeypots in the attacks. This will be further analyzed in the next subsection.

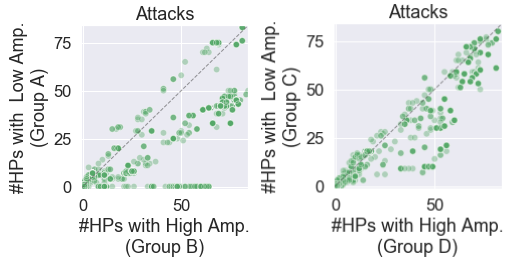


(a) **(b)**

Figure 6.4.17 - Honeypot selection in QOTD attacks

SSDP

SSDP shows a much more varied use of the honeypots in the different groups. In figure 6.4.18.a, three different clusters can be detected: a cluster that shows no, or only a little preference. A secondary cluster, that uses more honeypots in group B (higher amplification) than honeypots in group A (lower amplification), and finally, a third cluster that uses no honeypots in group A. Figure 6.4.18.b shows similar clusters, only there seem to be no attacks that used honeypots in group D and no honeypots in group C. This is odd because both groups A and C provided a lower amplification than groups B and D. A possible explanation for this is discussed in section 6.5, fake services.

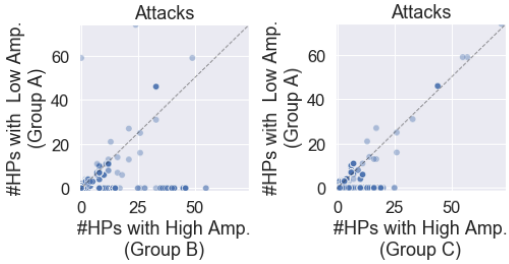


(a) (b)

Figure 6.2.18 - Honeypot selection in SSDP attacks

NTP

Figure 6.4.19.a shows the honeypot selection of group A (small amplification) compared to group B (vulnerable NTPD server). Figure 6.4.19.b shows group A compared to group C (static, large amplification). In both cases, there are attacks that show no preference for one or the other group. However, the figures also show a cluster on the bottom of the X-axis. These clusters indicate that there are many attacks that chose to use honeypots in group B and C (that provide a high amplification) while specifically not using the lower amplification honeypots from group A. Interestingly, of the attacks that only used honeypots in group B and C, the attacks in group B seem to have been larger - a share of attacks used more different honeypots than in group C.

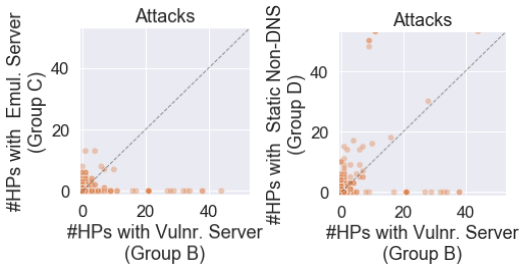


(a) (b)

Figure 6.2.19 - Honeypot selection in NTP attacks

DNS

With DNS, both groups A and C provided a lower amplification than group B and D. Figure 6.4.20 a and b show that there is a large cluster of attacks on the bottom of the Y-axis. These attacks only used honeypots from the higher amplification groups B and D.



(a) (b)

Figure 6.2.20 - Honeypot selection in DNS attacks

The Squeeze Experiment

Similar to the analysis above for the main experiment, the honeypot selection is also analyzed for the squeeze experiment. However, the results are very similar and the number of data points is much smaller. For this reason, the full analysis for the squeeze experiment is included in Appendix E. The summary of that analysis is discussed here.

For RIP, there were victims that were targeted with attacks that excluded the smallest squeeze group. The results of CHARGEN are similar to that of the main experiment and show no differences between the different squeeze groups. QOTD also shows similar results: there are victims that were attacked without honeypot selection preference, and victims that were attacked by only using high amplification honeypots. SSDP is also similar and the squeeze experiment shows three similar clusters of data points. Finally, NTP and DNS also both show similar honeypots selection preferences compared to the analysis of honeypot preference of the main experiment.

Honeypot Selection: Overview

In summary, varying from service to service, there are different observations of honeypots selection methods that appear to have been used:

1. No selection preference: any available honeypot is used regardless of the amplification factor that it provides. These attacks do not have any apparent selection preference towards a type of honeypot.
2. A slight selection preference: mostly high amplification honeypots are used, but lower amplification honeypots are also still used. This especially seems to be present with SSDP attacks, and to some extent, also with QOTD attacks.
3. A very strong selection preference: only the high(est) amplification honeypots are used for attacks. This preference seems to be present with four the six services: SSDP, QOTD, NTP, and to some extent, with DNS. Both RIP and CHARGEN see no selection strategy at all: both types of honeypots are used equally much.

These results indicate may indicate that there are different types of adversaries at work: more sophisticated attackers may be selecting only the highest amplification honeypots for their attacks, whereas less sophisticated attackers may select any honeypot.

6.2.5. Number of Honeypots used in Attack in different Groups

The previous subsection showed that there are three different honeypot selection strategies. However, the analysis focused on the bigger picture and did not take into account how many attacks used the given number of honeypots for attacks. The data points could be concentrated in a few different spots. Therefore, this subsection will analyze the number of honeypots that are used in attacks amongst the different groups of the main experiment. The goal of this analysis is to determine whether the groups that provided a higher amplification are also detecting significantly larger attacks than the groups that provided a lower amplification factor.

This subsection follows the same method of analysis as was previously used to compare the influence of the amplification factor for the main and squeeze experiments. Instead of comparing the distributions of the average daily attack rate, the distributions of the number of honeypots of attacks are compared. More specifically, the lower amplification groups are compared to the higher amplification groups. The different services are analyzed separately. The included figures for the services show a boxplot of the distribution of the number of honeypots for the different groups of that service. The number next to each group label indicates the number of data points that are used for analysis. For similar reasons as in the aforementioned analyses, the Mann-Whitney U-test is used to test for significant differences between the distributions. If there are no significant differences between groups, this means that there are no significant differences that meet the chosen alpha level 0.05.

RIP

Figure 6.2.21 shows the number of honeypots that were used in attacks using RIP. For RIP, the average number of used honeypots is high. There appear to be two groups that are popular: attacks that use around 40 honeypots, and attacks that use around 70 honeypots. Although attacks using group A honeypots appear to have a slight difference, this is not a significant difference. In the squeeze experiment, all attacks that used group 1, used exactly 10 honeypots, but that were only 2 attacks - so this provides not enough data points to build an accurate conclusion.

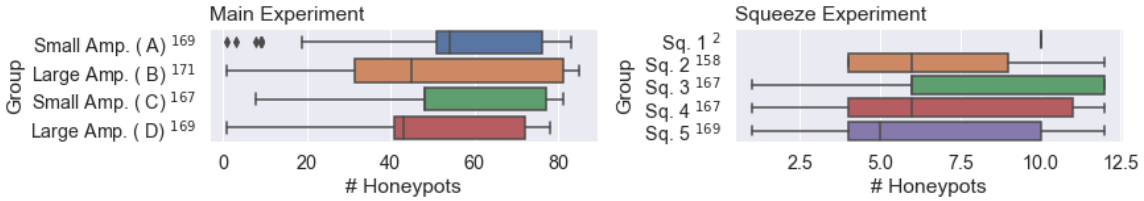


Figure 6.2.21 - Distributions of the number of honeypots used in RIP attacks for different groups

CHARGEN

Figure 6.2.22 shows the number of honeypots that were used in attacks using CHARGEN. For all the groups in the main experiment, there either seems to be used a small number of honeypots or a large number of honeypots that are used. The same applies to the groups in the squeeze experiment. However, as the figure shows, there is again a large overlap of the different groups. There are no significant differences.

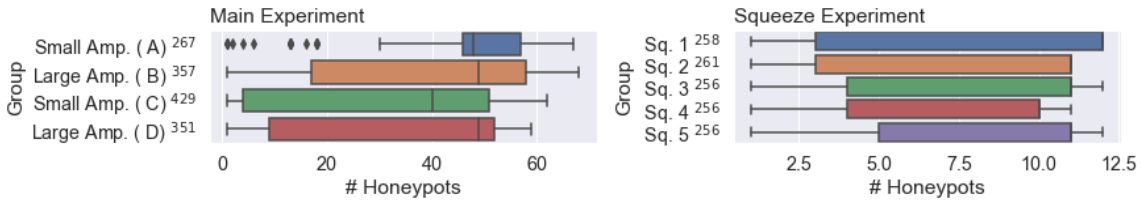


Figure 6.2.22 - Distributions of the number of honeypots used in CHARGEN attacks for different groups

QOTD

Figure 6.2.23 shows the number of honeypots that were used in attacks using QOTD. For QOTD, there is a significant ($p<0.01$) difference in the number of honeypots that are used in attacks between group A and B, and between C and D. Thus it appears that the higher amplification groups attract attacks that are generally larger than the groups with a lower amplification factor. For the squeeze groups, both a large share of attack in the group 4 and 5 appear to use around 8 honeypots in their attacks, whereas a large share of attacks that use group 3 use 9 different honeypots. Similar to RIP and CHARGEN, the number of honeypots that are used in attacks are is generally a large share of the honeypots that were available.

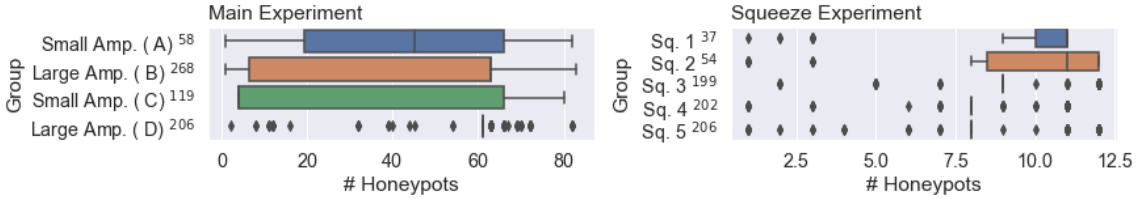


Figure 6.2.23 - Distributions of the number of honeypots used in QOTD attacks for different groups

SSDP

Figure 6.2.24 shows the number of honeypots that were used in attacks using SSDP. SSDP shows a variety of number of honeypots that are used in attacks. For all the groups, a large share of attacks uses only a small number of available honeypots or a very large number of honeypots for their attacks.

There is a significant ($p < 0.01$) difference between attacks that used group A compared to group B, and group C compared to group D. The difference between the groups shows that the higher amplification honeypots appear to have attracted larger attacks overall compared to the smaller amplification honeypots. This is also the case for the squeeze groups, where the number of honeypots that were used in attacks in groups 1, 2, and 3 was significantly ($p < 0.01$) less than in the higher amplification groups 4 and 5.

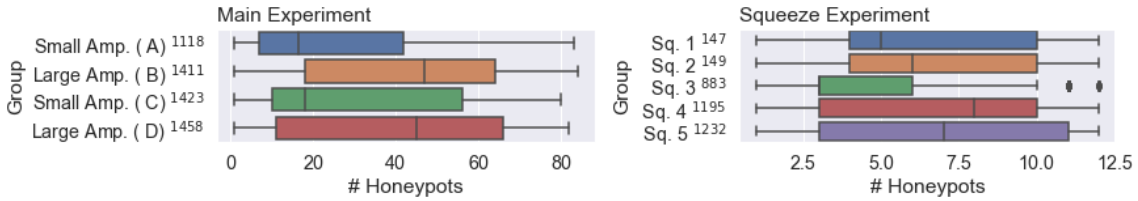


Figure 6.2.24 - Distributions of the number of honeypots used in SSDP attacks for different groups

NTP

Figure 6.2.25 shows the number of honeypots that were used in attacks using NTP. NTP shows a significant ($p<0.01$) difference between the lower amplification group A, and the higher amplification groups B and C, although group B elicits the average highest number of honeypots that are used in attacks.

It is notable that the number of honeypots that are used in attacks is much lower compared to QOTD, SSDP, CHARGEN, and RIP. The squeeze group also shows several differences: squeeze groups 1, 2, and 3 use a significantly ($p < 0.05$) lower number of honeypots in attacks than attacks that used honeypots in groups 4 and 5. Thus, overall it appears that for NTP, attacks that use the larger amplification honeypots also use more honeypots in NTP attacks.

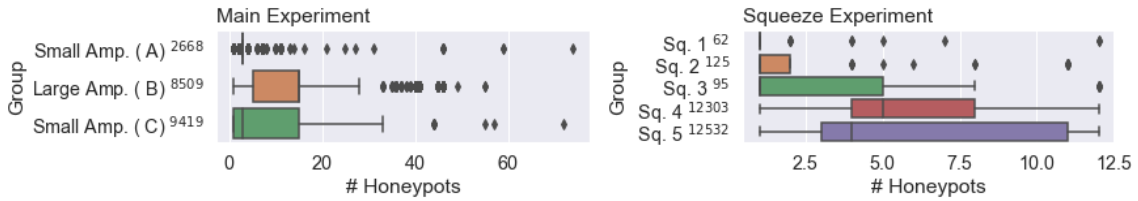


Figure 6.2.25 - Distributions of the number of honeypots used in NTP attacks for different groups

DNS

Figure 6.2.26 shows the number of honeypots that were used in attacks using DNS. The results for DNS show that the majority of attacks use only a small number of honeypots for their attacks. This observation is present in both the groups form the main experiment and also for the squeeze experiment. Furthermore, there is a significant difference ($p < 0.01$) between the lower amplification group C and the higher amplification group D. The higher amplification group elicits larger attacks overall.

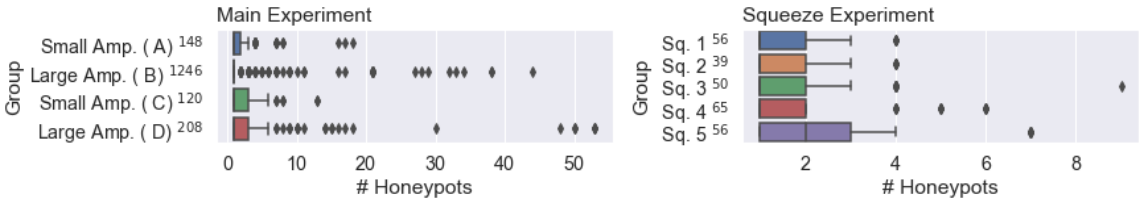


Figure 6.2.26 - Distributions of the number of honeypots used in DNS attacks for different groups

Overview

For the services SSDP, NTP, and QOTD, there are significant differences that show that higher amplification servers also attract larger attacks in terms of the vulnerable servers that are used in an attack. For CHARGEN, RIP, and DNS, there is no evidence that this is also the case. Further observations include that for RIP and CHARGEN - although they show hardly any preference - the number of honeypots that are used in their attacks is generally higher than SSDP, NTP, and DNS. This is also the case for the deprecated protocol QOTD. One explanation is that the amount of available vulnerable servers for these deprecated and legacy protocols is lower, thus a larger share of the available honeypots needs to be used in order to generate an attack of a certain size.

6.2.6. Packets

The received packets for each protocol can be analyzed to gather more information on the purpose of and tools that adversaries have used during the attacks. Specifically of interest is if different packets are primarily detected among the different honeypot types. For example, did the higher amplification honeypots also detected different packets than lower amplification honeypots? And if that is the case, does this provide an insight into the choices that adversaries made while selecting the honeypots?

Some of the services saw an enormous amount of different packets that were used - whereof many were only used once. Therefore, this analysis focuses only on the most relevant observations. Appendix F contains tables and statistics on the most popular packets that were detected among the honeypots.

These tables contain information on the number of attacks of the different detected packets, the share of multi-vector and subnet attacks, and the share among the different groups wherein they were detected. Multi-vector and subnet attacks[26] could be an indicator of a more targeted or sophisticated attack. During this analysis, strings, or parts thereof, that contain non-printable characters are represented in a hexadecimal format, recognizable by the prefix 0x.

RIP

There were 6 different unique packets received by the RIP honeypots. Although RIP replied with a response to any incoming packet, there was a single request packet that was used in more than 98% of the attacks. This packet was also detected in 91% of the multi-vector attacks that used RIP. This packet is the only proper packet that - according to the specification - should trigger the vulnerable response. However, the RIP emulator was able to respond to any incoming packet. This suggests that even though any request could trigger the response, the adversaries that used the RIP servers in attacks, were aware of the fact that it was a RIP server, and with that, were probably looking specifically for these vulnerable servers.

[26] the glossary explains what a subnet and multi-vector are. These special types of attacks are further analyzed in section 6.7.

This is an interesting observation because previous analyses showed that adversaries that use RIP in attacks show no preference towards high amplification servers. There are multiple possible explanations: it could imply that more advanced adversaries do not use a higher-amplification selection strategy but were merely looking for these old legacy servers to use. Or, RIP is used by less sophisticated adversaries that still try to use these old vulnerable servers in attacks, and they were specifically looking for it, while a more sophisticated adversary perhaps just tries to get a response with a more efficient request.

CHARGEN

There were 12 different unique packets received by the CHARGEN honeypots. However, only 2 different packets were used in a large share of attacks. The 0x01 packet is used most frequently and shows no clear preference, it is used among all the groups, and specifically, it is used in more than 99% of the attacks in the squeeze experiment. It is also used in 45.9% CHARGEN multi-vector attacks and in 13.6% of CHARGEN subnet attacks.

The second most popularly used packet, the packet with 0x30 as the content is not used in attacks that used group A. It was also not used in any attacks of the squeeze experiment. It was used in 62.4% of the CHARGEN multi-vector attacks and in all of the CHARGEN subnet attacks. This is a very odd selection of types of honeypots that were used. All the squeeze groups had a static response, but groups C and D also had such a static response. On the other hand, especially groups 4 and 5 also provided an amplification factor that is at least as high as groups B and D. In summary, there is no logical explanation that explains these observations.

QOTD

There were 25 different unique packets received by the QOTD honeypots. Only 5 different packets were used in more than 5 attacks. The most notable observations are discussed. The most frequently used packet was 0x01. With a length of only 1 byte, it is the shortest packet to trigger a response. This packet was not used in the lower amplification group A, or in the lower amplification groups 1 or 2. It has been used in group C, although less than in the higher amplification groups B, D, 3, 4, and 5. The packet was also used in 28.6% of all QOTD multi-vector attacks.

The second most frequently used packet was bigbo. It was used among all the groups evenly in 30 different attacks. Using this packet for attacks also results in a lower amplification factor than the most frequently used packet. The packet was not used for multi-vector or subnet attacks. With an extensive search attempt, there are no references to be found to any usage of this specific packet. In summary, it appears that the adversary that executed attacks using this packet had no preference for high amplification honeypots and perhaps was also not aware that the vulnerable server he was using was a QOTD server.

The third packet was used in 24 attacks, and used all of the different groups in attacks, except for the lowest amplification group 1. The content of the packet was getstatus0x01. The packet was also used in multi-vector attacks. There are no known references to the contents of the packet. The content closely resembles the trigger for vulnerable Quake3- engine servers, but it misses a prefix.

The fourth most frequently used packet is also of size 1, as was the first packet. This packet was only present in attacks that used the higher amplification groups B, D, and group 4. It was also used in 14.3% of the QOTD multi-vector attacks.

In summary, the two packets that provide the highest amplification factor to an adversary, are the packets that are mostly used with attacks that had a selection preference towards high amplification honeypots. This implies that adversaries that used high amplification honeypots crafted packets to specifically use vulnerable QOTD servers. This suggests that there are different adversaries with different strategies: those that verify that the server is a QOTD server to optimize their attacks, and adversaries that use a honeypot whenever a response is sent back.

SSDP

There were 70 different unique packets that were used in SSDP attacks. The three most frequently used packets are all similar in content, but the honeypots where they were detected differ. The IPs that can be found in the content of the packets are merely standard multicast IP-addresses and provide no further insight. Finding these IPs in the packets is not unexpected.

The first packet was detected in 98.2% of multi-vector attacks and in all the subnet attacks. It was detected in more than 90% of all the groups, except in the lower amplification groups 1 and 2 In groups 1 and 2 the packet was only detected in less than 70% of the attacks. Thus, for this packet, there appears to be a slight preference for higher amplification honeypots. The content of the first packet is the following:

M-SEARCH * HTTP/1.1
Host:239.255.255.250:1900
ST:ssdp:all
Man:"ssdp:discover"
MX:3

The packet that was detected the most thereafter shows less clear distinctions between the groups. Notable is that the share in group A is much lower than in group B, and also that group C is lower than group D. However, the share in the lowest squeeze amplification groups (1-2) is much higher than in the higher squeeze amplification groups 3-5. This suggests that the difference is not due to the amplification. The packet is as the following:
M-SEARCH * HTTP/1.1
Host:239.255.255.250:1900
ST:upnp:rootdevice
Man:"ssdp:discover"
M

The third most frequently used packet is similar to the one above but ends with MX: 3 instead of M, and has two additional newlines. The packet is used mostly in the lower amplification groups. One explanation could be that it belongs to the same adversary as the previous packet, but that a part of the DDoS generated content is being cut off. The length is exactly 90, whereas the length of the full packet is 99.

The only packet in the top 10 most frequently used packets for SSDP that does not adhere to the specification, is the fourth packet. The content of that packet is disconnect. The share among the groups for this packet is similar to the previous packet. The packet could be generated by the generator library that an adversary uses. Upon completion of a stream of packets, disconnect is then transmitted, ending the attack.

In summary, the most frequently used SSDP packets appear to have a slight preference towards higher amplification honeypots. Furthermore, the high number of similar packets indicates that there are many different generators for the traffic. This suggests that there are many different adversaries at work. Furthermore, many of the detected packets adhere to the SSDP specification, suggesting that adversaries that executed these attacks were aware that the servers were SSDP servers. However, they did not use a more efficient packet to execute an attack, although that was a possibility.

NTP

There were 11867 different unique packets received by the NTP honeypots. Only 11 packets were used in more than 5 attacks. One packet, in particular, was used most frequently and was detected in more than 95% of all attacks. This packet is precisely the 'magic' packet - the packet that triggers a vulnerable server to send the largest MONLIST response. This packet shows a preference for the highest amplification groups. It was used in 99.9% of all attacks that used the vulnerable NTPD servers. It was also used in 97.7% of the attacks using group C, which also provided a high amplification factor. It was used less in group A that provided a lower amplification. The packet was also used in 99.8% of attacks using squeeze groups 4 and 5, but much less in the squeeze groups 1, 2, and 3. Thus, a share of adversaries that specifically used the 'magic' packet appear to have been aware of the NTP server and were specifically looking for the servers that provided a higher amplification response to amplify their traffic.

Other packets were used in fewer attacks than the first packet. Although all of the packets are similar in length and content, they differ from the magic packet, meaning that they elicit no response of honeypots in groups A, B, C and D. Therefore, it is odd that they are still used. The second most frequently used packet was used to generate a lot of traffic in the lower amplification squeeze groups 1-3. Although providing a smaller amplification than the other squeeze groups, these groups still provided a small amplification.

DNS

There were 4919 different unique packets received by the DNS honeypots. Contrary to NTP, there were not only a few packets that were used in almost all attacks. The most frequently used packet has a share of 8,5% and is a DNS packet requesting the version of BIND (the DNS server). This packet was detected the most at the server that responded with random data upon receiving a request.

The most notable observation of all the different packets that are used for DNS, is that the different packets are specifically detected only in certain groups. For example, some packets are mostly detected in attacks targeting the open DNS resolver, whereas others see a much larger share in a static emulated response. The packets that were detected among the squeeze groups - that responded to any incoming query - see a much more equal share than the four groups of the main experiment. This suggests that adversaries were trying to get responses to specific queries, and if a response was provided, they would use the honeypots in their attacks. The higher number of different packets that were used can be explained because there are many different queries that provide an amplified response.

Overview

The packet analysis showed that there is a variation in the number of different packets that were used for each of the services. For all the services, except for DNS, only 3-5 packets are used in more than 99% of the attacks. For RIP, CHARGEN, QOTD, and NTP, the most optimal packet was used in the majority of attacks. For SSDP, it was interesting that almost all of the most frequently used packets were packets crafted according to the SSDP specification, while a random packet with a length of 1 could also have triggered a response.

For NTP, QOTD, and SSDP, the analysis shows that when a packet is used that is optimal to elicit a response, there is also a clear preference towards higher amplification honeypots. Overall, it appears that there are different adversaries that use different packet generators in attacks, and whereas some seem to be crafted for the specific service, other packets indicate that some adversaries are not aware that the vulnerable server is of a specific service.

6.2.7. Conclusion H1

Hypothesis *H1* stated that there will be a difference in the abuse frequency between the honeypots that provide a higher amplification factor than their low amplification factor counterparts. For RIP and CHARGEN, no evidence was found that proved this statement to be correct. However, for QOTD, SSDP, NTP, and DNS, there are significant differences that show that the honeypots that provided a higher amplification factor are used more than the honeypots that provided a low amplification factor.

Further analysis has shown that there are certain values for the amplification factor, thereafter increasing the amplification factor does not further increase the number of attacks that these honeypots detect. These boundaries, or tipping points, were found for QOTD, SSDP, and NTP, where the boundary appeared to be with at least an NBAF of 2.0.

Furthermore, we found that there are different honeypot selection strategies that adversaries use to choose which honeypots they use in their attacks. These included using both high and low amplification honeypots equally as much, selecting more high amplification honeypots, and finally, solely use high amplification honeypots for attacks.

For SSDP, NTP and QOTD, the attacks that used the high amplification honeypots also generally had more honeypots that were used in the attack, than attacks that used the low amplification honeypots. For these same protocols, it appeared that packets that in theory would elicit the highest response from the vulnerable servers were primarily used in the attacks that had a strong preference for high amplification honeypots.

In conclusion, there appear to be different types of adversaries that use different honeypot selection strategies. The results have indicated that the amplification factor is of influence on this selection strategy. Higher amplification honeypots are generally preferred. In summary, this means that hypothesis *H1* is accepted.

6.3. H2 - Network Quality Factors

Hypotheses *H2.a*, *H2.b*, and *H2.c* describe three network connection quality properties that may be of influence on adversarial behavior. Each of the three hypotheses states a property that may be of influence on the number of attacks that a service running with that property will see. For each of these hypotheses, a corresponding experiment was constructed. These three experiments were grouped under the name 'network quality factors' and include the experiments to test the influence of rate-limiting, packet loss, and packet delay on the abuse frequency by adversaries.

The three network quality factor experiments ran with the same four different groups and six services as the main experiment. This allows for an analysis that compares each of the groups in the experiments to the same group in the main experiment. There are three possible outcomes of this comparison:

1. There is no (significant) difference between any of the experiments and the main experiment group. This means that there is no evidence that the factor has any influence on the behavior of the adversaries.
2. There is a significant difference among all groups of a service, in this case, the factor is of influence.
3. There is inconclusive evidence if the different groups show different behavior.

A similar approach as the amplification factor analysis is used, as described in section 6.2.2. The groups A, B, C, and D of each of the experiments are compared to the same four groups in the main experiment. The primary analysis compares the distributions of the average daily number of attacks between the corresponding groups. The Mann-Whitney U Test is used to test for a significant difference between the distributions. The next three subsections discuss these three experiments in the aforementioned order.

6.3.1. H2.a - Rate-limit

Hypothesis *H2.a* states that there will be a difference in the abuse frequency between the honeypots that provide a fast network connection than honeypots with a slow connection. This hypothesis is tested by using different rate-limits that are imposed on the honeypots. However, there is an external variable that has influenced the experiment. Figure 6.3.1. shows a schematic overview of the experiment and the external influence. To do accurate analysis, the control group should not have a rate-limit applied. Then, the test group has a form of rate-limiting, and differences between the groups can be analyzed. For the test group, only attacks in the test group that were actually rate-limited should be included, otherwise, an adversary could not have known that a rate-limit was applied.

However, due to the ethical considerations, a global rate-limit was also applied. This global rate-limit was applicable both to the test and the control group, and thus both groups were actually rate-limited. In the control group, 67% of the flows were rate-limited. In the rate-limit group, 54% of the flows were rate-limited. Thus, selecting only the data from the control group that was not rate-limited and selecting data from the rate-limit group that was rate-limited, will create a data selection bias, possibly leading to incorrect conclusions. Thus, the rate-limit experiment leads to no conclusions, and hypothesis H2.a that states the speed of a network connection is of influence to the abuse by adversaries will remain unanswered.

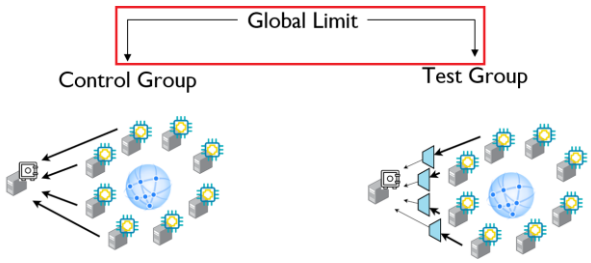


Figure 6.3.1 - Control and test group in the rate-limit experiment

6.3.2. H2.b - Packet Loss

Hypothesis *H2.b* states that there will be a difference in the abuse frequency between the honeypots that have no packet loss than honeypots with a high(er) amount of packet loss. Thus, we try to disprove the null hypothesis that states that there is no significant difference in the abuse frequency between the honeypots that have different amounts of packet loss. This hypothesis is tested by using three different packet loss settings that are imposed on the honeypots. These settings were 33% packet loss, 50% packet loss, and 66% packet loss.

Because of the limited number of data points, the data points of the 50% and 66% packet loss group are grouped together under the name "≥ 50% Packet Loss". Similar to previous analysis, the distributions of the average daily attack rates are compared. The analysis is done for each of the services and is discussed separately.

The analysis of each of the services contains a figure that shows a boxplot of the distributions that are compared from the different groups. A table is also included that shows the 4 different groups and the 3 different settings. If there is a significant difference in the distributions of the daily attack rate, the median of the distribution is noted in the table, and the table cell is colored green. If there was no significant difference, a dash "-" is noted in the table for that observation and the background of the table cell has a red color. Finally, if 33% packet loss shows that there are fewer attacks per honeypot but ≥ 50% packet loss does not, this suggests that the differences are not related to packet loss itself. Therefore, if this is the case, the results are also considered to be insignificant. If hypothesis *H2.b* is correct, it is expected that the distributions of higher packet loss settings are centered around lower values than of the honeypots that ran without any packet loss.

RIP

For RIP, only the groups A and B show a significant difference where the number of daily attacks is significantly ($p<0.05$) lower when the packet loss is not zero. In group A, the number of attacks per honeypot per day was lower for the honeypots with 33% packet loss than for the honeypots with ≥ 50% packet loss. In group B this was not the case, but a similar result is visible in group D. Since groups C and D did not provide a real RIP response, the packet loss may only be of influence for actual RIP servers.

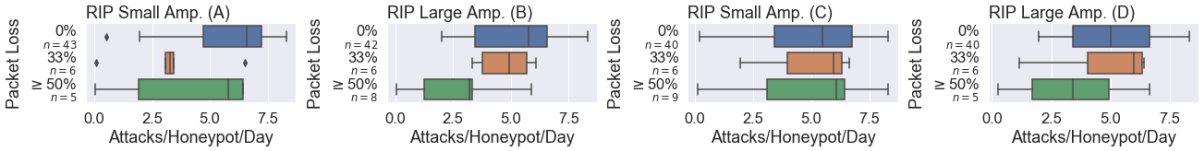


Figure 6.3.2 - Average attack/day/honeypot for RIP among the different groups using different packet loss settings

Table 6.3.1 - RIP significant (p<0.05) differences between main groups and rate-limit groups - median values

The green cells denote significant (p<0.05) differences. The red cells denote groups where there were no significant differences between groups.

	Group A (Small Amp.)	Group B (Large Amp.)	Group C (Small Amp.)	Group D (Large Amp.)
0%	6.58	5.79		
33%	3.26			
≥ 50%	5.81	3.21		

CHARGEN

Figure 6.3.2 shows the distribution of the data points among the different groups. With only 33% packet loss, there is no significant difference. However, with ≥ 50% packet loss the distribution of data points appears to be centered further to the left than the data points from the main experiment. The median values that table 6.5.2 shows for ≥50% packet loss are also much lower. The difference in group D is not large enough to be significant, but for the other three groups, there is a significant ($p < 0.05$) difference. Overall, it appears that a high amount of packet loss influences how often the CHARGEN honeypots are used in attacks.

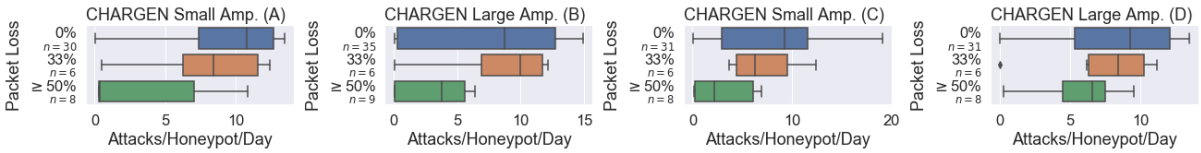


Figure 6.3.2 - Average attack/day/honeypot for CHARGEN among the different groups using different packet loss settings

Table 6.3.2 - CHARGEN significant (p<0.05) differences between main groups and rate-limit groups - median values

	Group A (Small Amp.)	Group B (Large Amp.)	Group C (Small Amp.)	Group D (Large Amp.)
0%	10.79	8.71	9.26	
33%				
≥ 50%	0.37	3.79	2.21	

QOTD

Similar to CHARGEN, figure 6.3.4 shows that for QOTD the distributions with a high packet loss appear to be more centered to the left the distributions without packet loss. The results in group A are not significant. However, groups B, C, and D show a significant ($p < 0.05$) difference where the attacks per honeypot per day are lower with more packet loss. However, the differences vary between the groups. The largest differences are in the groups B and D, which are the highest amplification groups for QOTD. This may indicate that adversaries that are more sophisticated and only selected high amplification honeypots, also do further verification to test the quality of the vulnerable service.

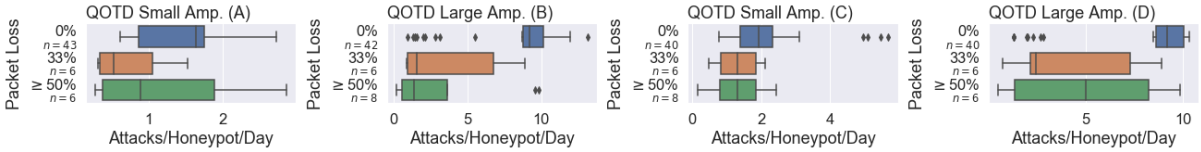


Figure 6.3.4 - Average attack/day/honeypot for QOTD among the different groups using different packet loss settings

Table 6.3.3 - QOTD significant (p<0.05) differences between main groups and rate-limit groups - median values

	Group A (Small Amp.)	Group B (Large Amp.)	Group C (Small Amp.)	Group D (Large Amp.)
0%		9.21	1.93	9.18
33%		1.61	1.31	2.45
≥ 50%		1.42	1.32	5.03

SSDP

SSDP is the only service that has a significantly lower number of attacks for 33% and ≥50% packet loss among all the groups. Figure 6.3.5 also shows that all the distributions of higher packet loss settings are centered further left than the distributions with less packet loss. Furthermore, the median number of daily attacks per honeypot is lower for 33% packet loss and even lower for ≥ 50% packet loss**.** Overall, it appears that for SSDP, the packet loss is of influence on the abuse by adversaries, and that more packet loss has a stronger influence.

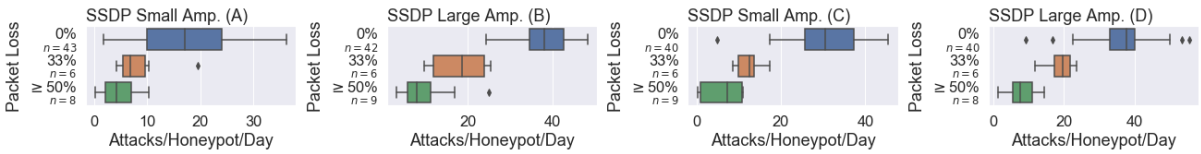


Figure 6.3.5 - Average attack/day/honeypot for SSDP among the different groups using different packet loss settings

Table 6.3.4 - SSDP significant (p<0.05) differences between main groups and rate-limit groups - median values

	Group A (Small Amp.)	Group B (Large Amp.)	Group C (Small Amp.)	Group D (Large Amp.)
0%	17.16	38.06	30.49	37.68
33%	6.87	18.63	12.48	19.60
≥ 50%	4.11	7.90	7.26	7.61

NTP

NTP had three different groups that attracted attacks. Group A provided the lowest amplification, but with this lowest amplification group, more than ≥50% packet loss received several factors fewer attacks on average. Furthermore, it is interesting that group B does see a large decline in the average number of daily attacks with a lot of packet loss, but group C does not. Both groups provided a large amplification factor. However, group B provided the highest (potential) amplification, with it being an actual vulnerable server, compared to group C, that provided a static response. This could mean that adversaries that were looking for real (vulnerable) NTP servers had a clear preference, whereas for a fake NTP server it did not make a difference. These results may strengthen the suspicion that was discussed for SSDP. Adversaries that appear to be well aware of what they are using for attacks, may also be more selective in choosing a server that has less packet loss.

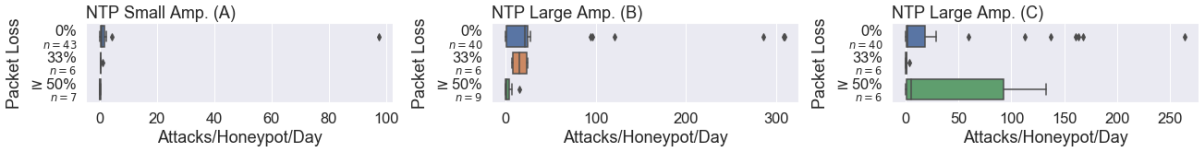


Figure 6.3.6 - Average attack/day/honeypot for NTP among the different groups using different packet loss settings

Table 6.3.5 - NTP significant (p<0.05) differences between main groups and rate-limit groups - median values

	Group A (Small Amp.)	Group B (Large Amp.)	Group B (Large Amp.)
0%	0.71	21.89	
33%			
≥ 50%	0.14	0.81	

DNS

DNS shows a significantly lower attack rate with high packet loss for both the emulated (group C) and static response (group D) DNS service. Group A is a patched DNS server that is not prone to DDoS attacks, but still, it saw a lower rate of attacks with more packet loss. Finally, the actual vulnerable DNS server shows no significant difference. Altogether, these results provide no clear answer to whether packet loss is *actually* of influence for DNS servers, since the only type that is actually a vulnerable DNS server shows no significant difference. Thus, there is no conclusion on how the packet loss is of influence for DNS servers.

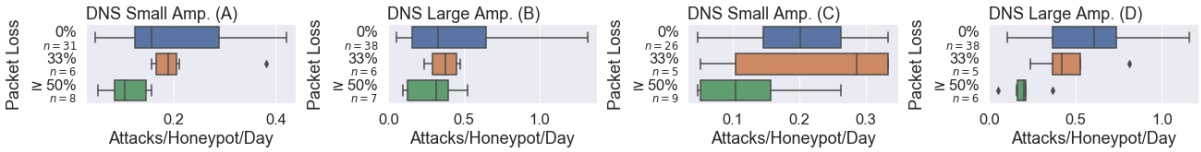


Figure 6.3.7 - Average attack/day/honeypot for DNS among the different groups using different packet loss settings

Table 6.3.6 - DNS significant (p<0.05) differences between main groups and rate-limit groups - median values

	Group A (Small Amp.)	Group B (Large Amp.)	Group C (Small Amp.)	Group D (Large Amp.)
0%	0.16		0.20	0.61
33%				
≥ 50%	0.11		0.11	0.20

Overview

The six different services all show different results. With RIP, only with real servers, the packet loss shows an influence, whereas with CHARGEN also a small fake group shows a difference. For QOTD, the difference seems to be most noticeable in the larger amplification groups, both in the fake and real response groups. NTP shows a difference in the small group, like CHARGEN, but not in the fake large group, like QOTD. NTP shows a very large difference with the only actual vulnerable server, but for DNS there is no difference at all with the real service. Overall it does appear that adversaries that are more aware of the properties of the honeypot they are abusing - because they know these servers provide for a higher amplification factor - are also (partially) influenced by the packet loss that these servers have. This would not be surprising, a higher packet loss means that the effective amplification factor will turn out to be lower because less traffic is sent to the victims.

It strongly appears that there are different adversaries of whom there is a group where the packet loss is of influence on their choice to use a honeypot or not. The null hypothesis should be rejected because there are significant differences with different amounts of packet loss. Thus, the alternative hypothesis *H2.b* is accepted: packet loss is a factor of influence on the abuse frequency by adversaries.

6.3.3. H2.c - Packet Delay

Hypothesis *H2.c* states that there will be a difference in the abuse frequency between the honeypots that have more delay in their response that honeypots that have no such additional delay. Thus, we try to disprove the null hypothesis that states that there is no significant difference in the abuse frequency between the honeypots that have different response times. All regular honeypots were configured to use no additional delay, aside from the delay that is caused by packet processing in the honeypots itself. For the packet delay experiment, additional honeypots were configured to run with a 250 ms delay, whereas others were configured to run with a 500 ms delay.

There is no significant ($p<0.05$) difference between the distributions of the daily average number of attacks for any of the six services. There are also no significant ($p<0.05$) differences between the distributions of the number of honeypots that are used in attacks. Figure 6.2.8 shows an example distribution of the daily attack rate with SSDP. Although there are some fluctuations, none of the distributions are significantly different from the distributions that show no delay. Figure 6.2.9 shows the distribution of the different settings for the number of honeypots that are used in attacks.

These results are not necessarily unexpected. From the viewpoint of an adversary: whether a vulnerable server is slower in responding or not, even if that is the case, the amplification will be just as large. It could still be the case that adversaries use the delay to make an assessment of its quality or assume the honeypot is placed in a less qualitative network, but it appears that they do not specifically select the honeypots that provide the quickest response. Thus, hypothesis *H2.c* is rejected: additional delay is not of influence on the abuse frequency by adversaries.

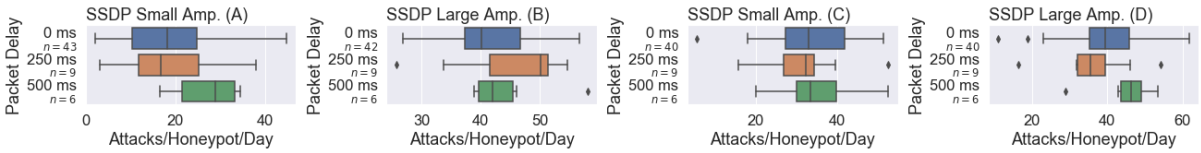


Figure 6.3.8 - Average attack/day/honeypot for SSDP among the different groups using different packet delay settings

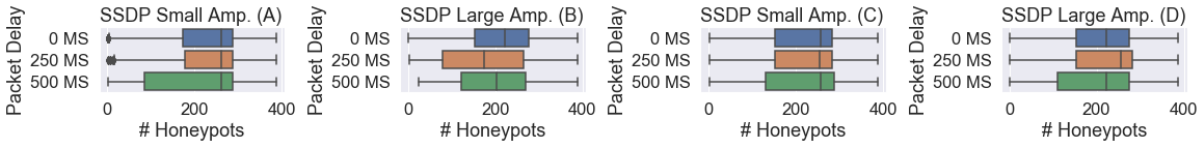


Figure 6.3.9 - Number of honeypots used in different groups using different packet delay settings with SSDP

6.4. H3 - Multi vs. Single Service

Hypothesis *H3* states that systems that run honeypots for multiple services on the same public IP address will see the same number of detected attacks as systems where one public IP address is only used for a single honeypot of a single service. Thus, we try to disprove the null hypothesis that states that there is a significant difference in the abuse frequency between systems that only ran one honeypot on a system and systems where multiple honeypots were running on. The multi vs. single service experiment was constructed to test this hypothesis. During the experiment, almost all VMs that were used ran honeypots for all of the six services. Additional honeypots were used that ran only a single service. Due to resource limitations, more than a third of these honeypots were only able to run until September 15th. Thus, for an accurate comparison, only data gathered before September 15th from both the main experiment and the data gathered for this sub experiment is used for analysis. The honeypots that ran the single service experiment were running on machines from Digital Ocean, Google Cloud, and Amazon AWS. To ensure that the provider and locations do not have an external influence on the results, only data from these providers is used for comparison.

For the services RIP, CHARGEN, NTP, and DNS, there are no significant ($p<0.05$) differences between the distributions of the average number of attacks during these first two weeks of the experiment. For the services QOTD and SSDP, there are several significant differences in the distributions of the average number of attacks. Table 6.4.1 shows an overview of these differences. The groups with no value contained no significant difference. The table shows the median number of attacks per day per honeypot and the number of data points that were available for the analysis (denoted with a #). For SSDP, the differences are in both the real response type groups and for QOTD, the differences are both in the fake response groups.

Table 6.4.1 - Significant (p<0.05) differences between multi and single service groups - medians

Attacks/Honeypot/Day	**Group A *Main***	**Group A *SingleServ.***	**Group B *Main***	**Group B *SingleServ.***	**Group C *Main***	**Group C *SingleServ.***	**Group D *Main***	**Group D *SingleServ.***
QOTD	-	-	-	-	0.95 (#18)	1.51 (#6)	1.02 (#18)	1.31 (#6)
SSDP	1.52 (#21)	5.18 (#6)	7.66 (#20)	8.42 (#6)	-	-	-	-

Unfortunately, the number of data points for this analysis is small, and there is no further indication that strengthens these findings. The timespan of the recorded data is also small, 9 days. Thus, although there are some (significant) differences for some groups for some services, it cannot be assumed that adversaries actually are more careful and refrain from using vulnerable servers because if there are multiple servers running on the same host. Thus, we believe that we cannot conclude on the validity of hypothesis *H3*.

6.5. H4 - Fake Services

Hypothesis *H4.a* and *H4.b* are related to the type of response that the honeypots provided. In summary, these hypotheses state that adversaries are indifferent as to whether a vulnerable service is an emulated service, is an obvious honeypot, or is a real service. While with TCP honeypots, the origin of the traffic is revealed, this is not the case with UDP, because the originating traffic can be spoofed, hiding the identity of the adversary. Thus, even if a vulnerable service is clearly a honeypot, adversaries might as well use it in an attack. The adversary will still not be revealing his identity because of the spoofed origin.

In the first subsection, the influence of the type of response and the influence of an emulated service are analyzed. The second subsection further analyses the behavior of adversaries upon finding vulnerable SSDP services, because the SSDP response provided an URL that could be followed by adversaries to gather more information about the service. The third subsection concludes this section.

6.5.1. Influence of Response Type

For the protocols, CHARGEN, RIP, SSDP, and QOTD, groups A and B provided real responses, and groups C and D provided obvious “fake” responses. As such, group A can be compared with group C and group B can be compared with group D. Both NTP and DNS had actual vulnerable servers running, as well as emulated servers and obvious fake responses. The groups for these services are reiterated during their analysis.

Similar to the previous analysis of the main experiment, the distributions of the average number of daily attacks is compared. The distributions are compared with the Mann-Whitney U Test. The chosen alpha value for the statistical test is 0.05. The figures of the distributions were already included in section 6.2.2. These are implicitly referred to during this analysis. The

order of the services is similar to previous analysis. Thus, figures 6.2.1 to 6.2.6 are used in the analysis from RIP to DNS respectively.

RIP & CHARGEN

For RIP and CHARGEN, there were no significant differences between the lower and higher amplification groups. There is also no significant difference between the groups A and C and the groups B and D. Thus, it appears that an obvious RIP or CHARGEN honeypot is used just as much as the variant that provides a real response.

QOTD

For QOTD, the real response group B does not have significantly more daily attacks than group D that provided a fake response. However, the fake small group C attracted more daily attacks than the real small group A ($p<0.01$). The fake group had a median of 1.93 attacks per day per honeypot, whereas the real response group has a median of 1.63 attacks/day. This means that the fake response honeypot detected *more* attacks than the real response honeypot.

This result is contradictory to expectations according to the hypothesis. A possible explanation is that the fake group had a slightly higher (53 bytes) response compared to the real group, which had a response size between 45 and 50 bytes. Another noteworthy observation is that the fake group provided a response that is static, and is thus always the same, whereas the real response was random.

SSDP

SSDP shows no significant difference between the two amplification groups with a different response (B and D). However, similar to the findings for QOTD, there is a significant ($p<0.01$) difference between groups A and C. The difference is much larger than with QOTD. With SSDP, the real response group A had a median of 17.15 attacks per honeypot per day, but for the fake response group, it was 30.49. Again, the real SSDP response varied between a length of 267 and 275 bytes and was dynamic. The fake response is static and had a length of 277 bytes.

This enormous increase seems odd at first. However, the squeeze experiment and resulting amplification factor provide a plausible explanation. Squeeze group 3, with a size similar to group A, had a median of 18.18 daily attacks per honeypot on average. This number is very close to the average group A. Group 4 had a median of 38.41 daily attacks per honeypot, much more. It also had a response size of 430 bytes, much larger.

Thus, the attack rate of group C appears to be somewhere in between. An SSDP request is generally 117 bytes. Thus, to get an amplification factor of at least 2, the response size needs to be $2 \bullet 117 = 234$ bytes. If then the size of the packet headers is added (42 bytes), the resulting minimum response size needs to be 276 bytes. This is one byte larger than the maximum response size of group A - which had a maximum of 275 bytes, and just 1 byte smaller than the response size of group C (277 bytes). Thus, this provides further evidence that the difference is caused by the difference in the amplification factor instead of the type of response.

NTP

There have been zero attacks that used the closed resolver NTP server. However, the closed resolver did not provide a reply to any incoming packet that was not a regular NTP time request. Thus, this cannot be used to build further conclusions. For NTP, there was a significant difference between the lower amplification group A and the higher amplification groups B and C. However, there is no significant difference between the vulnerable NTP server in group B, and the fake response that provided a large amplification in group D. Thus, with NTP it appears that adversaries are also indifferent between a fake response and an actual vulnerable NTP server.

DNS

With DNS, several types of services were run. Group A was running an actual, closed, and invulnerable DNS server. Honeypots in group B ran an open, vulnerable resolver. Group C provided an emulated DNS response and group D provided a fake response. Thus, there is an interest in comparing the emulated DNS response with the open vulnerable server and comparing the fake response to the open vulnerable server.

Figure 6.5.1 shows the number of honeypots that were used in DNS attacks. This shows several observations. First, there is a share of attacks that both use the emulated honeypots in group C, and the vulnerable DNS servers in group B. However, there are also attacks that use a lot of honeypots with only the actual DNS server, whereas the reverse is not true. There are no large attacks that use the emulated server, but not the open DNS server. Note that group C provided only a small response, so these differences may also be due to the amplification factor that was provided.

Group D provided a higher amplification, similar to the open server. Here, there are again no large attacks that merely used the fake service, while in reverse, there are large attacks that only used the open DNS server. These results show very different behavior from the other five services. This could mean that for DNS, some adversaries do not just verify that a response is returned with a threshold on the response size, but that some adversaries trigger on (parts) of content that is present in the response.

If some adversaries do make this distinction, it provides for a logical explanation of the observations in the figures: the responses of honeypots in groups C and D do not return such an expected DNS response, and thus an adversary may think that the honeypot is not usable in attacks. The attacks that only used group C or D, and not group B can also be explained: these honeypots replied to any request, even if the request was not a valid DNS response, whereas the open server would not reply to such a packet. Still, overall, the fake DNS server attracted a significant ($p<0.05$) number of attacks more than the open resolver.

(a) **(b)**

Figure 6.5.1 - Honeypot selection in DNS attacks

Overview

To conclude, for RIP, CHARGEN, and NTP, there are no observations that show that adversaries prefer a real server compared to a fake honeypot. For QOTD and SSDP, the fake honeypot was even preferred above the real system, although this appears to be related to the static response or slight difference in the amplification that the honeypots provided. Overall, for the five services, there is no conclusive evidence that shows adversaries evade the honeypots. The only service where a difference is observed is for DNS. Still, even with DNS, an obvious honeypot appeared to be more popular than the open DNS resolver.

6.5.2. Following SSDP Descriptors

The Simple Service Discovery Protocol provides an URL in its response that the requestor may use to obtain more information about the discovered service. This allowed for an additional test: do adversaries visit these URLs that are returned in the SSDP responses? If they do, this may be an indication that an adversary wants to verify if the server is an actual vulnerable SSDP service. Thus, to test whether adversaries follow up by querying this URL to verify the validity of the service, a web server providing this descriptor was also hosted on all the machines that ran the SSDP service. This web server then logged all the incoming requests. The URL led to a file called the rootDesc.xml.

In total, among all the servers, there were 145 requests for the rootDesc.xml. These requests were then matched with the scans that were in the logging database. 142 of 145 scans could be matched with this method. 14% of these scans requested the rootDesc.xml within a second after scanning. The other 86% of the scans had a delay between 10 and 14 seconds. The 10-second gap between the first and the latter groups may indicate a different scanning strategy: whereas one adversary may scan and upon receipt of a response may directly follow up by downloading the URL, another may collect responses separately, downloading and verifying the server in batches in a later stage.

Among the 142 matched scans, there were 64 unique scanning IPs. All these 64 IPs also only scanned the SSDP service, there are no scans recorded that show that these IPs scanned any other port, and there are also no attacks in the dataset that include these IPs. Furthermore, all of the scanned honeypots contained an URL to the rootDesc.xml, so there was no adversary that came across the rootDesc.xml by merely scanning for it.

97% of the unique IPs that scanned the SSDP services, also tried extracting more information from the control URL. Furthermore, 87% of the requests occurred only from the honeypots that also included a server version. The included version

was “Open WRT MiniUPnPd”. The other 13% can be attributed to group A that did not provide such a version. The URLs that squeeze group 3 provided were never visited. Only three different scanned packets were used, and the most used (87%) packet was not used in any of the SSDP attacks.

Based on these findings, it appears that the adversaries that actually looked deeper into the open SSDP servers were looking for vulnerable SSDP/UPnP servers rather than servers to exploit in DDoS attacks. Thus, adversaries scanning for vulnerable SSDP services to use for DDoS attacks have no preference for a real SSDP service compared to a fake SSDP service. This further strengthens the previous observation and acceptance of hypothesis *H4.a* and *H4.b.*

6.5.3. Conclusion

The overall conclusion is that based on the available observations, there is no evidence that suggests that adversaries actually *evade* honeypots, but that with more complicated systems, such as DNS, care needs to be taken while constructing the honeypots. If certain expected requests trigger an unexpected response, this may lead to adversaries not using these honeypots. In conclusion, hypothesis *H4.a* and *H4.b* are accepted, because there is no significant evidence that suggests that adversaries evade obvious honeypots or evade emulated services.

6.6. H5 - Honeypot Placement

Hypothesis *H5.a* and *H5.b* state that honeypots that are deployed in different locations - both geographically and at another provider - detect different attacks. In the first subsection hypothesis *H5.a* is analyzed - whether there is a difference between the honeypots in different geographical regions. In the second subsection, hypothesis *H5.b* is analyzed - whether there is a difference between the honeypots that ran at different providers.

6.6.1. H5a - Honeypots at Different Geographical Regions

Hypothesis *H5.a* states that there is a difference in attacks that are detected among honeypots that are located in different geographical regions. Thus, we try to disprove the null hypothesis that states that the honeypots that run in different regions detect the same attacks. Adversaries may be targeting specific geographical locations for their attacks, for example, because it is closer to their victim, or because they believe that one region has better overall connection reliability. To test whether adversaries prefer specific geographical locations - honeypots were distributed geographically during the experiment. However, the geographical location was not a choice for all the providers. To exclude the external variable of the provider having any influence in these results, only honeypots that ran on OVH are considered in this experiment. Honeypots on OVH ran in three different geographical regions: Europe, Singapore (Asia), and Sydney (Australia).

The honeypots at OVH only ran the main experiment, so no additional variables are of influence on these results. The number of honeypots that ran in the three different locations is similar: 24 honeypots were running in Europe, and 20 honeypots were running both in Singapore and Sydney - bringing the total number of honeypots to 64.

The method of analysis is similar to the previous subsection where the convergence of honeypots is calculated. In this section, the primary question is whether one region detected more attacks than another region. Similar to the previous analysis, the focus is on calculating the share of victims that an additional region is able to detect compared to another region.

Interpretation of Figures

This section covers each of the six services separately. The following abbreviations are used for the regions: EUR (Europa), SYD (Sydney, Australia), SING (Singapore, Asia). For each of the services, three plots are included. The title shows to which region the other regions are compared. Similar to the previous convergence analysis, the subsequent regions are cumulative. In other words, the third region is compared to the attacks that region one and two detected. Because the first region always detects 100% new attacks, it is excluded from the figure and the provider is named in the title. The y-axis shows the percentage of new victims that are detected.

RIP

The minimum number of detected victims among all regions is 135. Figure 6.6.1 shows the convergence of the different regions with RIP attacks. The European honeypots are able to detect all the victims. There are no additional victims that are detected by the honeypots in Asia and Australia. However, the other two regions also provide a good cover: with only Singapore selected, less than 1% of overall detected victims are missed. Australia is able to detect the least victims but still detects at least 92% of the victims.

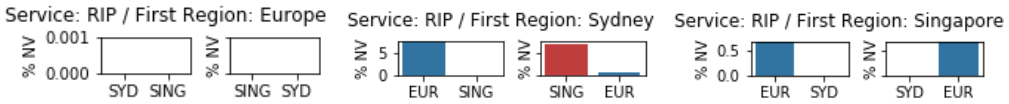


Figure 6.6.1 - Share of additional detected victims per regions for RIP

CHARGEN

The minimum number of detected victims among all regions is 90. Figure 6.6.2 shows the convergence of the different regions with CHARGEN attacks. The detected attacks vary greatly for each honeypot region. Roughly 50% of the attacks in any region is already detected by another region, but the other 50% of attacks are only seen in specific regions. For CHARGEN, there appears to be no specific region that detects most of the victims.

Figure 6.6.2 - Share of additional detected victims per regions for CHARGEN

QOTD

The minimum number of detected victims among all regions is 114. Figure 6.6.3 shows the convergence of the different regions with QOTD attacks. There are no detected attacks in Sydney or Singapore that were not already detected by the European honeypots at OVH. For both the Singaporean and Australian honeypots, it is also the case that a minimum of 96.61% of the victims was already detected by European honeypots, thus adding Europe only detects at most 3.39% more victims. Furthermore, when honeypots are placed in Australia, there are no additional victims detected in Asia. Overall, there are some minor differences, but it does not appear that for QOTD, the region is of any large influence.

Figure 6.6.3 - Share of additional detected victims per regions for QOTD

SSDP

The minimum number of detected victims among all regions is 879. Figure 6.6.4 shows the convergence of the different regions with SSDP attacks. All regions detect at least 85% of all victims that are targeted. Australia detects the most victims, both Europe and Asia detect less than 6% additional victims. Whenever Sydney is chosen, both Singapore and Europe provide the same addition to detected victims, so picking Sydney together with Europe or Singapore detects the most victims.

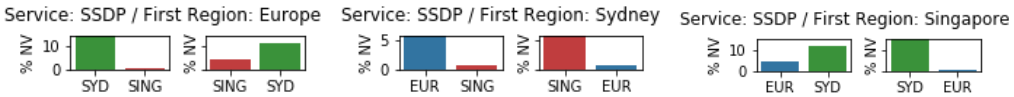


Figure 6.6.4 - Share of additional detected victims per regions for SSDP

NTP

The minimum number of detected victims among all regions is 517. Figure 6.6.5 shows the convergence of the different regions with NTP attacks. Europe is able to detect most victims compared to the other regions (82%). Thereafter, Sydney is able to detect the most attacks, and finally, Singapore does. Thus overall, to detect almost all attacks, at least two locations are required and it should include Europe.

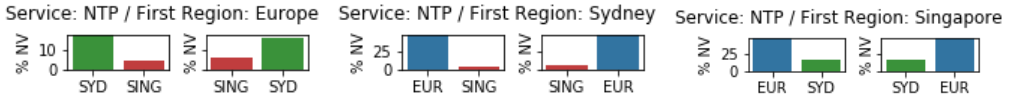


Figure 6.6.5 - Share of additional detected victims per regions for NTP

DNS

The minimum number of detected victims among all regions is 16. Figure 6.6.6 shows the convergence of the different regions with DNS attacks. The detected victims in the different regions vary greatly. Europe is able to detect most victims compared to the other regions (59%). Thereafter, Singapore is able to detect the most attacks, and finally, Sydney does. Thus overall, there is no specific region that is able to detect almost all victims.

Figure 6.6.6 - Share of additional detected victims per regions for DNS

Overview

Hypothesis *H5.a* is accepted. There are differences in the attacks that can be detected in various regions. This means that it is important to distribute honeypots among different geographical regions in order to detect a larger share in the attack landscape. However, depending on the service, a smaller or larger share can be detected by a specific region. Overall, Europe seems to be able to detect more of the attacks compared to Asia and Australia. Thus, while selecting between these regions, Europe might be the preferred location. In future work, more regions could be compared to see if the effect is similar in other regions such as Africa and America.

6.6.2. H5b - Honeypots at Different Providers

Hypothesis *H5.b* states that there is a difference in attacks that are detected among honeypots that are located at different providers. Thus, we try to disprove the null hypothesis that states that the honeypots running at different providers detect the same attacks. To validate the hypothesis, the attacks that are detected by honeypots running at different providers are analyzed. Thus, the main question is *if* there is a difference, and additionally *what* these differences are.

An external variable influencing these results is the location of where the providers were located. However, the honeypots running at the three providers Digital Ocean, Google Cloud, Azure, and a part of the OVH honeypots were all running in Europe. Thus, all these European honeypots were selected for comparison.

To further reduce other influences, only the main experiment is taken into account. To measure if there is a difference between the providers, a similar method as the previous convergence analysis is used. In this case, that means that the convergence is measured by, in each step, adding the additional victims that a new provider is able to detect compared to the other providers. This works as the following: if the first provider is Azure, the victims that this provider detected were all not previously detected, so the convergence value would be 100%. Then, a second provider is added (for example, OVH). The victims that this provider is able to detect grows the pool of detected victims by 25%, thus its convergence is 25%. This is then repeated for all five providers.

Because the order of which the providers are analyzed is of influence on the resulting convergence for that provider, all possible orders of providers are analyzed. The individual observations are then grouped in a box plot that is analyzed. This section will now cover each of the six services separately. The following abbreviations are used for the providers: Azure (Microsoft Azure), DO (Digital Ocean), Google (Google Cloud), and OVH.

RIP

Figure 6.6.7 shows the convergence of the different providers with RIP attacks. All the providers are able to detect a large share of victims - at least 94%. There are some slight variations on the detection share between the providers, and Microsoft Azure appears to add the most additional detections.

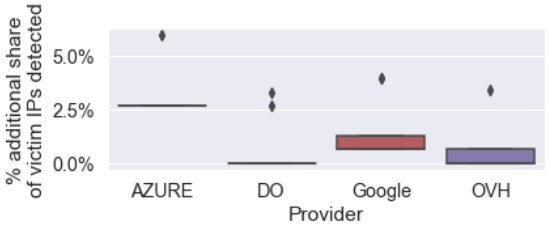


Figure 6.6.7 - Provider honeypot convergence for RIP

CHARGEN

Figure 6.6.8 shows the convergence of the different providers with CHARGEN attacks. For CHARGEN, any provider already detects the majority of victims. However, there are two providers that contribute more to additional detection: both OVH and AZURE. Further analysis shows that most of the AZURE detections also contain victims that OVH is able to detect.

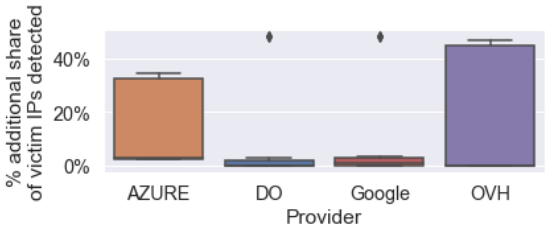


Figure 6.6.8 - Provider honeypot convergence for CHARGEN

QOTD

Figure 6.6.9 shows the convergence of the different providers with QOTD attacks. At least 69% of all victims are detected by honeypots at any of the providers. Honeypots that ran at Digital Ocean detect at least 20% of the total victims that were not detected at honeypots running at the other providers. The other three providers each detect a similar set of victims.

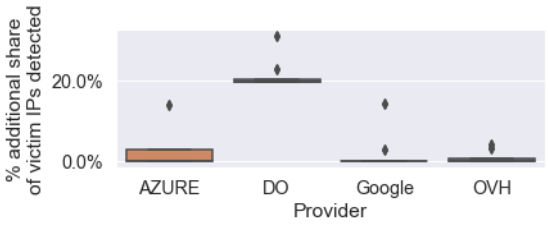


Figure 6.10.9 - Provider honeypot convergence for QOTD

SSDP

Figure 6.6.10 shows the convergence of the different providers with SSDP attacks. At least 83% of all victims are detected at any of the providers. Aside from the share of victims that are detected on honeypots running at any provider, there is also a small share of victims that are only detected by honeypots at honeypots running at specific providers. Overall, both AZURE and OVH appear to be the best addition, but the differences are relatively small.

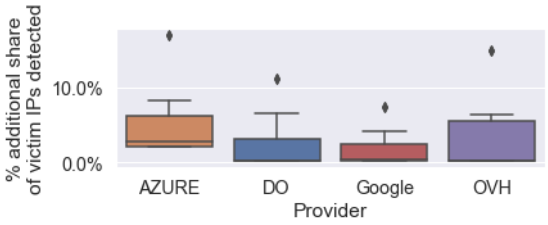


Figure 6.10.10 - Provider honeypot convergence for SSDP

NTP

Figure 6.6.11 shows the convergence of the different providers with NTP attacks. Only 16% of victims are detected at all the providers. This is much lower compared to the previously discussed services. However, the overall convergence already showed that for NTP and DNS, more honeypots were required for a good convergence. A plausible reason for this was the overall lower number of honeypots that were used in attacks. This suggests that many attacks used only honeypots in sets that were hosted at the same provider. Both Digital Ocean and OVH appear to detect a large share of victims that other providers did not detect.

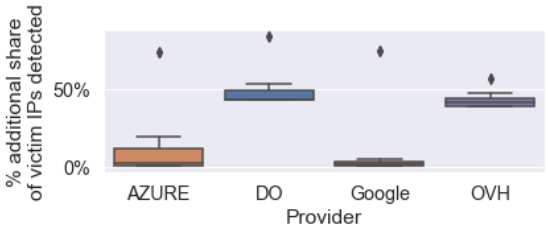


Figure 6.10.11 - Provider honeypot convergence for NTP

DNS

Figure 6.6.12 shows the convergence of the different providers with DNS attacks. Similar to NTP, the shared detection rate is much lower, at 36%. Google appears to detect the least additional victims, and AZURE detects the most additional victims. But also Digital Ocean and OVH have their own uniquely detected victims.

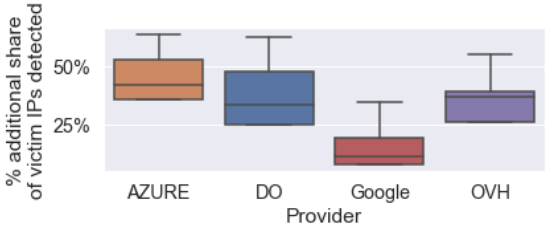


Figure 6.10.12 - Provider honeypot convergence for DNS

Overview

Hypothesis *H5.b* is accepted. Different providers detect different victims and attacks. There is no clear 'best choice' for a provider that is able to detect a majority of all attacks, although Digital Ocean shows the best overall detection share. Thus, depending on the service, different selection strategies may be used. However, in general, these findings suggest that it is best to distribute a honeypot network over a variety of providers to ensure that there is no bias of the data that is related to a provider.

6.7. Attack Types

Analysis for *H1* showed that there are different adversarial strategies. Section 6.2 showed that a portion of the attack landscape consists of multi-vector attacks. During the discussion on the convergence in section 6.1.5, subnet attacks were also first mentioned. Both multi-vector attacks and subnet attacks may be an indicator of more sophisticated adversaries. Thus, exploring both types of attacks may provide further inside in different adversarial strategies and sophistication.

The first subsection further analyzes multi-vector attacks, and the second subsection further analyzes subnet attacks. In both cases, the primary question of interest is whether there is different adversarial behavior with these two subtypes of attacks compared to regular attacks. In this analysis, the influence of the amplification factor is of primary interest.

6.7.1. Multi-vector Attacks

H1 states that higher amplification honeypots see more attacks than lower amplification honeypots. For single-vector attacks, this hypothesis is accepted. This section further analyzes the multi-vector attacks.

In total, there are 278 recorded multi-vector attacks, which is 1.37% of all detected attacks. There were 223 unique victim IPs registered among all the multi-vector attacks. Table 6.7.1 shows the number of attacks for each vector count in multi-vector attacks. Multi-vector attacks with 2 vectors are predominant with 262 out of the 278 multi-vector attacks. There are 14 attacks that use 3 vectors, and finally, only one attack using 4 and one attack using five different vectors.

Table 6.7.1 - Vector count overview in multi-vector attacks

Vector Count	**2**	**3**	**4**	**5**
Number of Attacks	262	14	1	1

The 4-vector attack used NTP, RIP, SSDP, and QOTD, and the 5-vector attack additionally used the CHARGEN protocol. In both these cases, DNS was not used. 4- and 5-vector attacks are not further discussed, because both of these only have one sample, and thus no representative analysis can be done.

Table 6.7.2 shows the combination of vectors in 2-vector attacks and 3-vector attacks. The largest share (65%) of 2-vector attacks use a combination of NTP and SSDP. 28% used the legacy protocol CHARGEN for multi-vector attacks. In 3-vector attacks, 71% used the legacy protocol CHARGEN. Thus, it appears that this legacy protocol is much more used with multi-vector attacks compared to single-vector attacks. On the other hand, both RIP and QOTD - the other legacy protocols - are hardly used. Furthermore, DNS is hardly used with 2-vector attacks, while with 3-vector attacks it is used much more, similar to single-vector attacks.

Table 6.7.2 - Multi-vector vectors used with 2-vector and 3-vector attacks

Vectors	**Services Used**	**# Occurrences**
2	NTP-SSDP	171
2	CHARGEN-SSDP	36
2	CHARGEN-NTP	32
2	CHARGEN-RIP	6
2	NTP-RIP	9
2	Other	8
3	CHARGEN-NTP-SSDP	9
3	DNS-NTP-RIP	4
3	CHARGEN-SSDP-QOTD	1

Figure 6.7.1 shows the number of multi-vector attacks over time. From the beginning until the end of the experiment, the number of attacks in each 12-hour window has been between 0 and 20 attacks. Similarly to the main experiment, the number of attacks seems to have increased after September 15th. However, there is no extreme number of increases, similar to what was seen with single-vector NTP attacks. This is interesting, because a large share of multi-vector attacks also use NTP, so this may suggest that the adversaries that use NTP in single-vector attacks and multi-vector attacks differ.

Figure 6.7.1 - Number of multi-vector attacks over time

Group Preference

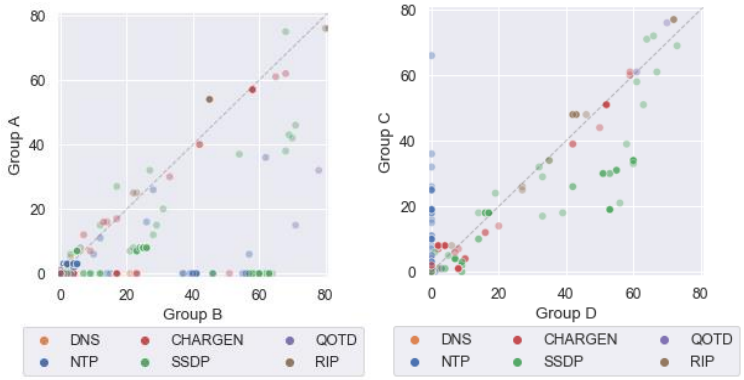
The main and squeeze experiment showed that there is an overall preference for higher amplification honeypots compared to lower amplification honeypots. Further analysis showed that there are possibly different adversaries that use different selection strategies: whereas some adversaries use all types of honeypots, others focus only on high amplification honeypots. Since multi-vector attacks use honeypots from different services in conjunction, one might assume that executing a multi-vector attack is perhaps more complex as more coordination and tools are required to execute these attacks. Thus, adversaries executing multi-vector attacks may be more sophisticated and therefore perhaps also will have a stronger preference for high amplification honeypots than single-vector attacks.

Figure 6.7.2 shows the preference for the amplification of honeypots for single and multi-vector attacks. The interpretation is similar to the analysis of honeypot selection preference in section 6.2.4. The figure shows the selection preference is separately for each of the six services. Each victim contributes to a data point, and the number of data points is included below the name of the service. The first number indicates the number of data points for single-vector attacks and the second number indicates the number of data points for multi-vector attacks. The results show that, except for DNS, the preference for high amplification honeypots seems to be *less* for multi-vector attacks than for single-vector attacks. These results are counterintuitive and are not according to the expectations. The next subsections further analyze these surprising results.

Figure 6.7.2 - Preference for high amplification honeypots of single and multi-vector attacks

Each of the boxplots shows the preference for high amplification honeypots of adversaries that use honeypots of the given service. The number below the service denotes the number of data points in the shows distributions. A value of 50% means that there is no preference for either lower or higher amplification honeypots as they are used both as much in an attack.

Figure 6.7.3 shows the honeypot selection in multi-vector attacks. The interpretation of the figure is similar to figure 6.2.14 which was discussed in subsection 6.2.4. The X and Y-axis show the number of honeypots that are used in one group compared to another group. These results show similar behavior as with all single-vector attacks. In other words, some multi-vector attacks use services from both low and high amplification groups, whereas others specifically only use the higher amplification honeypots. This could explain the counterintuitive results: some adversaries have no preference, whereas others do. The next subsection contains further analysis of the different types of adversaries that use multi-vector attacks.



(a) **(b)**

Figure 6.7.3 - Honeypot selection in multi-vector attacks

Different Adversaries

There are many attacks, and some may be executed by the same adversary. Thus, we can attempt to cluster attacks that show similar characteristics. Different attacks use different honeypots during the attacks, thus one way of clustering is to cluster attacks together that use an overlapping set of honeypots in attacks. The overlap can be quantified with similarity. In that case, when two attacks use the same set of honeypots in an attack, they are completely similar, and their similarity score is 100%. When one attack uses half of the honeypots that another attack uses, their similarity score would be 50%, and when two different attacks use two completely different sets of honeypots, their similarity score would be 0%.

The next step is to determine when an attack is considered to be part of the same cluster. By manual inspection, it appeared that sometimes an attack used one more or less honeypot in an attack compared to another attack. Thus, while clustering, there needs to be a small margin for error, for when the attacks both use many of the same honeypots in an attack, but not all.

To account for this margin of error, attacks are considered to be clustered if 95% of the honeypots that were used in an attack overlap. This means that at most 1 in 20 honeypots do not have to overlap and attacks are still considered to be part of the same cluster. By using this clustering, 82% of all multi-vector attacks can be clustered, leaving only 50 out of 278 attacks unclustered. The clusters were then manually verified to confirm there was no overlap in the selected honeypots, and also characteristics such as the total number of unique honeypots used, the duration of the attacks, and the rates of the packets and bytes were compared to confirm the attacks were clustered correctly.

This led to 20 different clusters consisting of 228 attacks. Most resulting clusters were small and contained less than 10 targets. Furthermore, there was not enough information to make any meaningful conclusion about these clusters. For example, the victims in these clusters had no reverse DNS record that provided more information, and there was no further information available about domain names that were hosted at these IP addresses. However, there were two large clusters that accounted for 27% and 23% of all multi-vector attacks. For these two clusters, a more in-depth analysis was possible because for the targets of these clusters there was more information available.

The first cluster consists of 76 of the 278 multi-vector attacks. The analyzed information includes the reverse DNS names of the victims as well as information about domain names that were hosted at these IP addresses[27]. This information shows that the victims in the cluster consist of various hosts without known hosted domains, a lot of different cloud providers as well as anti-DDoS providers. More interestingly, the hosted domains show a mix between sites that appear to have a normal domain name (such as a brand name) and random domain names (such as 60708066.com). The cluster has victims in various countries such as Hong Kong, China, Singapur, South Africa, and the US. The majority of attacks in this cluster targeted their victim IPs on port 80, which may indicate that the goal was to mainly target websites. This traffic was mostly generated with SSDP. Aside from SSDP, a lot of NTP honeypots were used in these attacks that targeted various ports, and finally, some attacks used QOTD, targeting port 44109. There is no known service that uses this port, so this may as well be a random port that is targeted.

The second large cluster consists of 63 of the 278 multi-vector attacks. These addresses are located all over the globe. Within this cluster, similar to the first cluster, mainly SSDP is used, targeting port 80. It also uses NTP while targeting several ports. The DNS information and domain names that are hosted among the various targets show a wide variety of victims: large telecom providers, Internet Service Providers (ISPs), and gaming communities. Reverse DNS records indicate that the victims within the large telecom providers and ISPs are largely customer home addresses.

The large variety of victims that is targeted, which specifically include gaming websites and even home broadband connections indicate that the actor that is responsible for the attacks in these clusters is offering these attacks through a booter/DDoS-as-a-service service. These services have many customers that can choose their targets for a small payment.

In the previous subsection, it appeared that multi-vector attacks had a less strong preference to higher amplification honeypots than single-vector attacks. This was an unexpected result because there was an expectation that multi-vector attacks were more sophisticated. Now that it is clear that there are different clusters, the analysis can be redone by comparing the suspected booter clusters with the other multi-vector attacks. Figure 6.7.4 shows this preference for the different types of clusters. Note that only NTP and SSDP are compared because the other services did not contain enough data points in all groups to make a meaningful analysis. The number below the services indicates the number of data points for each of the three groups, in the same order as the groups are displayed.

[27] This information was obtained by using RiskIQ DNS heatmap data API

The results show that the suspected booter service clusters appear to be *less* rather than *more* biased towards high amplification honeypots compared to single-vector attacks and other multi-vector attacks. This suggests that either our assumption is incorrect about the clusters being a booter service or that these booter services are less sophisticated. One example indicates that booter services may be less sophisticated. In 2016, one 15-year-old was arrested after earning $385,000 from providing such as service [27]. These booter services are perhaps less sophisticated and merely used for making quick money without advanced adversarial tactics, whereas more sophisticated attackers that are targeting specific victims might be more selective in the honeypots that they use.

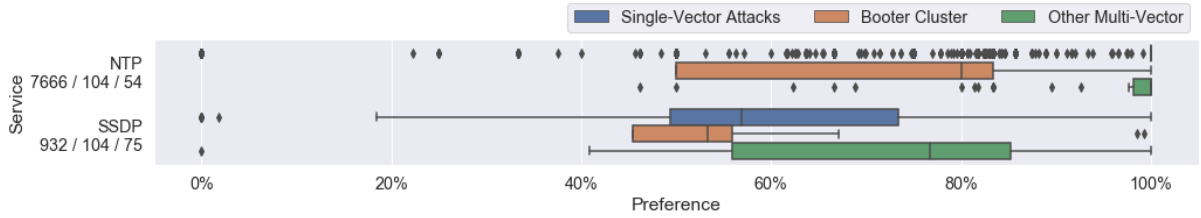


Figure 6.7.4 - Honeypot preference for high amplification honeypots of single vector and two types of multi-vector attacks

Repeating Attacks

Some victims are attacked multiple times. This allows us to analyze the honeypots and vectors that are used during each attack. 31 of the 223 unique victims in multi-vector attacks were attacked more than once. In 65% of these attacks, the same vector was reused in subsequent attacks. This means that with more than a third of multi-vector attacks, different vectors were used in subsequent attacks. This means that it is important to gather data over a longer timeframe, because a snapshot of the current attack landscape may provide an incorrect insight into popular vectors. When different vectors were selected in subsequent attacks, both SSDP and NTP were the most common vectors that were selected.

In the cases where the same vectors were reused, all attacks reused exactly the same set of honeypots for their attack. Thus, adversaries did not select a different set of honeypots for their attacks. This is not necessarily expected: rotating the pool of available vulnerable services and thus using different honeypots during each attack may assist in preventing that the traffic that the honeypots reflect is blocked by DDoS counter-measures, helping an adversary in achieving more effective attacks. However, this insight is helpful for researchers for detection and fingerprinting. When adversaries reuse the same set of honeypots in multiple attacks, this allows research projects to fingerprint adversaries by the set of honeypots that they use

Validation

This subsection analyzes which service ports are scanned in conjunction. This information is used to estimate if attacks were properly classified as multi-vector or single-vector attacks. This research only includes honeypots for six different UDP services, but there are also other services that are frequently scanned, and that can be used for amplification attacks. Therefore, attacks could be classified as being a single-vector only, but that are, in fact, multi-vector attacks. Although no certainty can be provided, scan combinations can be used to provide an estimation. For example, if CHARGEN is frequently scanned together with QOTD, and an attack is detected that only uses CHARGEN, there is an assumption that the attack was only a single vector, as otherwise there is the assumption that QOTD would also have been used.

Thus, the ports that are most frequently scanned in conjunction are analyzed. In other words, we want to know which other ports are the most frequently scanned, considering that CHARGEN is scanned. Table 6.7.3 shows the toplist of scanned ports given that one of the six services is also scanned. The six services of interest for this thesis are marked with a light blue background.

The main takeaway in table 6.7.3 is that for all of the six services, except DNS, all the other five services are among the most frequently scanned other ports. Even for DNS, four out of the five are in the toplist. This means that whenever one of the six services is scanned, adversaries appear to prioritize scanning to find servers for the other five services. This suggests that we are detecting multi-vector attacks correctly because when attacks only use one of the six services in their attacks, there is now a high chance that it was merely a single vector attack - if we assume that the scanning activity is a valuable indicator.

In 2-vector attacks, the most popular combination was NTP-SSDP. While DNS was scanned most frequently with NTP, SSDP was the 2nd most popular scanned protocol in conjunction with NTP. RIP and CHARGEN were also in the top 10 of the most popular scanned protocols in conjunction, and these combinations also appear in the vectors that are used in conjunction with multi-vector attacks. The protocols that were most popular with CHARGEN also appear in the 2-vector CHARGEN

attacks: NTP, RIP, and SSDP. This further suggests that multi-vector attacks were detected correctly as a multi-vector attack, but could also mean that the actual vector-count of the detected multi-vector attacks could be higher.

Table 6.7.3 - Toplist of scanned ports when given service is scanned

	Position									
Service	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
CHARGEN	LDAP 4.75%	QOTD 4.66%	SSDP 4.31%	Sun RPC 4.18%	NTP 4.09%	NetBIOS 4.06%	RIP 4.01%	SNMP 3.81%	DNS 3.72%	MSSQL 3.23%
QOTD	CHARGEN 4.77%	LDAP 4.42%	Sun RPC 4.20%	RIP 4.16%	NetBIOS 4.01%	SSDP 3.68%	SNMP 3.48%	MSSQL 3.36%	DNS 3.33%	NTP 3.30%
RIP	NetBIOS 4.49%	Sun RPC 4.35%	SSDP 4.18%	SNMP 4.08%	NTP 3.97%	DNS 3.86%	LDAP 3.83%	QOTD 3.64%	CHARGEN 3.59%	IPMI 3.08%
DNS	SSDP 6.87%	NTP 5.29%	SNMP 4.78%	NetBIOS 3.78%	RIP 3.75%	Sun RPC 3.54%	LDAP 3.45%	CHARGEN 3.24%	MSSQL 2.89%	IPMI 2.83%
SSDP	DNS 5.45%	LDAP 4.03%	NTP 3.89%	SNMP 3.53%	NetBIOS 3.38%	RIP 3.21%	CHARGEN 2.97%	Sun RPC 2.90%	QOTD 2.48%	MSSQL 2.36%
NTP	DNS 5.37%	SSDP 4.98%	SNMP 4.76%	LDAP 4.17%	NetBIOS 4.01%	RIP 3.90%	CHARGEN 3.61%	Sun RPC 3.60%	MSSQL 3.03%	QOTD 2.85%

Conclusion for multi-vector attacks

H1 states that higher amplification honeypots detect *more* attacks than lower amplification honeypots. For single-vector attacks, this hypothesis was accepted. This section showed that *H1* can also be accepted for multi-vector attacks, but that there are certainly differences.

At first, it appeared that multi-vector attacks did not have a strong preference towards high amplification honeypots over lower amplification honeypots. However, further analysis showed that there appear to be different adversaries that execute multi-vector attacks that both show a different preference for the type of honeypots. Suspectedly, some attacks that are from suspected booter services show a less strong bias towards higher amplification, whereas other multi-vector attacks show a stronger preference to high amplification honeypots.

To ensure that multi-vector attacks were classified correctly as multi-vector attacks, scans were analyzed. While scanning for the six services that are used in this research, the other services also seem to be readily present as popular scanned services that are scanned in conjunction. This gives direction to the assumption that in multi-vector attacks, these services might be used together. Therefore, when single-vector attacks are detected, there is a higher probability that they were single-vector attacks compared to being a multi-vector attack together with an undetected vector.

6.7.2. Subnet Attacks

Many attacks target individual targets, for example, a broadband internet connection, a website, or a game server. Thus, in these attacks, only a single address is targeted. However, there are also attacks that target multiple addresses in the same network. For example, they target both a gameserver and a webserver of a gaming community, at the same moment in time. In this subsection, all attacks are analyzed that attacked more than one IP address in a single /24 network. A /24 network contains 255 unique IP addresses and contains all addresses that differ only in the last digit. For example, 1.2.3.4 is considered to be in the same subnet as 1.2.3.175 and 1.2.3.255.

There are several questions of interest: of how many addresses do these attacks consist? And is there a difference in the preference towards higher amplification honeypots if an attack used many addresses in a subnet, compared to only a few? The first step is defining when an attack is considered to be a subnet attack.

Table 6.7.4 shows the number of subnet attacks when attacks are considered to be subnet attacks merely when multiple IPs in the same network were attacked, as compared to the additional condition that these attacks should occur at the same time, and thus have an overlap in their start and end time. The difference between these two categorizations shows that more than 50% of the attacks that attacked multiple IPs in the same subnet, did not occur in the same time window. This is plausible since many services are hosted at cloud providers. Thus, attacks that merely attack the same subnet during the three-week duration of the experiment can realistically attack different customers at these providers. As such, an attack is

only considered to be a subnet attack if there is more than one IP address targeted in a network and when these attacks have overlapping start and end times.

Table 6.7.4 - Number of subnets that were attacked, split by the number of IPs in that subnet

Subnet Attack Size	2	3	4	5-256
All-Time	420	97	43	88
Time-overlap	130	44	11	36

Furthermore, table 6.7.4 shows that the majority of subnets that were attacked were only targeted on 2-4 unique IPs (with time overlap). These attacks were much more common than larger subnet attacks that targeted between 5 and 256 IPs in the same subnet.

Table 6.7.5 shows the number of attacks that subnets with a given unique number of IPs were attacked with. The majority of subnet attacks were targeted with 2-4 unique IPs, but the majority of attacks are from the larger subnets that were attacked. More than half (52%) of subnet attacks were caused by subnets attacks where the full subnet was attacked.

Table 6.7.5 - Number of attacks in subnet attacks with given size

IPs in Subnet	2-4	5-237	255	256
Attacks	1810	2703	3327	1517

Figure 6.7.5 shows the vectors that were used in subnet attacks where a given number of IPs were targeted in the same subnet. The most frequently used vector for all the attacks is NTP. For subnet attacks where 255 or 256 out of the 256 addresses were targeted, it was the only used attack vector. For subnet attacks that targeted between 5 and 237 addresses in a subnet, DNS was also used as a vector, but only in 7.5% of the attacks. Only with smaller subnet attacks, where 2 to 4 IP addresses were targeted, RIP, CHARGEN, QOTD, and SSDP are also used minimally.

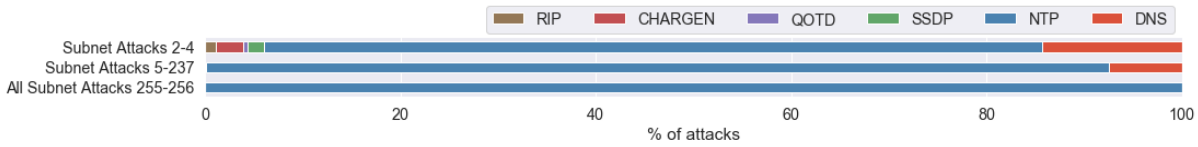


Figure 6.7.5 - Vector usage in different sizes of subnet attacks

Figure 6.7.6 shows an analysis of the preference for higher and lower amplification honeypots among the different groups for subnet attacks. Only NTP is included because this was used in almost all subnet attacks. The analysis follows a similar method as in section 6.2.4. Each unique victim represents a data point, the number of data points in the different categories are noted on the left side of the figure. There are hardly any differences between the different subnet groups. However, compared to all single vector, single target attacks, the preference towards higher amplification honeypots is much stronger for subnet attacks than for regular attacks. This suggests that adversaries that execute subnet attacks, use a selection strategy where high amplification honeypots are preferred. This suggests that adversaries that execute these attacks are more sophisticated than other adversaries.

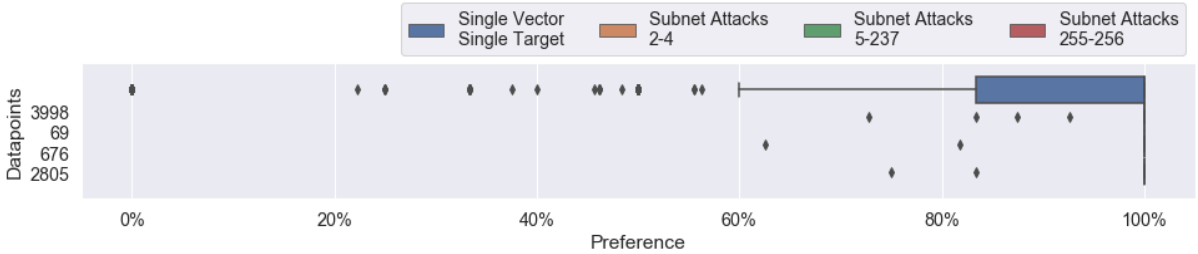


Figure 6.7.6 - Preference for high amplification honeypots of the subnet attacks with NTP

Subnet attacks target multiple addresses within the same subnet, but these attacks can be executed by using various honeypot selection methods. For example, adversaries could select a small set of honeypots for each individual target, and use another small set for the next target and so on. Another strategy would be to use the same set of honeypots for all the different targets. A benefit of the first strategy would be to either evade detection - the data would originate from more

different sources, and this strategy may also help to evade rate-limiting measures because more overall honeypots are used in that case. Figure 6.7.7 shows a cumulative density function of the overlap of used honeypots in subnet attacks. It shows the overlap of the set of honeypots that are used for each subnet that is attacked. The data shows that almost all subnets IPs that are attacked, are attacked with the same set of honeypots.

Figure 6.7.7 - CDF of overlap of the used honeypots within subnet attacks

Figure 6.7.8 shows the distribution of the number of honeypots used for regular attacks compared to subnet attacks. There are not enough data points for group A, because subnet attacks almost solely used the higher amplification honeypots in the groups B and C. For groups B and C, there are several observations. First, the average number of honeypots that are used for subnet attacks is significantly higher compared to regular attacks while comparing them with the Mann-Whitney U-test ($p \leq 0.01$). For regular attacks, the average number of exploited honeypots is 10.56, while for the three different subnet groups it is 31, 13, and 14.9 respectively. The second observation is that the number of used honeypots is much more concentrated with subnet attacks, whereas for regular attacks, a great variety of the number of honeypots is used in attacks.

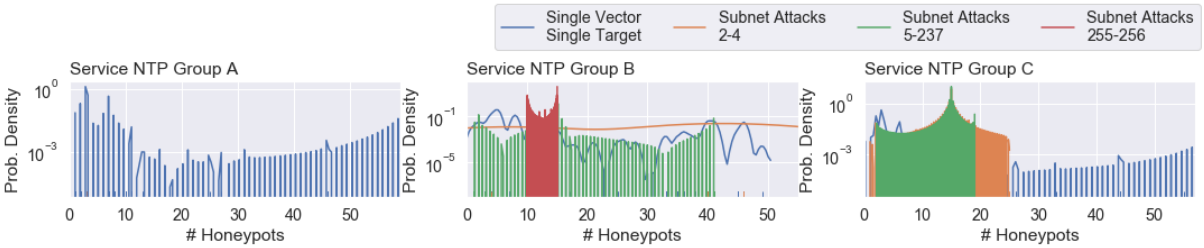


Figure 6.7.8 - Number of honeypots used in subnet attacks compared to other attacks

Conclusion for subnet attacks

H1 is also accepted for subnet attacks. In conclusion, subnet attacks differ considerably in size. Whereas some subnet attacks only attack a select few IP addresses in a subnet, there are also subnet attacks that target all the individual subnet addresses available. However, they have in common that NTP is the predominantly used protocol in the attacks. The largest subnet attacks are almost solely using this protocol. Furthermore, all subnet attacks seem to have a very strong preference for high amplification honeypots, significantly stronger than regular attacks. The majority of subnet attacks re-use the honeypots for all of the individual targets in the same subnet, and additionally, the average number of honeypots used in attacks seem to be larger than regular attacks. Overall, adversaries that use subnet attacks appear to be more sophisticated than other adversaries executing regular attacks.

6.8. Attack Effectiveness

This section analyzes the availability of the victims during the DRDoS attacks that took place during the experiment. There is no information available that allows us to directly analyze the availability of the victims during the DRDoS attacks. However, there is some indirect information that allows us to estimate the availability of the victims during the attacks.

The Internet Protocol (IP) suite has a diagnostic and control protocol that supports other protocols in the IP suite, such as IPv4. This protocol is the Internet Control Message Protocol (ICMP) and is also used to send control messages about unsuccessful operations. For example, when a packet cannot be delivered due to any reason or when the targeted host does not exist, the sender is notified with an ICMP packet. Although not all networks respond with the corresponding ICMP packets, the limited information that does get returned can provide insight into the impact of a DDoS attack. For example, the ICMP information may indicate that a host was no longer reachable during a DDoS attack, an indication of an attack that was successful. On the other hand, it may also provide information that notifies the sender that the packet could not be delivered because delivery was blocked, which may indicate counter-measures that are deployed.

During the experiment, ICMP packets that were received by the systems that the honeypots were running on, were recorded. The ICMP messages also contain an ICMP type, that indicates what type of information the packet contains. This research focuses on ICMP type 3, which is a return type that indicates that the destination was unreachable, and thus a packet could not be successfully delivered.

Figure 6.8.1 shows the number of ICMP responses over time that are caused by undelivered packets because the destination host was unreachable. The graph looks similar to the number of attacks that occurred over time. The data points are not concentrated only in a few moments in time, but data is available during the full duration of the experiment.

Figure 6.8.1 - Number of ICMPs received over time

Each ICMP response not only contains a type but also contains a code that further describes the control message. In the case of ICMP messages that notify that the destination is unreachable, these codes further specify *why* the destination was unreachable. Possible reasons include that packets were blocked, or that the host was no longer reachable. Thus, analyzing these return codes may provide more insight into the effectiveness of a DDoS attack.

During the experiment, 7 different ICMP codes were detected among the destination unreachable ICMP types. The ICMP packets are then linked to detected attacks, by using the destination, ports, and timeframe. Note that it is possible that one attack has multiple registered ICMP types. For example, it could be possible that first, a destination is unreachable because the host is not responding, but that later the packets are blocked and then codes are received that indicate that the packets cannot be delivered to the network. Table 6.8.1 shows the different ICMP return codes and for how many attacks these different codes were detected. The codes are now discussed individually.

Table 6.8.1 - Number of attacks where ICMP return codes were detected

ICMP Code	0	1	2	3	4	10	13
Occurrence	135	64	5	2183	39	269	504

Code 0 is a reply that states that the destination network is unreachable. This indicates that not only the host went offline or is unrouteable, but that the network that the host contains is also not reachable. This could be an indication that the attack caused collateral damage and not only brought down the target but also the network the target resided in. Code 1 indicates that the target host is no longer reachable. These codes were returned 134x for code 0 and 64x for code 1. For code 0, 33% of these codes were related to subnet attacks and 36% for code 1. Thus, it is not the case that code is only detected with subnet attacks.

Codes 2 (protocol unreachable), 3 (port unreachable), and 4 (fragmentation needed and do not fragment was set) are unreliable indicators and cannot be used for analysis. For example, a port or protocol may be unreachable at a destination, but a DDoS attack could attack the host on different ports that indeed are not used - but still cause congestion for the host. The following example provides a better understanding of why these codes are not a reliable indicator. Consider a server that runs a web server that serves websites at port 80. The server is attacked at port 100. During an attack, packets that our honeypots reflect target port 100, and one of the previously mentioned codes are returned. This does not provide any information about whether port 80 (the webserver) is offline or not. Thus, with these detected codes, no reliable estimations can be made.

Codes 10 and 13 are used to communicate that communication with the destination host is prohibited. This could have several reasons, for example, because the target host has its traffic null routed or because the receiver disallows the packets to be delivered - because, for example, it detected that the traffic was DDoS traffic. For analysis, these types are considered to be activated as a countermeasure in a DDoS attack. Code 10 was returned for 269 attacks and code 13 in 504 attacks. In 70 cases, other ICMP codes were detected before the codes 10 and 13 were detected. This is in 9% of the attacks where a code 10 or 13 was detected. Since in 91% of the cases, no other codes were received before codes 10 and 13, this suggests that these counter-measures were mostly already in effect before our honeypots started participating in attacks, or that counter-measures were already deployed before a host went offline.

Figure 6.8.2 shows the share of the different codes in ICMP messages that were received for the six different services. For all the different services, the majority of control messages include prohibited routing of the packets (codes 10 and 13). A much smaller portion is a notification that the host or network is currently unreachable (codes 0 and 1). Further analysis will put the number of received ICMPs for each service in perspective to the number of attacks that each of the honeypots with the different services detected.

Figure 6.8.2 - Share of ICMP codes per service

Table 6.8.2 shows the total number of targets that were targeted for each protocol and the number of unique targets that were offline due to an attack or were countermeasures were deployed. The share of victims that went offline is relative to all detected victims for that service. The targets are normalized by grouping subnet attacks. This means that IP addresses that belonged to the same subnet attacks are considered to be a single target.

Attacks using the legacy and deprecated protocols QOTD, RIP, and SSDP seem to be more effective than other protocols in bringing a target offline. DNS and NTP seemed to be less effective overall. Furthermore, it appears that in more cases counter-measures that were deployed than there are cases where an attack was successful in bringing a target offline.

Table 6.8.2 - ICMP unreachable and countermeasure-codes and share of total attacks - normalized for subnet attacks

Protocol	# Targets for Protocol	# Targets Unreachable	# Targets counter-measures	% of targets offline	% of targets counter-measure
QOTD	172	5	4	2.91%	2.33%
RIP	158	4	9	2.53%	5.70%
DNS	657	1	7	0.15%	1.07%
CHARGEN	405	9	43	2.22%	10.62%
SSDP	1109	21	82	1.89%	7.39%
NTP	5311	53	189	1.00%	3.56%

Some targets that went offline during the attack, were already offline when our honeypot system detected the attack. Some attacks were blocked by counter-measures. One question remains, and that is if, and how the detected codes are concentrated on attack characteristics. In particular interest: are ICMP messages detected only with larger attacks, or also with smaller attacks?

Figure 6.8.3 shows the distribution of the number of honeypots that an attack used for the attacks where an ICMP response was received. For the legacy and deprecated services, most ICMP responses are detected when many honeypots were used in an attack. DNS does not have enough data points for a meaningful analysis. SSDP shows a similar trend to the first three services, but more interestingly, it appears that when the attacks grow larger, attacks see less successful host/network unreachable ICMP responses, and see more countermeasures deployed. This suggests that launching a smaller attack is more effective in some cases than merely generating the largest attack. NTP shows similar results as with SSDP, but the overall number of honeypots is much smaller.

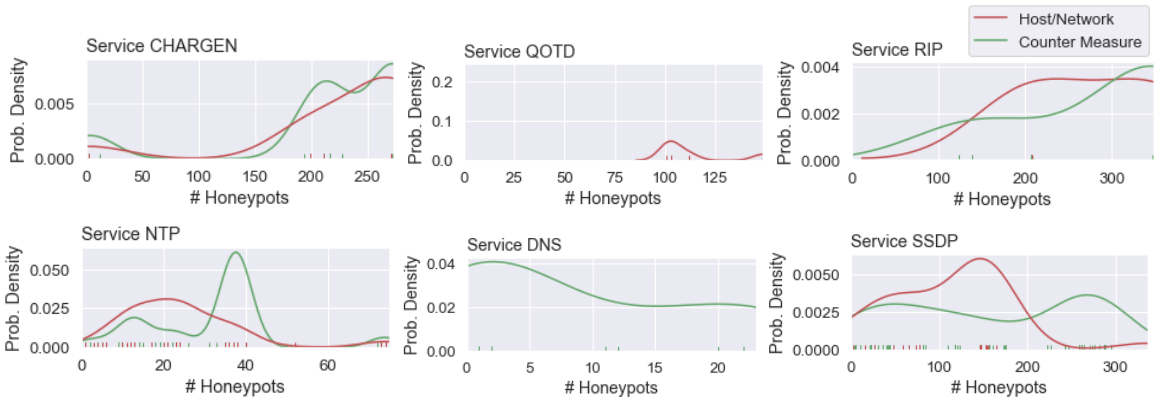


Figure 6.8.3 - Probability density function of the number of honeypots used that were used in attacks where ICMP messages were returned

Overall it appears that larger attacks receive more ICMP responses than smaller attacks. Thus, this may be an indication that larger attacks have more influence on bringing a target offline or triggering counter-measures. However, different providers handle ICMP return codes differently. Some providers may block the traffic without returning ICMP messages. Unfortunately, no further data is available to test these assumptions. For future work, introducing probes to verify the reachability of a destination during and after an attack may provide more insight into the effectiveness of attacks. Analysis of these probes may also be used to verify that the detected ICMP return codes provide an accurate representation of the effects of attacks.

6.9. Retention

After a vulnerable service is no longer available and therefore is no longer usable in attacks, it takes a while before an adversary is aware of the unavailability of the vulnerable service. Therefore, it is of interest to analyze for how long adversaries try to keep using these now longer available services in their attacks. If a server is no longer used in attacks, an adversary has probably rescanned the server and discovered that his resources can be better spent. Furthermore, different adversaries may show different behavior. This section will also analyze if there are differences in the detected attacks that were detected among the honeypots in the retention period.

In this thesis, the name for the period from the end of the 'active' experiment until there are no more attacks, is called the retention period. To measure the retention, some servers kept running after the experiment concluded and were configured to run honeypots in passive mode again. This allowed it to simulate an offline server, while still being able to record all incoming data towards the honeypots that were running on them.

The first subsection analyzes returning scans. This may provide a first indication of how quick adversaries are aware of a vulnerable server that is now offline. The section thereafter describes the number of servers that were still active and were running in passive mode after September 27th. This is followed by a subsection that describes the number of attacks that were detected after the active experiment stopped. The final subsection concludes this section.

Returning Scans

Kührer et al [50] found that many vulnerable servers were only temporarily available. As soon as the vulnerable services appeared, a portion of the available server disappeared again within a week. For some protocols, this share may be up to 50%. Therefore, adversaries should verify that the vulnerable servers they discovered are still active to ensure that their attacks will be efficient. The cyclic behavior of available vulnerable servers is also called IP churn. Because of this IP churn, adversaries may be rescanning the honeypots frequently.

The analysis of returning scans should ideally only include scans that originate from adversaries and excludes any scans from research projects. To make the distinction between adversaries and research projects, reverse DNS records of the scanning IP address are used. These reverse DNS records reveal more information about the scanning IP addresses. During a manual inspection, it became evident that a lot of these projects contained the string 'scan' or 'research' in their reverse DNS string. Thus, with a best-effort attempt, all scans that originated from sources containing these strings were excluded for this analysis.

For some days there is a low number of data points. Thus, while merely comparing the scans of a day to the day before, there are varying results. Analyzing the scans with a three-day lookback yields a more clear pattern. Figure 6.9.1 shows the daily share of repeating scans. Overall, a large share of the scans that were detected, were from source IPs that also scanned the same service in the days before. This could mean that a high number of adversaries is rescanning the services to validate whether they are still vulnerable and available. In reality, this number could be higher because the same adversary may use multiple scanning IPs.

Figure 6.9.1 - Repeating scans with a 3-day lookback

Each data point shows the share of scanning IP addresses that were already detected in the 3 days prior to that day.

Runtime

Some machines where the honeypots were running on ran out of resources on September 27th, when the experiment concluded. However, many servers had resources left and were thus available for a longer duration. For that reason, their configuration was changed from the active response phase back to the passive listening phase.

Figure 6.9.2 shows the number of honeypots that kept running in individual groups and experiments. Until October 15th, there was a continuous decline in the number of running honeypots, and thereafter only the main experiment and squeeze experiment honeypots remained operational. After November 13th, the number of honeypots dropped to 13. Taking the convergence into account, the number of servers that kept running should include enough honeypots to make an accurate estimation of the number of attacks that happened until at least November 13th.

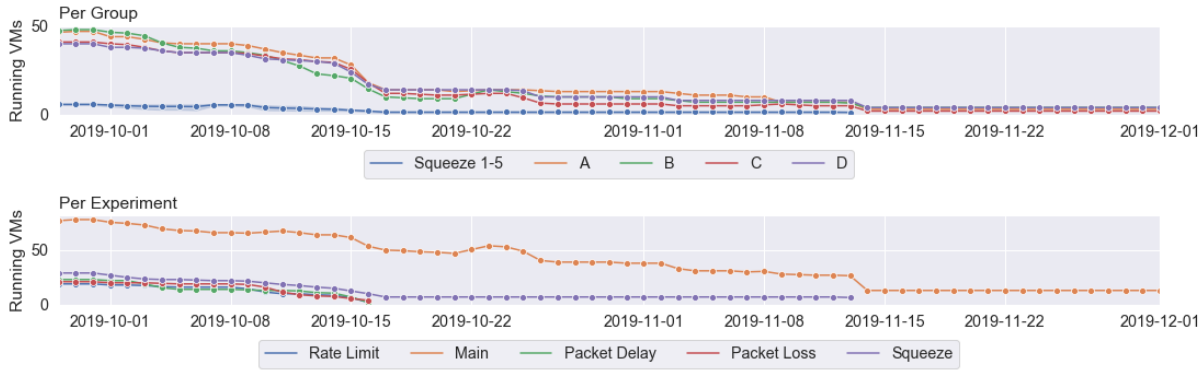


Figure 6.9.2 - Number of running VMs per group and experiment after the active experiment concluded

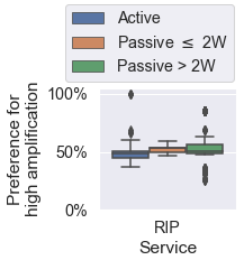
Attacks

This subsection analyzes the attacks that were detected by the passive honeypots that continued running after the primary experiment concluded. The six services are analyzed separately. All the services include a figure that shows the timeline of the number of attacks before and after September 27th. The vertical red line in the timeline plots denotes the switch from the active to the passive mode Furthermore, the first two weeks of the passive phase is compared to the phase thereafter with an analysis similar[28] to the analysis of the preference for higher and lower amplification honeypots. The included figures show a boxplot with the preference for higher or lower amplification honeypots among attacks in the three different time windows. Significantly different distributions may indicate a difference in adversarial behavior. These significant differences and their implication are discussed with each service.

[28] the calculation of preference was discussed in section 6.2.4.

RIP

RIP shows a pattern where attacks kept being detected for a long duration after the honeypots stopped responding. After November 4th, hardly any attacks were detected until November 22rd, but then new attacks start occurring again. This did not decrease further until the end of the measurement period. The honeypots that were used in the attacks in the passive phase do not show a stronger preference for high amplification honeypots.



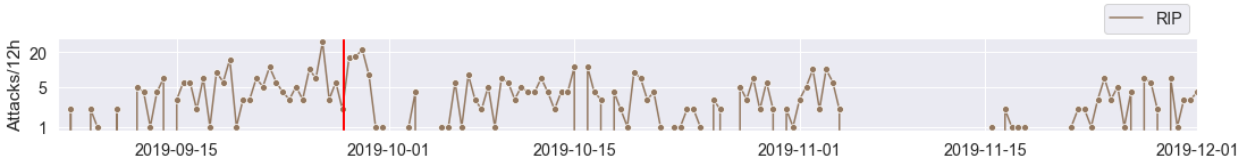
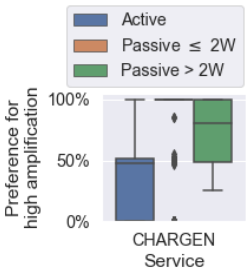


Figure 6.9.3 - Number of attacks during the active and passive phase for RIP

CHARGEN

CHARGEN honeypots started to detect more attacks after the honeypots already stopped responding. In fact, there are more CHARGEN attacks detected after in the passive phase than in the active phase of the experiment. After October 10th, the number of detected attacks dropped to almost zero. From that time on, there are very similar patterns visible than with RIP. First, there are some attacks until November 4th, then there is a gap without any attacks, with a small peak around November 16th, and on November 22nd.



Previous analysis showed that for CHARGEN, there was no significant difference between high and low amplification. However, the large number of attacks that occurred in the first two weeks of the passive phase show a very strong - an almost complete preference for high amplification honeypots. The attacks that occurred thereafter again show a much lower preference for high amplification honeypots. Overall, attacks in the retention period show a significant ($p < 0.01$) stronger bias towards higher amplification honeypots than the active phase. This suggests that the findings for hypothesis *H1* of this thesis report for CHARGEN may be incorrect, and that also for CHARGEN there are adversaries that mostly use high amplification honeypots. Thus, also for CHARGEN, the amplification factor might be a primary factor of influence. Furthermore, attacks that occurred after two weeks of the passive phase have a less strong bias towards higher amplification honeypots. This may suggest that adversaries that preferably use high amplification honeypots, also more often verify that the vulnerable servers are still available.

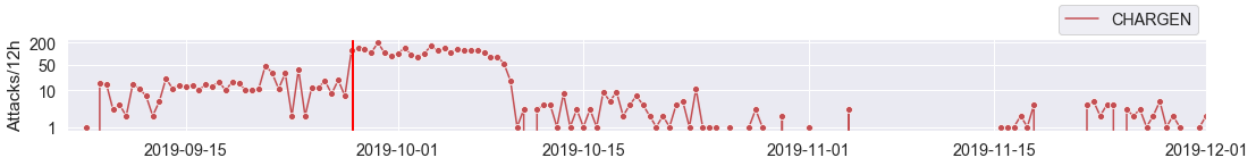
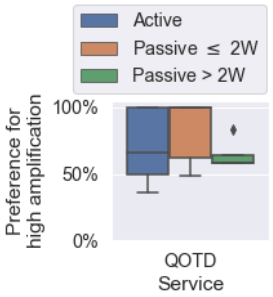


Figure 6.9.4 - Number of attacks during the active and passive phase for CHARGEN

QOTD

The QOTD honeypots saw a decline in the detected attack count much sooner than RIP and CHARGEN. After only 7 days, the detected attacks drops to zero. After that, there are only a few attacks detected on October 20th, and at the end of November. The attacks that were detected at the beginning of the passive phase have a significantly ($p < 0.01$) stronger preference for higher amplification honeypots than attacks after two weeks of the passive phase. The findings are similar to CHARGEN and may suggest that adversaries that select mainly high amplification honeypots for their attacks verify the availability of the vulnerable servers more often.



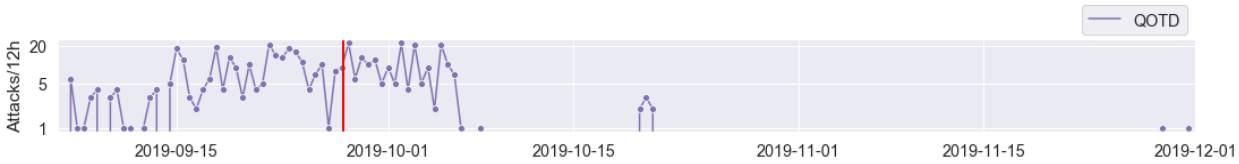
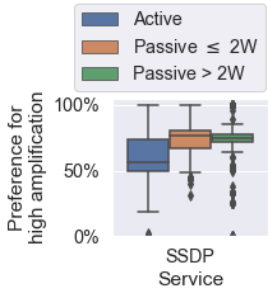


Figure 6.9.5 - Number of attacks during the active and passive phase for QOTD

SSDP

SSDP honeypots continued to detect new attacks until the beginning of December. This is longer than with any of the other protocols. The number of detected attacks kept decreasing after the active experiment concluded, but even after two months, there are still at least 10 daily attacks that are being detected. Although the attacks in the passive phase appear to have a slightly stronger preference for higher amplification honeypots that were used, there are many outliers and the differences are small.



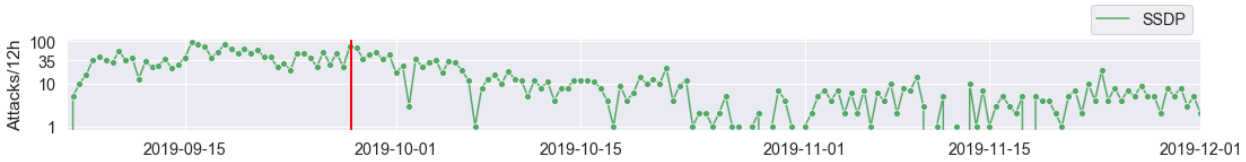
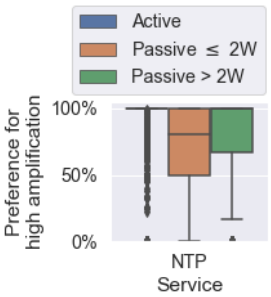


Figure 6.9.6 - Number of attacks during the active and passive phase for SSDP

NTP

NTP honeypots still detected peaks in the number of attacks weeks after the honeypots no longer provided responses. Only after the start of November, the number of detected attacks decreases to almost no attacks. One explanation could be the scanning activity of an adversary: adversaries might have defined scanning intervals where the adversaries verify that the vulnerable server still exists.

For NTP, the overall selection preference for higher amplification honeypots is significantly ($p < 0.01$) lower in the passive phase than in the active phase. This may suggest that adversaries that have a stronger preference for high amplification honeypots, verify more often that the servers they want to use in attacks are still available.



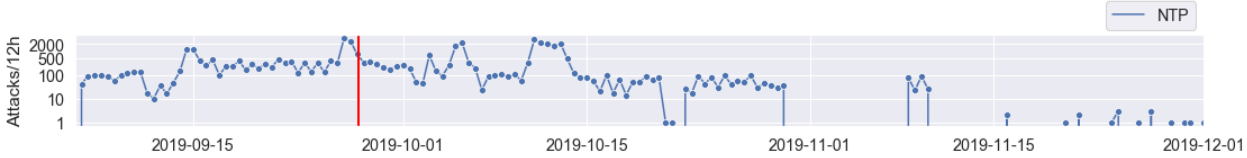
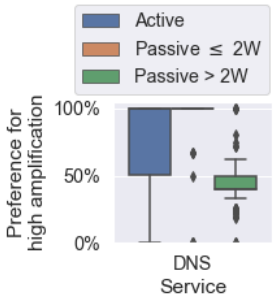


Figure 6.9.7 - Number of attacks during the active and passive phase for NTP

DNS

DNS sees the quickest decline in the detected attacks after the honeypots were put in passive mode. On the first day of the passive phase, no attacks were detected. Thereafter, there are only a few small peaks with attacks. The attacks that occurred in the first two weeks of the passive phase have a very strong preference ($p < 0.05$) for honeypots that provide a high amplification factor. For the attacks in the weeks thereafter, there is no preference for lower or higher amplification honeypots. This again suggests that more advanced adversaries that primarily select high amplification honeypots more often verify the availability of the vulnerable servers.



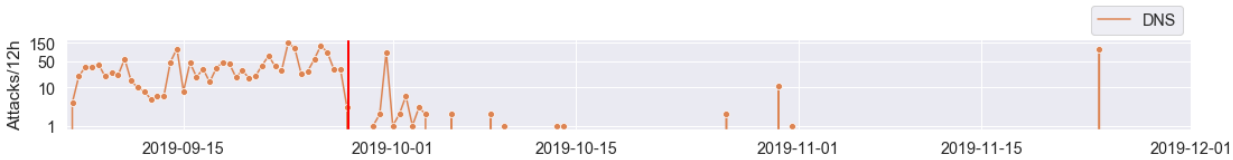


Figure 6.9.9 - Number of attacks during the active and passive phase for DNS

Conclusion

Honeypots for some services still detect attacks in passive mode after more than two months: this is primarily the case for RIP, CHARGEN, and SSDP. NTP sees a decline after a month. DNS and QOTD see a decline much sooner, after the first week hardly any attacks are detected anymore. Furthermore, the further in the retention period, the less strong the preference for high amplification honeypots becomes. This suggests that adversaries that have more stringent selection policies that select primarily higher amplification honeypots, also more often verify the availability of the vulnerable servers. Finally, the large number of high amplification biased CHARGEN attacks that were detected in the passive phase suggest that also for CHARGEN, high amplification honeypots will attract more attacks than lower amplification honeypots.

6.10. Further Developments

There are additional findings that were discovered during this thesis. These findings are not directly related to the research question and hypothesis of this thesis but may provide valuable information for future work. Thus, this last section may be skipped while reading this thesis. The findings are related to passive scanning analysis. The first subsection explains how and which data was gathered. The second subsection discusses ports that are popularly scanned. Finally, the last subsection discusses IP identification numbers in scans.

6.10.1. Scans

The first stage of exploiting vulnerable servers is to find vulnerable servers. Adversaries try to find these servers by scanning the Internet. Thus, there may be information that can be extracted by analyzing all these scans. However, the honeypots that were deployed on the VMs at the different cloud providers only gathered data specifically for the six services that were selected for the experiments. This means that information about scans on the other ports is not included in data gathered by the honeypots.

However, it was possible to run additional software on the VMs that the honeypots were running on. Therefore, during the experiment, additional software was deployed that captured all the incoming traffic on the VMs where the honeypots were deployed on. The traffic that is received by the honeypots is included in these captures.

Since packet logging facilities captured all packets, including attack traffic, there needs to be a distinction to what classifies as attack and what classifies as a scan. For this analysis, a data point is considered a scan if the average number of received packets among all ports that an IP address scanned that day, is less than or equal to 2. This ensures that only data points are included that sent one packet or those that included an additional second attempt of a port scan, for example, to compensate for a lost packet.

During this analysis, the scans are combined into hourly intervals to limit the number of data points and to provide a better insight into patterns over time. A scan is considered to be unique if it had a unique combination of a source IP and a destination IP. If the same combination is detected in a subsequent hour, it is considered to be a new scan. For each scan, the source/destination is stored together with the ports scanned and the number of scans of each of the ports, aggregated in the hourly intervals.

6.10.2. Scanned UDP Ports

Each scan contains information about the ports that are scanned. This means that it is possible to calculate in how many scans each of the different UDP ports were used. Table 6.10.1 shows an overview of the 25 most frequently scanned UDP ports. The table also includes a description of the usage of the port, and whether the service has known vulnerabilities that allows the service to be used in amplification attacks. All of the six services that are used in this thesis research are among the top 25 protocols - they are highlighted in bold font. Notably, NTP, SSDP, and DNS take the top three, if only services that provide amplification are considered. The legacy and deprecated protocols CHARGEN, QOTD, and RIPv1 are also all present in the most popular scanned UDP ports. This is interesting because although they have long been outdated, adversaries still actively scan for them.

This toplist shows a clear shift for many protocols in comparison to 2014 when Rossow [45] published a top 10 of the most frequently scanned UDP ports. At that time, both NTP and SSDP were not present in the top 10. CHARGEN was in second place as a service that could also be used for amplification but is now at place 10. DNS was already in the top 5. QOTD and RIPv1 did not make it to the top 10, so no comparison is available for these protocols. Although there are still some similarities, it is evident that during six years the landscape has changed. This means that it is important to frequently redo these experiments in order to keep the intelligence updated to the current state.

Table 6.10.1 - Toplist of scanned UDP ports

	Top 22 Scanned UDP Ports		
#	Port / Protocol	Amplification?	% of scans that included this port
1	5060 (SIP)	No	1.22%
2	**123 (NTP)**	Yes	0.54%
3	**1900 (SSDP)**	Yes	0.42%
4	**53 (DNS)**	Yes	0.40%
5	389 (LDAP)	Yes	0.35%
6	161 (SNMP)	Yes	0.29%
7	137 (NetBIOS)	Yes	0.23%
8	1434 (MSSQL)	Yes	0.18%
9	5353 (mDNS)	Yes	0.17%
10	**19 (CHARGEN)**	Yes	0.16%
11	53413 (Netis Router)	No	0.16%
12	111 (Portmap)	Yes	0.16%
13	10001 (Ubiquiti Router)	Yes	0.15%
14	623 (IPMI)	Yes	0.14%
15	**17 (QOTD)**	Yes	0.14%
16	11211 (Memcached)	Yes	0.13%
17	69 (TFTP)	Yes	0.13%
18	3283 (Apple RMS)	Yes	0.12%
19	5632 (pc Anywhere)	No	0.12%
20	5351 (NAT MPM)	No	0.11%
21	3702 (WS-Discovery)	Yes	0.11%
22	**520 (RIPv1)**	Yes	0.09%
23	1604 (Citrix WinFrame)	No	0.09%
24	1194 (OpenVPN)	No	0.09%
25	5093 (SPSS LM)	No	0.08%

6.10.3. IP IDs in scans

Each IPv4 packet has a unique ID. This ID is normally automatically generated and increased for each packet by the operating system. Since there are 2^{16} = 65,536 different IP IDs, the expected occurrence of each unique IP ID is $\frac{1}{2^{16}}$ or roughly 0.00153%. For this analysis, data is aggregated for each unique source IP and destination on a single day. This aggregation was used to analyze a scan on a higher level - thus, if an adversary scans one port in the morning and another in the afternoon, this is considered to be a single scan. To distinguish between scanning and attack data, a scan is defined as a recorded flow of data that, at maximum, used two packets towards the same destination port that day. This is a rather strict definition but prevents that we accidentally capture very small attacks in this data. There is no further analysis of the IP Identification data for attacks, as the IP Identification data in attacks provide no notable observations.

In total, 6,915,606 scans were recorded. Figure 6.10.1 shows a histogram of all the IP Identification values that the scanning data contained. This leads to several observations. First, there are two very clear peaks in this data. These peaks are caused by IP ID 54,321 and IP ID 0. IP ID 54,321 comprises 7.34% of all the scans. This IP ID is used specifically by ZMap [60] and likely by derived scanning tools. IP ID 0 comprises of 1.48% of the data, this may indicate that these scans originate from a machine running Linux with kernel version 2.4.x [61]. This Linux kernel is end-of-life since 2011 [62], this could mean that there are perhaps unpatched vulnerabilities on these servers that adversaries used to overtake the server, and that adversaries used these hijacked machines to do their scans.

Secondly, some ranges of IP Identification values have a higher frequency than other ranges. In this case, three distinct ranges can be extracted from the data. The first range starts at IP ID 1 and ends at 4,096 (2^{12}), the second range starts immediately thereafter and ends at 32,768 (2^{15}). The different ranges indicate that different generators, tools, or systems are used by adversaries to scan for vulnerable services. Finally, 80.58% of the scans scanned more than one port, but only 73,139 (1.06%) of the scans scanned more than one port using the same static IP ID. This is a surprising result because Krämert et al. [44] found that 90% of scanners scan only one protocol. The difference is that the scans in this work originated from honeypots that provided a response whereas the data from Krämert et al. used scanning data from darknets - networks that had no actual running hosts in the network and thus provided no response to any query.

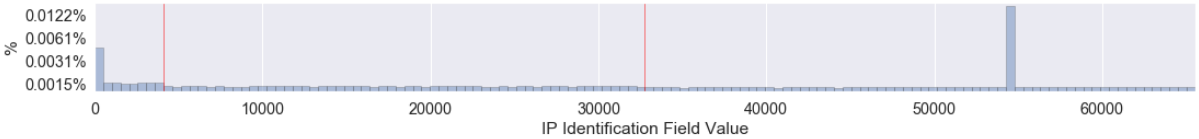


Figure 6.10.1 - Histogram of IP IDs with UDP scans

7

Evaluation & Conclusion

This chapter starts by concluding our research findings and providing an overview to answer the hypotheses. Thereafter the ethical considerations of the experiment are discussed. Section 7.3 discusses the limitations of this work. The section thereafter consists of proposals for future work. Finally, section 7.5 provides a short overview of the main contributions of this work.

7.1. Conclusions

In this section, we discuss the results from the previous chapters of this report, guided by the research questions and hypotheses that were discussed in section 4.1.

Hypothesis 1: "There will be a difference in abuse frequency between groups that provide a high amplification response and groups that provide a lower amplification response."

Hypothesis *H1* is accepted. Section 6.2 has shown that honeypots using a low amplification factor for the services QOTD, SSDP, NTP, and DNS are significantly less used in attacks than their configured counterparts with a higher amplification. Moreover, section 6.2.3 has shown that are distinct boundaries on the amplification size where an adversary does and does not use the server for attacks. For RIP, there was no conclusive evidence of these same patterns. For CHARGEN, there was also no conclusive evidence, although the retention showed that many attacks started occurring after the end of the experiment. Therefore, repeating the experiment may yield different results for CHARGEN.

There results also intuitively sound. When a service provides a higher amplification factor, an adversary that uses that service is able to create larger attacks with fewer resources. Thus, it is in the best interest of the adversary to use such selection strategies.

Furthermore, for QOTD, SSDP, NTP, and DNS different adversaries show a different preference towards the honeypots that were used in attacks with different amplification factors. Some adversaries had no preference for lower or higher amplification honeypots, some had a preference for higher amplification honeypots but still used (some) lower amplification honeypots, and finally, some adversaries used only the highest amplification honeypots.

For SSDP, NTP, and QOTD, there was not just an overall preference for higher amplification servers. The higher amplification honeypots also attracted larger attacks, where a higher number of honeypots was used for the attacks. For these same protocols, packets that in theory would elicit the highest response from the vulnerable servers were primarily used in the attacks that had a strong preference for high amplification honeypots.

Multi-vector attacks showed less of a preference for the higher-amplification services. However, there were different clusters of attacks that showed a different preference. Two suspected booter clusters showed a less strong preference for high amplification honeypots, whereas other multi-vector attacks showed a stronger preference for high amplification honeypots. Analysis of subnet attacks showed that these subnet attacks showed the strongest preference for high amplification honeypots, thus these attacks appear to be the work of more sophisticated adversaries.

Hypothesis 2a: "There will be a difference in abuse frequency between groups which have a fast network connection and a network which has a slow network connection."

Due to ethical considerations, there was an overarching rate limit that was applied to the majority of the attacks. The data showed that a larger share of attacks was influenced by the global limit than was anticipated. This meant that there was not enough reliable and unbiased information available to draw a statistically significant conclusion about the validity of the hypothesis.

Hypothesis 2b: "There will be a difference in abuse frequency between groups which have almost zero packet loss and a network which has a high amount of packet loss."

Hypothesis *H2.b* is accepted. For all the services that were used in this experiment, at least one of the groups noticed a significant difference in the number of attacks that were attracted. For SSDP, there was a difference noticeable in all the groups, and in addition, the number of daily attacks between the differently configured servers was significantly different. The conclusion is that there is definitely a difference in the number of attacks between low and high packet loss honeypots, but it depends on the service and configuration as to how large these differences are.

Analysis for *H1* showed that there are different adversaries that use different selection strategies. Similarly, the different results between honeypots that used different packet loss settings could also be caused by different adversarial selection strategies. A higher amount of packet loss would cause that an attack would become less effective, even if the services provided a higher amplification. Thus, it would not be surprising if more sophisticated adversaries are ignoring the vulnerable services that have a high amount of packet loss.

Hypothesis 2c: "There will be a difference in abuse frequency between groups which have a fast response time and a network which has a slow response time."

Hypothesis *H2.c* is rejected. There are no significant differences that show that adversaries have any preference for the response times of the honeypots. The different groups receive the same amount of attacks, and the attacks they receive do not show any different characteristics. Therefore, analysis shows that having a slower responding honeypot will not significantly influence the results. For an adversary, having a slower responding system will still yield an attack of an equal size and impact. Thus, it is not surprising that adversaries show no preference for systems that respond faster than others.

Hypothesis 3: "There is no difference in abuse frequency between groups in a network where each IP contains multiple vulnerable services compared to a network which only runs one vulnerable service per IP".

There has been no conclusive evidence to conclude on the validity of hypothesis *H3*. The available data for analysis is insufficient to reach a conclusion. For an adversary, the effectiveness of an attack is unchanged whether there are multiple vulnerable services on a single host or not. Similarly, even if running a lot of different honeypots on a single host could be an indication of a honeypot, an adversary is still anonymous when it uses the vulnerable services in an attack.

Hypothesis 4a: "There is no difference in abuse frequency between real responses and responses that indicate that the system is a honeypot."

Hypothesis *H4.a* is accepted. Adversaries do not show different behavior towards services that operated as an actual vulnerable system compared to a system that identified itself as a honeypot that is used for research purposes. In fact, the results even suggest that the fake honeypot responses were preferred. A possible explanation for this is is that these responses were static compared to the dynamic real responses. For SSDP, a trap to see which adversaries try to verify the authenticity also suggests that the only adversaries that follow up are merely looking for vulnerable UPnP servers, and not vulnerable servers to be used for DDoS attacks. A similar reasoning that explains these results can be applied that explains the results for hypothesis *H3*.

Hypothesis 4b: "There is no difference in abuse frequency between emulated services and actual implementations running in a virtualized container.".

Hypothesis *H4.b* is accepted. Adversaries do not show different behavior towards actual implementations of vulnerable services and emulated versions. Detectable differences can be explained by the difference in amplification that the systems provided. One of the concerns is that adversaries could detect a system being a honeypot and therefore, will ignore the system to prevent gathering intelligence of the adversary. Although it is always possible that an unknown external variable has influenced the research, with the (lack) of any evidence, we believe that an adversary is indifferent to the system being real or a honeypot. The adversary can use the servers in perceived anonymity and will use the servers - honeypot or not - as long as it contributes to its attack power.

Hypothesis 5a: "There is a difference between attacks that are detected among honeypots that are located at different geographical regions."

Hypothesis *H5.a* is accepted. There are differences in the attacks that can be detected in various regions. This means that it is important to distribute honeypots among different geographical regions in order to detect a larger share in the attack landscape. However, depending on the service, a smaller or larger share can be detected by a specific region. Overall, honeypots that were located in Europe seem to have been able to detect more of the attacks compared to Asia and Australia. These apparent selection preferences of adversaries could have a multitude of reasons. For example, some adversaries may perceive that certain locations have better overall connectivity and prefer these locations.

Hypothesis 5b: "There is a difference between attacks that are detected among honeypots that are located at different providers."

Hypothesis *H5.b* is accepted. Different providers detect different victims and attacks. There is no 'best choice' for a provider that is able to detect a majority of all attacks, although honeypots running at some providers do appear to detect more. In general, the findings suggest that it is best to distribute a honeypot network over a variety of providers to ensure that there is no data collection bias that is related to a provider. Similar to *H5.a*, adversaries may prefer one provider over another as different provides use different security measurements. Furthermore, each provider may attract certain types of hosted services that may be particularly of interest.

7.2. Ethical Considerations

During the operational phase of the thesis research, the deployed honeypots participate in several attacks. In addition, the number of open servers may influence trends which are generated by other research projects. However, it is believed that this research will provide valuable insight into the strategies employed by adversaries - and that there is no other way to gather such information.

To ensure the impact of our research is kept to a minimum for both cases, we used different strategies. The first strategy is rate-limiting, and the second strategy is blacklisting. Blacklisting is used to discard any incoming packet that originates from sources that are on our blacklist. The used blacklist included many known research projects. During the initial data collection phase, we have gathered newly found scanners and research projects to also exclude them before starting the active phase of the experiment.

Rate limiting was used on both a per-agent basis as well as on a global basis. Locally, for each incoming IP address, the allowed throughput was at maximum 0.5 Mbit/s with a 2.5 Mbit/s burst rate or 35000 packets/s - whichever was lowest. Globally, all connected agents to the network were allowed to send out a combined maximum throughput of 20 Mbit/s or 10000 packets/s to any target. When the global is reached, all agents received proportional rate limits to ensure the global limit would not be exceeded. Considering the size of many attacks on the Internet, the imposed rate limits would ensure that our honeypots will not significantly contribute to these attacks.

7.3. Limitations

While working on this thesis, and during analysis, several limitations arose. These limitations are discussed in this section, and the limitations include proposals for future work.

Packet Sampling

The method of packet sampling in the agent was suboptimal for deeper packet analysis. During the experiment, the first, last, and every 100th received packet was stored. The reasoning for this was that it was unfeasible to simply count the

occurrence of each packet - as each received packet could be different. However, as this thesis shows, most attacks only use a single packet repeatedly. An improved strategy would be to count the frequency of the different received packets. If more than a certain amount of unique packets are found, a fallback on the original sampling strategy could be used.

Attack Information

Each registered flow contains some statistics of the flow. However, these statistics only described the total flow. More real-time information was also available for the rate limiter, but this information was not stored. This prevents a deeper analysis of the behavior of an attack over time. Thus, to analyze how the attacks evolve over time, more sampled flow data could be saved.

Duration

Due to resource limitations, the experiment was only able to run in the active mode between September 6th and September 27th. However, as shown in the retention section, many attacks occurred after September 27th. For some services, there are even more attacks after the 27th than before. As these attacks are excluded in analysis, this may influence some of the findings of this thesis, for example, statistics about the popularity of the different protocols. One of the main limitations of this research is thus the duration of the experiment. If the experiment is to be repeated for more data gathering, the recommendation would be to focus less on the number of available honeypots and to focus more on running the available honeypots for a longer duration.

Content of Responses of Honeypots

During the work of this thesis, it appeared a dynamic versus a static content might be of influence on the results. However, this was not a hypothesis and could not be further tested with the available data. Thus, the dynamic or static nature of the response could be tested in future work.

Rate Limit

The rate limit sub experiment has not provided a conclusion. This is because the main experiment could not properly be compared to the rate limit experiment. For a proper comparison, attacks that were not rate-limited should be compared to the attacks that did receive rate limiting. Due to ethical considerations and the size of the experiment, choices were made that prevented this comparison as the main experiment also had a form of rate-limiting (so be it less strict). For future work, a small set of both sub-experiments could be used that impose much higher rate limits for the main experiment. Since the smallest - depending on the protocol - only requires a minimal number of honeypots to cover most of the attack landscape, its total impact will be lower.

Honeypot Design

Some of the packet handlers for the different types of services in the MD-Honeypot were designed to always return a response, irrespective of the received packet. Although it appears that this has not been of major influence during the experiment, it would be better to investigate several different implementations of a protocol and make a deliberate choice to do similar packet validation or not, before returning a response.

7.4. Future Work

The findings of this thesis provide a foundation for future research. There also have been additional findings that were not related to the research question of this thesis, but these findings were discovered while analyzing the data.

SSDP Vulnerabilities

This thesis discovered that there are adversaries that scan for vulnerable SSDP services. These adversaries seemingly did not do so to use these vulnerable services for DRDoS attacks, but for other reasons. These adversaries follow the description files of SSDP to the actual server and seem to be triggered by the type of system or service that runs the SSDP service. Future work can do further research as to what these adversaries are looking for and study the behavior of these adversaries.

Location of the Honeypots

The convergence method showed that although only a limited amount of honeypots are required to measure (most) of the attacks in the attack space, there are certainly differences noticeable between both providers and locations where the honeypots are hosted. Although it appears that with the spread of providers and locations that were used in this thesis, many attacks were detected, it provides an interesting question for follow-up research. Are there more exotic providers or locations that detect a different set of attacks? Furthermore, if this is the case, can a reason for this difference be found?

A Foundation for Attractive Honeypots & DDoS detection with the MD-HoneyPot Framework

Honeypots can be used by researchers and security specialists to gather valuable information. With the research question of this thesis answered, more information is available that allows us to design more attractive honeypots. The information that this work provides also allows us to selectively target different types of adversaries. These tools may assist researchers in future work to gather data specifically for their needs. Section 7.5 contains a summary of these contributions.

The final section of the introduction outlined a proposed anti-DDoS system that can be built by extending upon the work of this thesis. This work contributes to such a solution by:

- Providing a framework, the MD-HoneyPot Framework that contains a central data collection unit that controls flows and processes real-time flow information. This framework can be extended to produce real-time information about ongoing attacks that should be blocked.
- Providing information that helps to build attractive honeypots. These attractive honeypots are a key element in capturing valuable real-time data to make such a system work effectively.

7.5. Summary of Contributions

The main contributions of this work are the following:

- We have shown that the amplification factor of a service is a driving choice for an adversary that determines whether an adversary does or does not use a vulnerable server.
- We have shown that a delay in responses has no effect on the use of honeypots in attacks.
- We have shown that packet loss has an effect on the use of honeypots in attacks.
- We have shown that adversaries will also readily use vulnerable systems that are clearly honeypots, in contrast to TCP services.
- We have shown that adversaries are indifferent to systems being emulated/fake and will use them whenever they are usable in attacks.
- We have shown that the geographical region where the honeypots are deployed is of influence on the attacks that are detected.
- We have shown that the provider where the honeypots are deployed is of influence on the attacks that are detected.
- We have created a honeypot framework that can be further developed into a fully-fledged DRDoS detection and prevention system

Appendix A - MD-Honeypot Component Details

This appendix discusses all the different components of the MD-honeypot framework which were introduced in chapter 5. For each component, first, an outline of the functionality is given. This is followed by the technical details of the component. The first component that is discussed in this appendix is the configuration server. This is followed by the dashboard server, the logging server, the UDP flow controller, and the agent.

A.1. Configuration Server

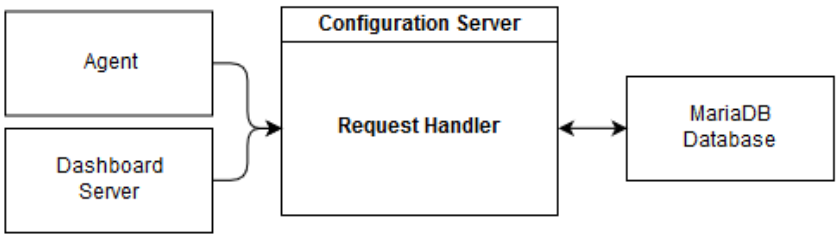


Figure A.1 - Schematic overview of the configuration server

The configuration server is running in Python and provides a variety of functions. Its role is to provide the agents with configuration files. A configuration is dynamically determined upon the first connection and reused when the same agent reconnects. All the options the configuration server provides will be explained according to the endpoints it provides:

Endpoint: /get_my_ip
The get_my_ip endpoint simply returns the IP from the connecting agent. This is useful for getting a running agent its public IP while running behind a network that uses Network Address Translation (NAT).

Endpoint: /connect_agent
When an agent starts up, and the configuration server settings are enabled, it will connect to this endpoint, providing its local configuration options. For example, an agent can specify which services it will accept to run. The server will then generate a new configuration file or returns a previous one if it is a reconnect. The configuration server automatically distributes the different connecting agents among different groups based on the available options and their weights.

Endpoint: /heartbeat_agent
Active agents will frequently send a heartbeat towards the server. This is useful for keeping an up to date state of the network, and will also help to normalize the gathered data. For example, if an agent would not be online for 24 hours, it is expected that it also did not gather data for 24 hours.

Endpoint: /test_connection
After startup, an agent can run connection tests to test whether the configured ports and services are reachable on its interface. Agent hosts may not always be reachable publicly due to network address translation (NAT), often found in businesses and homes. Firewalls could also block connections.

Endpoint: /get_active_agents
To provide an overview in the statistics dashboard, the configuration server can output a list with currently active agents.

Endpoint: /blacklist
During deployment of the agent, a blacklist file is included. This endpoint provides a method to dynamically add entries to the blacklist after deployment.

A.2. Dashboard Server

The dashboard server provides a static webpage with an overview of logged data. Figures A.2 and A.3 show a graphical representation of the webpage. On the top left side is a 3D model of the globe with lines between the source and destination countries. The more connections a country has in total, the faster and more moving bullets it has. On the top right is this

same data represented in a heatmap of countries. In the row below are the total connection and packet statistics for all deployed protocols.

Next is a table that shows the latest incoming connections for the IP address that is also viewing the dashboard. For each connection, the following information is displayed: the source IP, source port, destination IP, destination port, protocol, service (UDP or TCP), the number of packets in the connection, and the time in UTC when the connection was first started. A system administrator may login to view an overview based on all the logged data in the database.

The visual framework is server by the dashboard server, and the data is then dynamically loaded from the logging server. While having the dashboard open, the entries are automatically refreshed every minute.

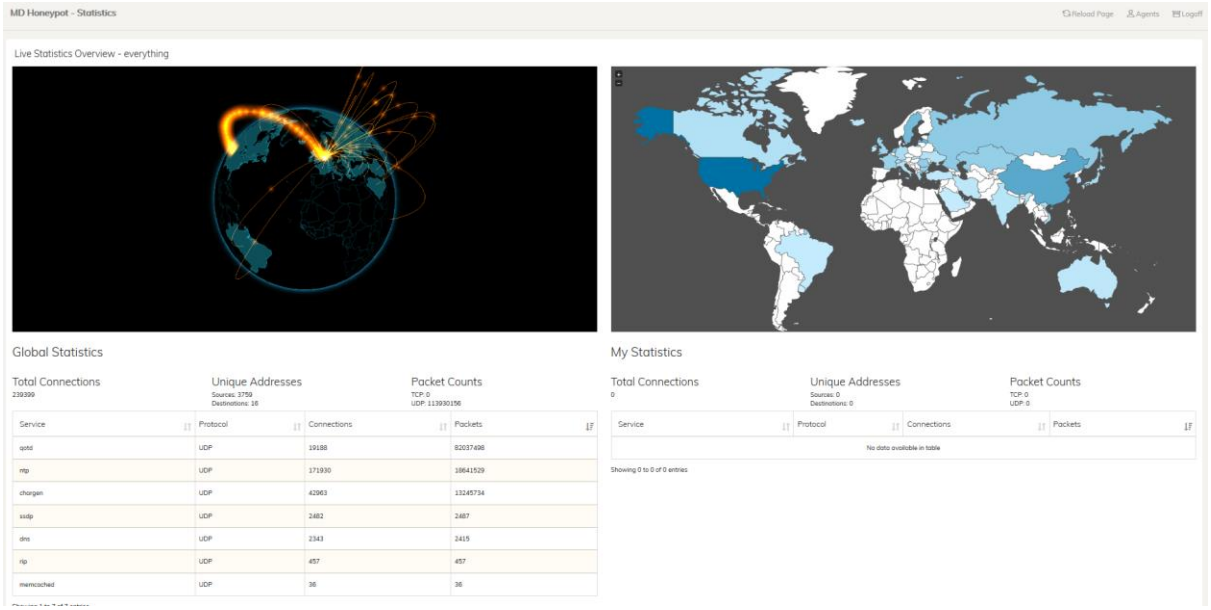
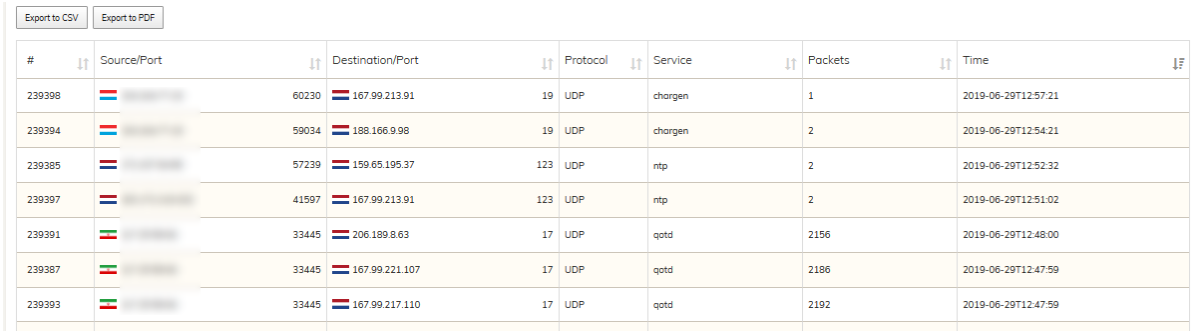


Figure A.2 - MD-honeypot dashboard overview



#	Source/Port	Destination/Port	Protocol	Service	Packets	Time
239398	60230	167.99.213.91	19 UDP	chargen	1	2019-06-29T12:57:21
239394	59034	188.166.9.98	19 UDP	chargen	2	2019-06-29T12:54:21
239385	57239	159.65.195.37	123 UDP	ntp	2	2019-06-29T12:52:32
239397	41597	167.99.213.91	123 UDP	ntp	2	2019-06-29T12:51:02
239391	33445	206.189.8.63	17 UDP	qotd	2156	2019-06-29T12:48:00
239387	33445	167.99.221.107	17 UDP	qotd	2186	2019-06-29T12:47:59
239393	33445	167.99.217.110	17 UDP	qotd	2192	2019-06-29T12:47:59

Figure A.3 - MD-honeypot dashboard connection log

A.3. Logging Server

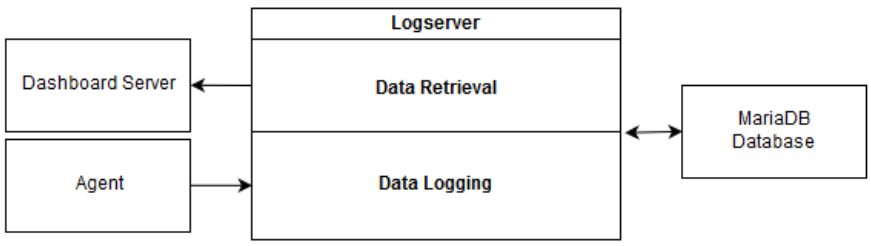


Figure A.4 - Schematic overview of the logging server

The logging server consists of two different components. The first component listens for incoming logging data. The second component handles requests to retrieve (aggregated) data. This data retrieval is currently used for the dashboard but could be extended for other purposes. The endpoints and the functionality of both components are now explained.

Logging Component

Endpoint: /status
The status endpoints merely returns back an 'OK'. It is used to test the initial connection to the logging server.

Endpoint: /add-admin, revoke-admin, add-trusted, revoke-trusted
These endpoints are used for the management of the logging system. An admin may add or revoke trusted sources. The data of trusted sources is stored in a different database that the data of untrusted systems. This distinction is used as additional data integrity checking may be used on data coming from external servers that are not in direct control of the research group.

Endpoint: /batch-insert-v2
An agent uses this endpoint to send data to the logging server. To reduce load and overhead, updates are sent in batches of data. As such, each batch insert may contain one or more entries to be logged in the database. The logger allows for three different types of data objects to be logged:

- Messages: it is cumbersome to continuously check the status of all the agents and view their log files to verify that no critical errors have occurred. For this reason, an agent may send out messages that need to be logged globally. In general, only logging entries with a severity of critical or higher are sent to the logging server. The entries are stored with four properties in total: the severity, the service it applied to, the message, and the agent which logged the entry.
- Packets: individual packets can be logged. This feature is not used for the UDP protocol as individual packets of a DDoS flow are generally the same. Fields included here are the source IP and port, the destination IP and port, the IP protocol, the application protocol, the handler which was handling the connection, a timestamp, a payload, and finally, the logger IP for verification purposes.
- Connections: for each connection, the following data is logged: the service (application protocol), source IP and port, destination IP and port, start time of the flow, end time of the flow, total packet count, a data field*, whether the connection was blacklisted, the blacklist reason and finally the logger IP.
- For the purpose of quicker data analysis, the country of the source and destination address are also saved and are automatically looked up by the logging server. It makes use of the Maxmind GeoLite2[29] database to lookup the country belonging to the IP address. In addition, the logging server queries a reverse lookup on the source IP addresses. This also provides for a quick analysis of the originating addresses (or victims). It also helps to detect new scanner services used for research so these addresses can be added to the blacklist.

  * the data field will be discussed in more detail in the description of the agent component.

In each case, some data sanitization is automatically done before inserted any data in the database. This happens for both the trusted and untrusted database. The data sanitization that is currently used checks whether all required fields are present in the request. The destination IP addresses of the packets and connections are also verified. If an address is either a private Internet address in the rfc1918 address space, or is the locally bound interface 0.0.0.0, the address is replaced by the IP address of the logger. In these cases, the agent was running locally in an internal network with no public address available. As such, it is assumed that the actual public Internet address, which was used to gather the data is equal to the Internet address the agent uses to report the logging data.

Retrieving Data Component

A part of the endpoints below is protected and requires valid credentials to be queried. The endpoints which require valid credentials are marked with an asterisk (*). Unless stated otherwise, all requests and responses are in JSON format.

Endpoint: /get_last_messages*
This endpoint returns the last messages which were received by the agents.

[29] https://dev.maxmind.com/geoip/geoip2/geolite2/

Endpoint: /test_auth*

This endpoint can be used to test the authentication with the provided credentials. This is used by the dashboard to verify a login. If the credentials are valid, a valid JSON is returned, indicating a successful login. If the login is invalid, it will serve a 403 Forbidden header.

Endpoint: /stats_summaries, /stats_summaries_all

This endpoint provides a list with a few statistics on all the logged connections. It shows the aggregated amount of connections and the number of packets per protocol. The default endpoint filters the statistics to only include connections targeted at the host, which requested the statistics. The latter endpoint does not apply such a filter.

Endpoint: /stats_country_get, /stats_coungry_get_all*

This endpoint provides a list with the total amount of connections, aggregated by the source country. The default endpoint filters on the destination IP of the host, which requested the statistics. The latter does not apply such a filter.

Endpoint: /stats_export_to_csv, /stats_export_to_csv_all*, /stats_export_to_pdf, /stats_export_to_pdf_all*

As described above, but returns the data as CSV or PDF download.

Endpoint: /stats_connections_get, /stats_connections_get_all*

Returns a list of connections that are logged in the database. The same fields which are logged are returned, with the exception of the data field. The request format uses the format produced by the Datatables ajax request. The all method does not apply IP filtering as in the endpoints described above.

A.4. UDP Flow Controller

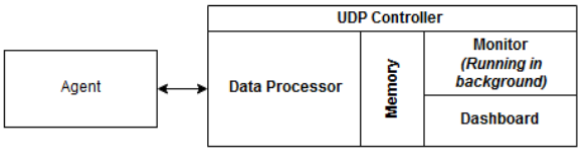


Figure A.5 - Schematic overview of the UDP Flow controller

The UDP flow controller consists of two components: a front-end dashboard and the flow controller. The dashboard can be used to access the current state of the system with a web browser and provides an overview of all the currently active flows. Figure A.6 shows the dashboards that shows the current incoming and outgoing connections.

Current Flows

Currently Active Flows, No Historic Data

Search:

Source IP	Connection Count	Packets/sec	Bytes/sec	Packets	In	Out	Bytes	In	Out	Servers
SOURCE.IP	16	16	16.88 kbit/s	16	16	0	2.11 KB	2.11 KB	0 Bytes	[redacted]->145.94.137.12 0s [redacted]->145.94.79.170 0s [redacted]->145.94.126.107 0s [redacted]->145.94.24.168 0s [redacted]->145.94.72.61 0s [redacted]->145.94.119.108 0s [redacted]->145.94.137.77 0s [redacted]->145.94.119.126 0s [redacted]->145.94.122.57 0s [redacted]->145.94.131.155 0s [redacted]->145.94.7.236 0s [redacted]->145.94.137.142 0s [redacted]->145.94.67.82 0s [redacted]->145.94.93.82 0s [redacted]->145.94.85.123 0s [redacted]->145.94.119.102 0s
SOURCE.IP	12	12	13.22 kbit/s	12	12	0	1.65 KB	1.65 KB	0 Bytes	[redacted]->145.94.67.66 0s [redacted]->145.94.89.145 0s [redacted]->145.94.24.81 0s

Figure A.6 - Flow controller dashboard

The flow controller accepts flow updates and provides information about source addresses that should be rate limited. It frequently updates the total flow to and from each source address. It has three endpoints:

Endpoint: /flow_update

Each agent submits all current connections it has open to the UDP flow controller at regular intervals. It also provides current flow metrics. The UDP Controller continuously calculates the total flow from each of the unique source address it has registered. Once the total combined flow throughput reaches a threshold, the defined rate limit is divided by the number of flows receiving packets from the source and is then applied. The threshold is used to prevent a short spike from directly rate-limiting all current connections at a speed that is noticeable to an adversary. Besides a regular update interval, the agent also contains additional logic, so it will update its current flows quicker in a new flow is started. This ensures that the rate-limiting can be imposed much quicker.

Endpoint: /flow_status

A connecting agent may request the flow status. The flow status returns an object with all source addresses, which should be rate-limited, and it also provides the current rate limits which are imposed on each individual entry. These values may change over time, which is why an agent should frequently reload the flow status and update settings it receives by the endpoint.

Endpoint: /flow_settings:

The flow settings endpoint is used to dynamically adjust the thresholds and rates limits to impose on all the flows. Using this method prevents the server from needing a reboot in case the other settings need to be used. When restarting the server, all intermediate data is lost as it is only saved in memory.

A.5. Agent

The agent is the most complex part of the system. This paragraph discusses the general structure of the agent and its typical use case and startup procedures. The different configuration possibilities are discussed in Appendix B.

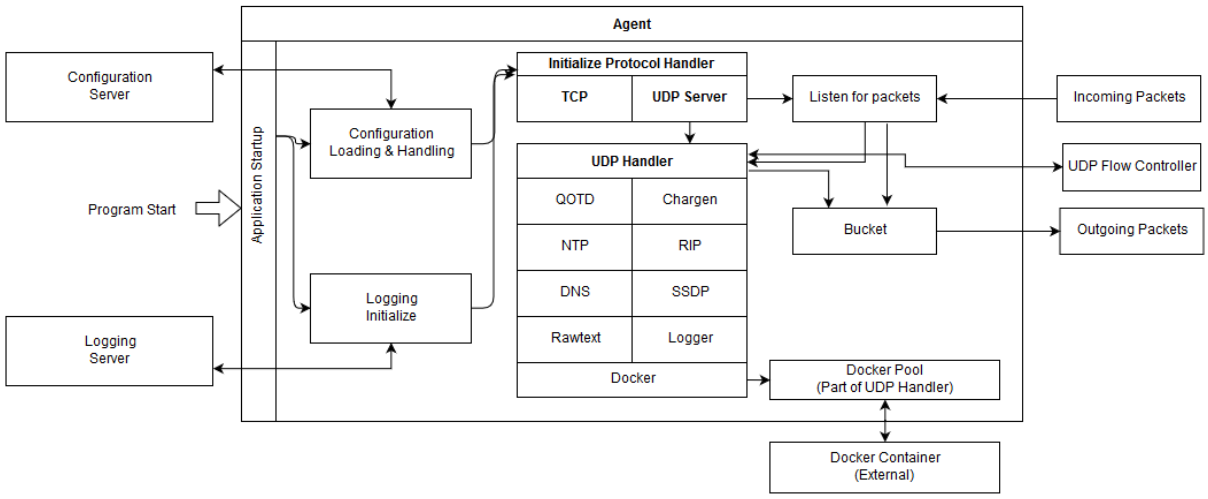


Figure A.7 - Schematic overview of the agent

The application root is the MD-Honeypot and is always started with a configuration file. The first steps involve reading the configuration files. If a remote configuration server is specified, it will connect to the specified configuration server and overwrites its config by the configuration it received. During this stage, the systems network interface and available interface addresses are also detected. Finally, the blacklists are loaded.

Once the configuration is fully loaded, connection tests are run if required. Services that are not reachable or have other startup issues, such as a port in use, are dropped from the active configuration. The second last step of the initialization includes starting up the weblogger - which is used for logging messages, packets, and connections to the remote logging server. If raw packet capturing is enabled, this is also started at this point. Finally, the system moves on to the actual service initialization.

For each service, the correct (network) protocol handler is determined. MD-Honeypot currently supports both TCP and UDP, but this thesis only focuses on UDP. To read up on the details and implementation of TCP, the work presented by colleagues Bart de Jonge and Ahmet Gudek may be consulted.

A flow dashboard is available if one or more UDP services are defined, and the flow dashboard feature is enabled in the configuration settings. The dashboard provides an overview of all currently active flows and the flow metrics of those flows. This feature is disabled by default but is very useful for debugging. The dashboard is comparable to the dashboard, which is available for the UDP flow controller but contains a bit more metrics, and the data is real-time.

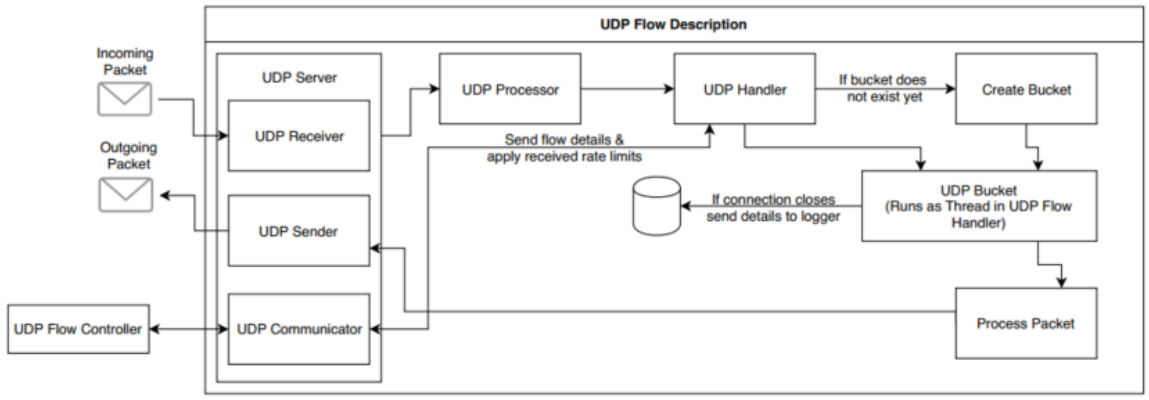


Figure A.8 - UDP Flow Description

For UDP, the corresponding network protocol handler is the UDP_Server. Each UDP_Server consists of the following four different components: a UDP_Sender, UDP_Receiver, UDP_Processor, and finally, the UDPHandler. For each address that is configured, one UDP_Receiver and one UDP_Sender is assigned. All the receivers share a queue. Once a packet is received, the receiver preprocesses the data and puts it in the incoming queue. The processor runs in another thread and calls a handler function on each of the packets. Later, when a packet has been processed, the handler can put a packet in the send queue. The sender will then pick up this packet and sent the packet.

Each UDP Handler is responsible for handling all the packets of a certain protocol. For this thesis, that means there may be up to six UDP Handlers running at the same time. The UDP Handler maintains control of individual flows and determines the correct packet handler for each connection.

Once a packet comes in, the UDP Handler checks whether a flow of the given source address already exists. If this is not the case, then a new UDP Flow Handler and UDP Bucket are created. In the other case, the packet is added to the bucket of the corresponding flow. The UDP Handler will also create a new packet handler and assign it to the UDP Flow Handler. The handler is determined in the configuration and will never change during runtime. It is straightforward to extend the system to incorporate new handlers. The currently supported handlers can be found in table A.1.

Table A.1 - MD-Honeypot UDP Handlers

Handler Setting	Handler	Description
logger	PacketHandlerLogger	Simply drops the packet.
emulate_qotd	PacketHandlerEmulateQOTD	The QOTD emulator loads a random quote from memory and sends it back to the client, which requested the quote. The behavior of the emulator is determined by the provided quote data file, which contains the quotes.
emulate_chargen	PacketHandlerEmulateCharGen	The CHARGEN emulator works on the mode specified in the configuration file. In the short mode, it generates a shifted string of ASCII characters. In the long mode, it returns a much larger static string of data. In the fake mode, it will return with a predetermined fake string.
emulate_ntp	PacketHandlerNTP	The NTP emulator will check whether the received packet is a monlist packet. If it is, it will respond with a monlist reply packet

		followed by data which is provided by the configuration file. If the packet was not a monlist packet, the packet is parsed by an internal NTP parser and will behave like a regular NTP server.
emulate_dns	PacketHandlerDNS	The DNS emulator will pretend that the request was for example.com and respond based on the provided example zone file.
emulate_ssdp	PacketHandlerSSDP	The SSDP emulator will respond to any incoming packet, regardless of the validity of the request. The return data is specified in the configuration. The emulator replaces some values of the specified data to generate dynamic data. This includes the time, a timestamp, and the IP address of the server.
emulate_rip	PacketHandlerRIP	The RIP emulator runs in predefined modes. The server will always respond on a packet, even if the request was invalid. The included modes are: short, which returns a small routing table, large, which returns a large routing table, and fake, which returns a text byte string indicating the server is a honeypot.
emulate_rawtext	PacketHandlerRawText	The raw text emulator returns any data which is loaded as specified by the configuration file. It may be used if no processing of the incoming query string is required, and the returned data is always static.
docker	PacketHandlerDocker	The docker handler is the most complex handler. When a new connection is started, the handler will query a docker pool and will return an IP address to which the packets should be forwarded. It opens a socket from which it will send these packets. The packets are then processed by a service that is running in a virtualized container. Once a reply is received by the socket, it knows to whom to send the packet back as a new socket is opened for each individual source connection. Once the connection is closed, the socket is also closed. The docker pool is discussed separately.

The UDP Bucket is used to impose a rate-limiting. The rate-limiting can be abstracted as follows: each connection is assigned a leaky bucket. This means that there is a maximum rate on which packets can flow out of the bucket. The input rate of the bucket may vary, but a fixed output rate is guaranteed. Once the bucket is full and new packets are added, it will overflow, and the overflowing packets are dropped. Figure A.9 shows a schematic overview of this process.

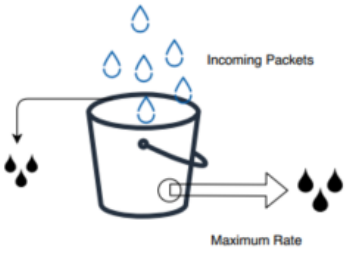


Figure A.9 - Bucket Rate Limiting

The implementation of this system uses tokens. A bucket is considered empty when all tokens are available. Each leaving packet will cost a token, and as soon as there are no more tokens, no new packet may exit the bucket. Meanwhile, tokens automatically refill at a fixed rate. As soon as the packets do not enter the system at a higher rate than the refill rate of the tokens, no rate limiting is essentially applied. The refilling of tokens will only continue up to a certain amount, called the burst rate. The burst rate is used so that a new connection can use a bit more than the average maximum rate for a short duration. For example, assume the following scenario: The bucket contains 50 tokens (50 is the burst rate). An adversary starts to send 20 packets per second. The refill rate is 10 tokens per second (10 is the regular rate limit). In the first five seconds, 20 packets are sent back. However, after five seconds, the available tokens are zero, and the refilling only happens at a rate of 10 tokens per second. As such, after five seconds, the connection is effectively rate limited to 10 packets per second.

The rate limiter in MD-Honeypot does not only rate limit on the number of packets per second but also on the maximum transmitted bytes per second. It checks both the available tokens for the packets and the bytes and uses whichever one is lower to check whether a packet may be released from the bucket. To summarize, it uses whichever limit it reaches first and applies that limit. However, since the application can only see the payload size and not the actual amount of transmitted bytes, 42 extra bytes are added in the calculation for each packet. This exactly matches the size of a UDP and IPv4 header.

The UDP Bucket also has support for simple filters to introduce packet delay and packet loss. These filters can be applied dynamically if deemed necessary. Each filter can be applied to only incoming traffic, only outgoing traffic, or both. For packet loss, the percentage of dropped packets can be specified, and for packet delay, a percentage for which the filter should be applied. The minimum and maximum delay can also be specified in milliseconds. Because historical data is not stored to determine whether a value will be applied or not, a random number is used to determine whether to apply the filter or not. By the law of large numbers, this will lead to an average delay or drops equal to the configured option. The first packet in a connection is never dropped.

The UDP Flow Handler runs in a separate thread and will retrieve packets from its bucket. It has a configurable timeout, but the default timeout value is 60 seconds. If no new packet has entered the bucket for longer than the timeout value, the connection is considered closed. Note that this is required, as UDP is a connectionless protocol. When the connection is closed, the connection data is logged to the logging server, and the bucket and flow handler object are destroyed If the logging server is temporarily unavailable, the log entries are kept in memory, and the logging action is retried later. There is no maximum backlog size for the logging entries. This strategy is chosen so that even in the case of a logging server crash, no data is lost, even if the log server is restarted a day later.

The data sent to the log server includes the following information: the service protocol, the network protocol (UDP), the source IP and port, the destination IP and port (the interface and port where the packets were received), the start time of the flow, the end time of the flow, the total amount of packets in the flow, a data object, and whether the connection was blacklisted. If the connection was blacklisted, a reason is also provided.

The flow data object contains much more additional data related to the flow and is stored in a JSON format. The entries that are stored in the flow data object are listed in table A.2.

Table A.2 - Properties of flow information

Key	**Purpose. * Items marked with an asterisk are mainly used for real-time flow statistics.**
pkts_dropped_full_queue	The amount of packets that were dropped because the queue size of the bucket was not large enough. This may happen if packets are coming in much faster than the rate limit.
pkts_rate	The average packet rate of the flow.
pkts_total	The total amount of packets of the flow.
pkts_total_in	The total amount of received packets.
pkts_total_out	The total amount of transmitted packets.
bytes_rate	The average bandwidth of the flow.
bytes_total	The total amount of transferred bytes of the flow.
bytes_total_in	The total amount of received bytes.
bytes_total_out	The total amount of transmitted bytes.
rr_pkts_rate_in*	The average amount of received packets based on the last 50 packets.
rr_pkts_rate_out*	The average amount of transmitted packets based on the last 50 packets.
rr_bytes_rate_in*	The average amount of received bytes based on the last 50 packets.
rr_bytes_rate_out*	The average amount of transmitted bytes based on the last 50 packets.
rr_pkts_sec_in_min	The lowest received packet rate in the flow.
rr_pkts_sec_in_max	The highest received packet rate in the flow.
rr_pkts_sec_out_min	The lowest transmitted packet rate in the flow.
rr_pkts_sec_out_max	The highest transmitted packet rate in the flow.

rr_bytes_sec_in_min	The lowest received byte throughput rate in the flow.
rr_bytes_sec_in_max	The highest received byte throughput rate in the flow.
rr_bytes_sec_out_min	The lowest transmitted byte throughput rate in the flow.
rr_bytes_sec_out_max	The highest transmitted byte throughput rate in the flow.
time_diff	The total duration in (fractional) seconds of the flow. Measured as the time of the last packet minus the time of the first received packet.
time_start	Start time in epoch seconds.
time_end	End time in epoch seconds.
packet_samples	A sample of the received packet payload. The first received packet is always included. The last packet is included if the flow had more than 1 packet. Every 100th packet is also included.

A dockerpool manages the interface between a handler and the docker infrastructure. When a docker pool is started, it checks if there are already existing containers on the system. It will update its index with the detected containers and will attempt to restart them if their status is not online. The dockerpool managers a pool with available containers to ensure a new container is directly available when a new adversary connects. As soon as the available containers in a pool drop below a certain size, it will automatically create new containers.

The system also ensures that the required images for the containers are automatically pulled from docker if the image does not exist on the system. The docker pool system can be operated in different modes. By default, for each connecting adversary towards each individual destination, a separate container will be created. The generalize_source and generalize_destination options provide a way to change this mapping. By generalizing the source, all adversaries connecting to the same destination IP will connect to the same container. With this setting, the system behaves like a normal service available on a single address. The generalize_destination setting determines whether all destination IP's should be considered the same. This means that, if a system has multiple addresses on multiple interfaces, it would not matter to which one an adversary connects, they will end up in the same container. With these two options, four different running configurations can be achieved.

Appendix B - MD-Honeypot Configuration

Included below is an example configuration file containing all the options which can be configured. Only settings related to UDP are discussed. Features related to TCP are not discussed. These details, such as details on multiplexing containers and sub-services, are discussed in the work of my colleagues.

The configuration is shown on a block-per-block basis and then discussed. The description of a block is placed under the example settings in each case.

```
include = ["/assets/configs/live/_udp_live.toml"]
```

The include directive may be used to include another configuration file inside the loaded configuration. Includes can be used recursively. Once an include directive is found, it will first continue loading the include before processing the rest of the configuration. As such, overwriting included settings is possible.

```
title = 'MD-Honeypot UDP-DEV Config'
```

A title is optional but can be set to identify the configuration file.

```
[upnp]
enabled=false
```

UPnP can be used to open ports of the Internet routing automatically. This is usually used in networks where NAT is used. Because of the nature of this thesis, this feature is disabled as the system only runs on publicly facing networks with high available bandwidth.

```
[logging]
console_level = 'VERBOSE'
telegram_level = 'CRITICAL'
server_level = 'ERROR'
telegram_logging_enabled = true
server_logging_enabled = true
```

Different log levels may be specified for the different logging targets. The console-level determines which information is displayed in the console of the running system. The Telegram channel is used to forward messages directly to a Telegram chat in case of a serious issue, such as a crash. The server-level determines which type of message is sent to the logging server. Finally, the external dependencies of logging to Telegram and the logging server can be enabled or disabled.

```
[[logging.files]]
level = 'VERBOSE'
file = '~/md_honeypot/log/verbose.log'
```

Log messages may also be logged to a file. For each log level, a separate log file location can be specified.

```
[logging.server]
address = 'log-data.md-honeypot.nl'
port = 5003
enabled = true
delay = 1 # Delay in seconds between batch logging.

[logging.keys]
skey = "/assets/agent/keys/sig.key"
vkey = "/assets/agent/keys/ver.key"
hmackey = "/assets/agent/keys/hmac.sym"
```

The logging server address and port can be specified. It can also temporarily be disabled. Finally, a delay can be specified. Logging items such as messages, packets, and connections are sent in batches. The delay determines the interval in which

the log batches are transmitted to the server. The keys are used and are embedded by default. Trusted keys may be supplied to enter data in the trusted database directly.

```
[config_server]
enabled = false
address = 'https://conf-srv.md-honeypot.nl' # ip or url
port = 443
```

The configuration server is optional. If the configuration server is enabled, the agent will overwrite its local configuration file by the configuration file it receives from the remote server. Even if the configuration server is disabled, the agent will still retrieve the blacklist on a regular basis from the server.

```
[agent]
role = 'home_agent' # home_agent/vps_agent/forwarder
enabled_services = ['ntp','dns','qotd','chargen','rip','ssdp']
ip_bindings = ['0.0.0.0']
```

The agent role is not used locally but transmitted together with the allowed services towards the configuration server. The configuration server can then choose a configuration that is suitable given the role of the agent. The enabled services is an optional setting and is used to filter out the services which it allows to run.

By default, the agent will detect all network interfaces and addresses. If one or more public addresses are found, it will bind to all of them. If this is unwanted behavior, a user can specify which ip_bindings the system is allowed to use. This can be a list of one or more items.

```
[pcap]
enabled = true
interface = 'any'
rotate_seconds = 1800 # Max 30 minutes
rotate_mbytes = 128  #MB's
capture_filter = 'udp and (port 17 or port 19 or port 53 or port 123 or port
520 or port 1900 or port 11211 or port 27960)'
file_prefix = "145.94.0"
folder = "/home/honeypot/pcaps/"
```

PCAP filtering is another optional feature of the system and requires root to run. An interface can be specified. On windows, tshark -D can be used to find the correct interface as it may not run on multiple interfaces on windows. For easier processing, a time interval can be entered, after which a new file will be created. A maximum file size can also be provided. A capture filter can be set in the tcpdump filter format. This helps to keep the capture files only to capture the relevant data. Finally, a file prefix and file location can be specified to separate the different pcaps of the different agents running at the same time.

```
[udp]
flow_dashboard_enabled = false
flow_controller_address = "https://udp-control.md-honeypot.nl"
flow_packets_max_rate = 35000
flow_packets_max_burst = 35000
flow_bytes_max_rate = 5 # Mbits
flow_bytes_max_burst = 20 # Mbits
packet_loss = 50 # In percent
packet_delay = 500 # In milliseconds
```

For UDP, the flow dashboard may be enabled for debugging purposes. This should not be exposed to the internet. The controller address is used to communicate about the current flows and retrieve any bandwidth limitations which need to be imposed on the current flows. Finally, local limitations are specified. The local limitations are overwritten by the flow controller when required, but provide a maximum value during the time when the flow is not yet known to the flow controller.

For certain experiments, the packet_loss and packet_delay options can be specified. The packet_loss setting will be the average packet loss in a percentage between 0% (no packet loss at all) and 100% (drop all packets). The first packet is never dropped unless the percentage is 100%. The packet delay ensures that no packet is sent back before the specified wait time has passed.

Finally, the definition of the services needs to be provided. An example setting for an NTP service using docker and a basic SSDP emulation is provided.

```
[services]

[services.ssdp]
protocol = 'UDP'
port = 1900
handler = "emulate_ssdp"
server_logging_enabled = true
settings_file = "/assets/ssdp-real-small"

[services.ntp]
protocol = 'UDP'
port = 123
handler = "docker"
server_logging_enabled = true

[services.ntp.docker]
enabled = true

[[services.ntp.dockers]]
image = 'pauluzznl/ntp'
image_prefix = "_pnl"
port = 123
generalize_source = true
generalize_dest = false
max_pool_size = 1
```

For each service, a protocol, port, and handler are defined. Connection logging to the server can be enabled and disabled. Depending on the emulator, a settings file or settings mode with emulator specific settings can be provided. For the docker handler, additional settings need to be provided. This includes the docker container image and a port to which the data will be forwarded towards the docker container. An additional image prefix may be provided so the service name can be reused while running different images.

By default, docker runs in a 1-on-1 mode. This means that for each unique IP address connecting towards each unique destination address, it will assign a new container. If the generalize_source setting is set to true, it will consider all connecting addresses as the same. The same applies to generalize_dest, but this works on the destination addresses. These settings provide for four different modes of operation. For this thesis, the settings listed above are used in all docker configurations as this would mimic real-world behavior.

Appendix C - Content of Experiment Groups

This appendix contains the content of the experiment groups for all the six different services. For each of the services, an explanation of the protocol is included, as well as their typical (network) bandwidth amplification factor. Table C.1 contains an overview of the BAF and NBAF of the different services and groups.

Table C.1 - Overview of (network) bandwidth amplification factors

Service	Group	BAF	NBAF
QOTD	A	47	2.1
QOTD	B	1450	34.7
QOTD	C	53	2.2
QOTD	D	1437	34.4
CHARGEN	A	94	3.2
CHARGEN	B	1406	33.7
CHARGEN	C	96	3.2
CHARGEN	D	1450	34.7
NTP	A	11.8	2.7
NTP	B	> 46	>8.2
NTP	C	43.4	7.8
NTP	D	0	0

Service	Group	BAF	NBAF
DNS	A	1.6	1.9
DNS	B	> 1.6	> 1.9
DNS	C	4.7	3.0
DNS	D	6.8	4.2
RIPv1	A	3.5	1.9
RIPv1	B	21.8	8.6
RIPv1	C	3.7	2.0
RIPv1	D	17.2	6.9
SSDP	A	2.3	2.0
SSDP	B	3.7	3.0
SSDP	C	2.4	2.0
SSDP	D	3.7	3.0

C.1. Routing Information Protocol v1

The first version of the routing information protocol also has a very simple format, and does not have a rich feature set. As such, this protocol is only implemented purely as an emulation. The two groups that include responses according to the specification start with the official header format and further include custom crafted responses that contain IP addresses in the local address space.

Group A (RS) responds with a small routing table consisting of 4 entries. This response has a total size of 84 bytes. The BAF is 3.5, and the NBAF is 1.9.

Group B (RL) responds with a larger routing table consisting of 26 entries. This response has a total size of 524 bytes. The BAF is 21.8, and the NBAF is 8.6.

Group C (FS) responds with data that indicates that the system is a honeypot. The response does not contain the official header format and as such the response will not be able to be parsed by a RIP protocol format parser. The data contains 88 bytes, resulting in a BAF of 3.7, and an NBAF of 2.0.

```
THIS IS A HONEYPOT. YOUR IP IS LOGGED. DO NOT USE THIS. YOU NOW PARTICIPATE
IN                                                              RESEARCH!
```

Group D (FL) is similar to group C but contain a much longer response. The data contains 413 bytes and clearly indicates that the system is not a real server. The BAF is 17.2, and the NBAF is 6.9.

```
This is a honeypot! This is not a real server! You should not use this RIP
server! Your IP is logged. We use this server to investigate who connects to
it and what happens with anyone that is connecting! So it is really best you
do not use this server! And if you do, please leave a cool message, since we
log all the packets anyway ;) - Thanks a lot for participating in our research
- Project Honeypot Research.
```

C.2. Character Generation

The Character Generation protocol discards any received input. Upon receipt of a request, it will return a random number of characters.

Group A (RS) with the string shown below. The string is rotated by one byte after each received request. Each string contains 94 bytes. The BAF of this group is 94 and the NBAF is 3.2.

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefg
```

Group B (RL) uses a static string of all the printed characters but rotated 19 times. The data contains 1406 bytes in total. The BAF 1406, the NBAF is 33.7.

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefgh
"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghi
#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghij
…
```

Group C (FS) uses a 94 byte static string, indicating clearly that the system is a honeypot. The BAF and NBAF are equal to group A – 94 and 3.2.

```
THIS IS A HONEYPOT. YOUR IP IS LOGGED. DO NOT USE THIS. YOU ARE NOW
PARTICIPATING IN RESEARCH!
```

Group D (FL) contains a message with a size of 1450 bytes, indicating clearly that the system is a honeypot. The BAF is 1450 and the NBAF is 34.7.

```
This is a honeypot system! This is not a real CHARGEN server! You should not
be using this CHARGEN server! Your IP internet address is logged in a database.
We use this server to investigate who connects to it and what happens with
these quotes! So it is really best you do not use this server! And if you do,
please leave a cool message, since we log all the packets anyway ;) We will
publish any misuse of this service on the internet and detect your IP internet
address. (Repeated 3x) | Project Honeypot Research
```

C.3. Quote of the Day

The QOTD protocol discards any input it receives and sends a random quote back. Thus, the response can be triggered by any received packet, but is usually one byte - the minimum packet size. Any returned content is a valid response according to the protocol specification.

Group A (RS) has a small random quote with an average size of around 45-50 bytes. The BAF varies between 44-50 and the NBAF between 2.0-2.1. An example quote is the following:

```
"To infinity .... and beyond! - Toy Story"
```

Group B (RL) contains large quotes with an average size of 1450 bytes. This results in BAF of 1470 and a NBAF of 34.7. Quotes in this group are a selection of Lorum Ipsum[30] texts. For example:

```
"Quisque et arcu vitae mi congue sodales. Nullam varius felis turpis, eget
semper lectus sollicitudin id. Suspendisse condimentum auctor convallis.
Praesent accumsan maximus nisi ac hendrerit. Nam fringilla leo sed lectus
commodo laoreet…. Mauris efficitur vehicula erat at maximus. Vestibulum
suscipit quam leo, at placerat risus aliquet non. In vel orci sed. - Lorum
Ipsum"
```

Group C (FS) contains a single quote with a size of 53 bytes, indicating clearly that the system is a honeypot. The BAF and NBAF are similar to group A: 53 and 2.2 respectively.

```
"This is a honeypot attack detector - Researchers"
```

Group D (FL) contains a single quote with a size of 1437 bytes, indicating clearly that the system is a honeypot. The BAF and NBAF are similar to group C: 1437 and 34.4 respectively.

[30] https://lipsum.com/

```
"This is a honeypot system! This is not a real QOTD server! You should not be
using this QOTD server! Your IP internet address is logged in a database. We
use this server to investigate who connects to it ans what happens with these
quotes! So it is really best you do not use this server! And if you do, please
leave a cool message, since we log all the packets anyway ;) We will publish
any misuse of this service on the internet and detect your IP internet address.
(Repeated 3x) -  Project Honeypot Research"
```

C.4. Simple Service Discovery Protocol

The Simple Service Discovery Protocol is also completely emulated. Groups A and B, that provide a response according to the specification, contain some dynamic elements. These dynamic elements are denoted by a name of a variable that are enclosed in brackets {}. Upon receipt of a packet, these variables are replaced by a value and then sent back. The {time} variable is automatically replaced with the current time, and the {timestamp} variable is automatically replaced with the current timestamp. Finally, the {server_ip} variable is replaced by the IP address of the server that received the request.

Group A (RS) returns one of the smallest but plausible response on an M-SEARCH request with a size of 272 bytes. The BAF is 2.3, and the NBAF is 2.0.

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
DATE: {time}
CACHE-CONTROL: max-age = 1800
LOCATION: http://{server_ip}/rootDesc.xml
SERVER: UPnP/1.0
NTS: ssdp:alive
NT: upnp:rootdevice
USN: uuid:b4ca5004c5334bf4883046f2ee3e871a::upnp:rootdevice
```

Group B (RL) returns a longer response with a size of 430 bytes. This results in a BAF of 3.7, and a NBAF of 3.0.

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
DATE: {time}
CACHE-CONTROL: max-age=60
LOCATION: http://{server_ip}:5000/rootDesc.xml
SERVER: OpenWRT/OpenWrt UPnP/1.1 MiniUPnPd/1.9
NT: upnp:rootdevice
USN: uuid:822db064-5a71-4375-ba79-20b582cd9309::upnp:rootdevice
NTS: ssdp:alive
OPT: "http://schemas.upnp.org/upnp/1/0/"; ns=01
01-NLS: {timestamp}
BOOTID.UPNP.ORG: {timestamp}
CONFIGID.UPNP.ORG: 1337
```

Group C (FS) returns 277 bytes of text data clearly indicating the system is a honeypot. The response is not according to the SSDP specification and will not be parsable as such. The BAF is 2.4, and the NBAF is 2.0, the same values as group A.

```
This is a honeypot! This is not a real server!
You should not use this SSDP server!
Your IP is logged.
We use this server to investigate who connects to it and what
happens with these quotes! So it is really best you do not use
this server!
- Project Honeypot Research
```

Group D (FL) is similar to group C but contains a longer response with 429 of text data. The BAF is 3.7, and the NBAF is 3.0, the same values as group B.

```
This is a honeypot system! This is not a real SSDP server!
You should not use this SSDP server!
Simple Service Discovery Protocol part of UPnP
Your IP network address is logged.
We use this server to investigate who connects to it and what
commands are transmitted to this server!
```

```
So it is really best you do not use this server!
And if you do, please leave a nice message ;)
With Regards,
- Project Honeypot Research
```

C.5. Network Time Protocol

The Network Time Protocol is also a more sophisticated service than the first four services. Therefore, we use both real virtualized servers running in containers and emulated software to detect if there is any difference between the two, with each of the two again using two different settings: a configuration that provides a high and a low amplification. During the test phase of the system, the groups were ordered differently. To ensure that the same servers could be reused safely, this different order of groups was reused for the main experiment. For readability, the groups below are discussed in the same order as the previous services.

Group D (RS) forwards the connection to a virtualized container that is running NTP software. The NTP software used is called NTPD, which is the most commonly used NTP server running on the internet. The version that is running has the amplification vulnerability fixed, and will thus drop any MONLIST packets it receives. Furthermore, it works as a fully functional NTP server. There is no amplification factor.

Group B (RL) is similar to group D, but runs an older version of the NTPD software that still contains the amplification vulnerability. This is the type of server that is abused in the wild since the vulnerability became known. The BAF is at least 46, and the NBAF is at least 8.2.

Group A (FS) uses an emulated NTP server that responds with an expected answer to a normal time synchronization request. When a MONLIST request is received, the server responds with the below 94 byte data string in a plain packet, indicating that the system is not a valid NTP server - it could be a honeypot. The BAF for the monlist request is 11.8, and the NBAF is 2.7.

```
MONLISTMONLISTMONLISTMONLISTMONLISTMONLISTMONLISTMONLISTMONLISTMONLISTMONLIS
TMONLISTMO
```

Group C (FL) is similar to group A. It responds like a regular NTP server on a time synchronization request. When a MONLIST request is received, it responds with the below 347 byte data string in a plain packet. The data string clearly indicates that the system is not a real NTP server, but is in fact a honeypot that gathers adversarial behavior. The BAF is 43.4 and the NBAF is 7.8.

```
This is a honeypot! This is not a real server! You should not use this NTPD
server! Your IP is logged. We use this server to investigate who connects to
it and what happens with these quotes! So it is really best you do not use
this server! And if you do, please leave a cool message, since we log all the
packets anyway. Honeypot Research
```

C.6. Domain Name Service

The Domain Name Service is a more complicated service than the previously four services, and cannot be easily emulated. As such, for this service, a real DNS server is used running in a virtualized container for the groups that are to provide a real response to a request. To test the hypothesis whether it matters if the server is real or emulated, two different types of emulations are also included - which do not implement the full feature set. This brings this experiment to four different groups:

Group A (RS): Forward the connections to a virtualized container running a regular DNS server. The type of used DNS server is the most commonly running server on the Internet, the BIND DNS server. Only one zone is configured: the example.com zone file. The full zone file and BIND configuration can be found in Appendix D. The BAF is at least 1.6, and the NBAF is at least 1.9.

Group B (RL): Group B is configured like group A, but with two differences: the server is configured as an open resolver and allows for recursion. The zone file is the same. The BAF is at least 1.6, and the NBAF is at least 1.9. Although the factors are similar to group A, these are minimal factors that can further grow towards higher amplification.

Group C (FS) is an emulated DNS server. It also uses the same zone file and every query is parsed as a query for the example.com zone. The BAF is 4.7, and the NBAF is 3.0.

Group D (FL) does not parse the incoming packet. It will just send back a 352 byte text response that indicates that the system is not a DNS server, but is a honeypot. The BAF is 6.8, and the NBAF is 4.2.

```
This is a honeypot! This is not a real server! You should not use this QOTD
server! Your IP is logged. We use this server to investigate who connects to
it and what happens with these quotes! So it is really best you do not use
this server! And if you do, please leave a cool message, since we log all the
packets anyway ;)| Project Honeypot Research
```

Appendix D - DNS Zone File

```
$ORIGIN example.com.     ; designates the start of this zone file in the namespace
$TTL 3600                ; default expiration time of all resource records without their own TTL value
example.com.  IN  SOA    ns.example.com. username.example.com. ( 2090620 86400 7200 604800 3600 )
example.com.  IN  NS     ns                    ; ns.example.com is a nameserver for example.com
example.com.  IN  NS     ns1.somewhere.example. ; ns.somewhere.example is a backup nameserver for example.com
example.com.  IN  MX     10 mail.example.com.  ; mail.example.com is the mailserver for example.com
@             IN  MX     20 mail2.example.com. ; equivalent to above line, "@" represents zone origin
@             IN  MX     50 mail3              ; equivalent to above line, but using a relative host name
example.com.  IN  A      192.0.2.1             ; IPv4 address for example.com
              IN  AAAA   2001:db8:10::1        ; IPv6 address for example.com
ns            IN  A      192.0.2.2             ; IPv4 address for ns.example.com
              IN  AAAA   2001:db8:10::2        ; IPv6 address for ns.example.com
www           IN  CNAME  example.com.          ; www.example.com is an alias for example.com
wwwtest IN        CNAME  www                   ; wwwtest.example.com is another alias for www.example.com
mail          IN  A      192.0.2.3             ; IPv4 address for mail.example.com
mail2         IN  A      192.0.2.4             ; IPv4 address for mail2.example.com
mail3         IN  A      192.0.2.6             ; IPv4 address for mail3.example.com
```

Appendix E - Honeypot Selection in Squeeze Experiment

The analysis in this appendix is similar as with the main experiment. These details are discussed in section 6.2.4. The squeeze experiment contained five different groups, starting from group 1 that provided a low amplification, to group 5, that provided the highest amplification. To limit the number of figures, only the figures comparing group 1-3, 2-5 and 3-5 are shown. These comparisons already provide the key findings that are important for the analysis.

RIP

The figures for RIP show similar results to the previous analysis of the squeeze experiment. There is a difference between group 1 (lowest amplification), group 3 (medium amplification). However, anything other than group 1 does not show any bias towards any of the higher groups.

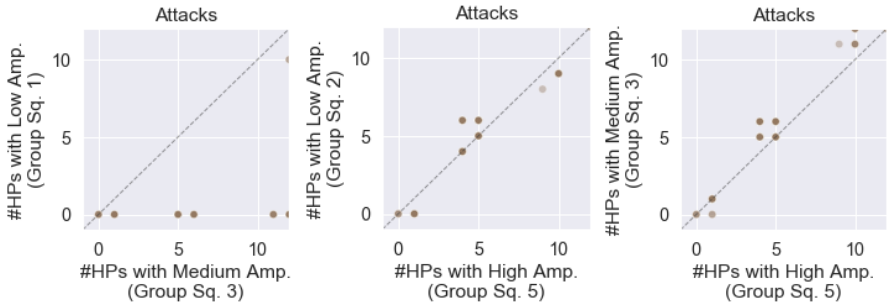


Figure E.1 - Honeypot selection in RIP attacks with the squeeze experiment

CHARGEN

CHARGEN does not show any difference among any of the groups. There is no bias to one squeeze group over another.

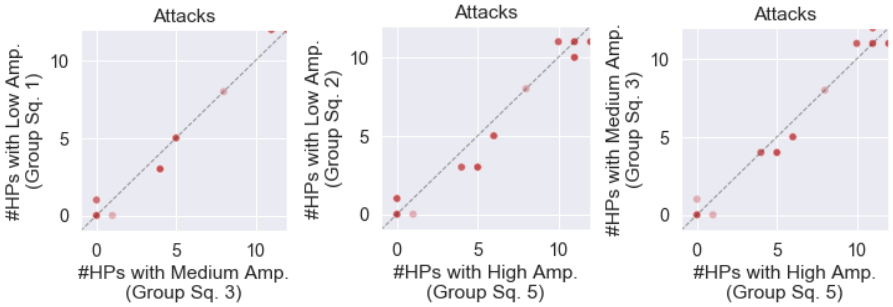


Figure E.2 - Honeypot selection in CHARGEN attacks with the squeeze experiment

QOTD

The results of the squeeze experiment for QOTD are similar to the results in the main experiment. There is a share of attacks that does not show any bias, and there is a cluster that specifically does not use the lower amplification honeypots. Similarly to the results for the main experiment, this seems to be especially the case with attacks that use a larger share of the available honeypots. There is no bias to one honeypot group or another between honeypots in groups 3, 4 and 5.

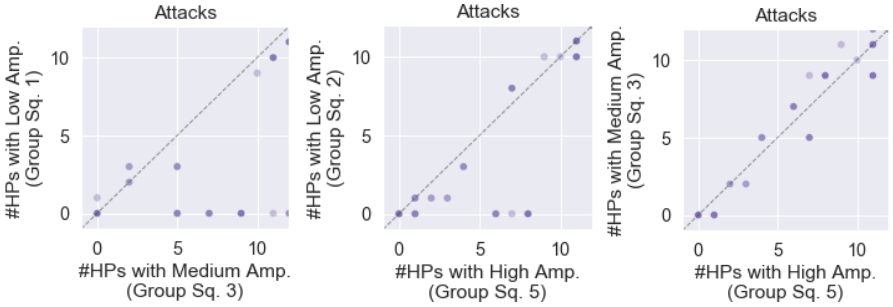


Figure E.3 - Honeypot selection in QOTD attacks with the squeeze experiment

SSDP

Previous analysis showed that the squeeze group 1 attracted the least amount of attacks, then group 2, and then the groups 3-5. Figure 6.4.24 shows that for the groups 1 and 2, there is either no bias, or a clear bias where only honeypots of a higher amplification group is used. However, the squeeze experiment also showed there was no significant difference between the groups 3-5. This may be the case, but it appears that there are definitely attacks that still specifically use honeypots with a higher amplification than in group 3.

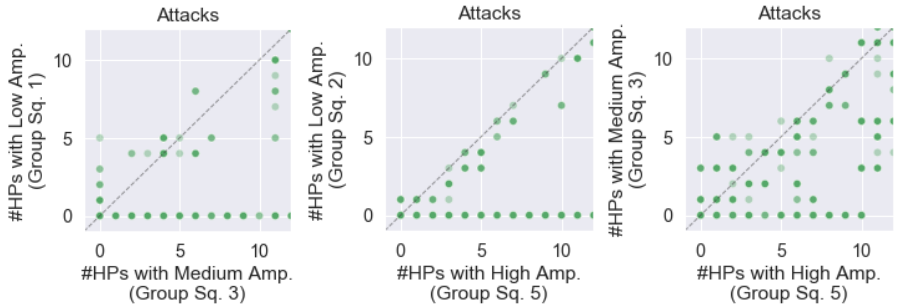


Figure E.4 - Honeypot selection in SSDP attacks with the squeeze experiment

NTP

NTP had a clear ordering where each subsequent group attracted more attacks than the group below, except the 4th and 5th group. The figures for NTP show that there is either no apparent bias, or a strong bias where attacks are only used in the higher amplification group (5).

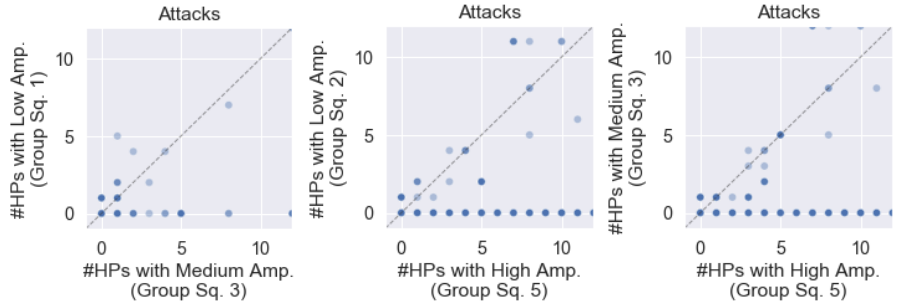


Figure E.5 - Honeypot selection in NTP attacks with the squeeze experiment

DNS

DNS shows that for a large part, there is no clear bias towards the higher amplification groups. However, there does appear to be a gap: attacks that use the majority of available honeypots do appear to have a stronger bias towards the higher amplification group 5.

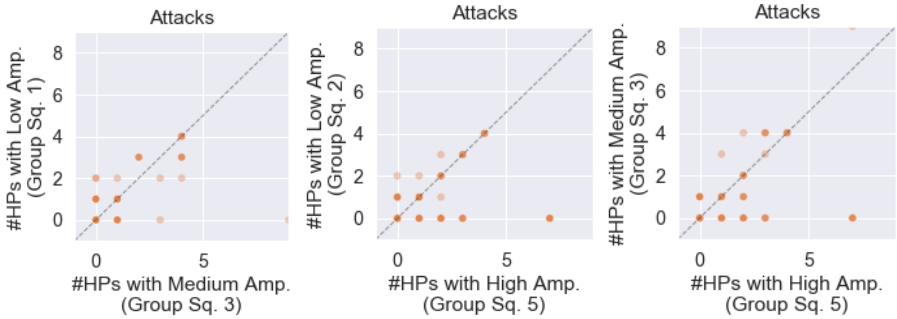


Figure E.6 - Honeypot selection in DNS attacks with the squeeze experiment

Appendix F - Tables of Used Packets

This appendix contains the most frequently used received packets of data for each of the services that the honeypots were running. Only packets that were used in at least 5 attacks and are in the top 10 used packets in attacks are included. The data that is presented is in base64 encoded format, so it is binary-safe and can be compared to other work without loss of data. The other columns show the number of attacks (Atts.), the share of multi-vector attacks where this packet was detected, the share of subnet (SN) attacks where the packet was detected and finally the share of attacks in each of the different groups of honeypots where the attack was detected. Groups A-D refer to the primary groups that were used throughout the experiment and the sub-experiments and the groups 1-5 refer to the squeeze experiment that used different groups.

Note that the share refers to the share of attacks within the selected column. For example, the first packet in QOTD has a share of 28.6%. This means that of all QOTD multi-vector attacks, 28.6% of attacks detected that packet. Note that multiple different packets can be used within the same attack, so the cumulative share can be more than 100%.

RIP

Table E.1 - Packets used with RIP with >= 1 attacks

Data	Atts.	MV	SN	Gr. A	Gr. B	Gr. C	Gr. D	Gr. 1	Gr. 2	Gr. 3	Gr. 4	Gr. 5
AQEAAAAAAAAAAAAAAAAAAAAAAAAAAAQ	192	91.3	0.0	100.0	99.5	98.9	99.5	100.0	100.0	100.0	100.0	99.5
AQIAAAAAAAAAAAAAAAAAAAAAAAAAAAQ	5	21.7	0.0	0.5	1.6	0.0	0.0	0.0	0.0	0.0	0.0	0.5

CHARGEN

Table E.2 - Packets used with CHARGEN with >= 5 attacks

Data	Atts.	MV	SN	Gr. A	Gr. B	Gr. C	Gr. D	Gr. 1	Gr. 2	Gr. 3	Gr. 4	Gr. 5
AQ==	383	45.9	13.6	99.6	71.7	73.2	71.7	100.0	100.0	99.6	100.0	100.0
MA==	164	62.4	100.0	0.0	32.1	31.9	32.6	0.0	0.0	0.0	0.0	0.0
Cg==	23	3.5	4.5	2.1	0.7	1.0	0.7	0.7	0.7	0.0	1.1	0.4
YQ==	9	2.4	0.0	0.4	1.4	1.0	1.7	0.4	0.4	0.0	0.4	0.0
Rw==	6	0.0	0.0	0.0	0.2	0.4	0.7	0.0	0.0	0.0	0.0	0.0

QOTD

Table E.3 - Packets used with QOTD with >= 5 attacks

Data	Atts.	MV	SN	Gr. A	Gr. B	Gr. C	Gr. D	Gr. 1	Gr. 2	Gr. 3	Gr. 4	Gr. 5
AQ==	212	28.6	0.0	0.0	77.9	50.0	70.8	0.0	0.0	73.6	72.5	70.8
YmlnYm8=	30	0.0	0.0	49.2	11.0	23.8	14.4	81.1	51.8	14.4	14.2	14.4
/////2dldHN0YXR1cxA=	24	42.9	0.0	39.3	8.5	19.0	11.5	0.0	35.7	8.5	9.8	11.5
Cg==	13	14.3	0.0	0.0	3.3	0.0	3.3	0.0	0.0	0.0	1.0	0.0
DQA=	7	0.0	0.0	11.5	2.6	5.6	3.3	18.9	12.5	3.5	3.4	3.3

SSDP

Table E.4 - Top 10 packets used with SSDP

Data	Atts.	MV	SN	Gr. A	Gr. B	Gr. C	Gr. D	Gr. 1	Gr. 2	Gr. 3	Gr. 4	Gr. 5
TS1TRUFSQ0ggKiBIVFRQLzEuMQ0KSG9zdDoyMzkuMj	1601	98.2	100.0	94.0	97.1	94.3	93.8	64.2	68.8	95.2	96.8	96.2

U1LjI1NS4yNTA6MTkwMA0KU1Q6c3NkcDphbGwNCk1hbjoic3NkcDpkaXNjb3ZlciINCk1YOjMNCg0K												
TS1TRUFSQ0ggKiBIVFRQLzEuMQ0KSG9zdDoyMzkuMjU1LjI1NS4yNTA6MTkwMA0KU1Q6dXBucDpyb290ZGV2aWNlDQpNYW46InNzZHA6ZGlzY292ZXIiDQpN	397	18.3	18.8	6.0	10.2	7.4	9.1	3.3	3.2	1.5	1.8	2.0
TS1TRUFSQ0ggKiBIVFRQLzEuMQ0KSG9zdDoyMzkuMjU1LjI1NS4yNTA6MTkwMA0KU1Q6dXBucDpyb290ZGV2aWNlDQpNYW46InNzZHA6ZGlzY292ZXIiDQpNWDozDQoNCg0K	126	0.0	0.0	2.1	0.0	2.7	2.7	4.0	2.6	0.2	0.0	0.3
/////2Rpc2Nvbm5lY3Q=	122	0.0	0.0	2.1	0.0	2.6	2.6	4.0	2.6	0.2	0.0	0.3
TS1TRUFSQ0ggKiBIVFRQLzEuMQ0KSE9TVDogMjM5LjI1NS4yNTUuMjUwOjE5MDANCk1BTjogInNzZHA6ZGlzY292ZXIiDQpNWDogMg0KU1Q6IHNzZHA6YWxs	73	2.2	0.0	1.3	1.9	1.8	2.1	0.0	0.0	0.1	0.3	0.5
TS1TRUFSQ0ggKiBIVFRQLzEuMQ0KU1Q6c3NkcDphbGwNCg==	46	0.4	0.0	3.6	2.8	2.8	2.7	30.5	27.9	4.4	3.2	3.2
TS1TRUFSQ0ggKiBIVFRQLzEuMQ0KSG9zdDogMjM5LjI1NS4yNTUuMjUwOjE5MDANCk1hbjogInNzZHA6ZGlzY292ZXIiDQpNWDogMw0KU1Q6IHNzZHA6YWxs	44	0.9	0.0	1.0	1.2	0.9	1.3	0.0	0.0	0.2	0.1	0.1
TS1TRUFSQ0ggKiBIVFRQLzEuMQ0KSE9TVDogMjM5LjI1NS4yNTUuMjUwOjE5MDANCk1BTjogInNzZHA6ZGlzY292ZXIiDQpNWDozDQpTVDpzc2RwOmFsbA0K	43	0.9	0.0	1.0	1.3	1.2	1.3	0.0	0.0	0.3	0.3	0.2
TS1TRUFSQ0ggKiBIVFRQLzEuMQ0KSG9zdDoyMzkuMjU1Lg==	30	0.4	0.0	1.0	0.5	0.4	0.8	1.3	1.9	0.0	0.0	0.1
TS1TRUFSQ0ggKiBIVFRQLzEuMQ0KSE9TVDogMjM5LjI1NS4yNTUuMjUwOjE5MDANCk1BTjogInNzZHA6ZGlzY292ZXIiDQpTVDogc3NkcDphbGwNCk1YOiAz	27	0.9	0.0	0.7	0.7	0.2	0.4	0.0	0.0	0.0	0.0	0.1

NTP

Table E.5 - Top 10 packets used with NTP

Data	Atts.	MV	SN	Gr. A	Gr. B	Gr. C	Gr. D	Gr. 1	Gr. 2	Gr. 3	Gr. 4	Gr. 5
FwADKgAAAAA=	15544	97.0	99.6	91.6	99.9	97.7	0.0	37.1	86.1	62.6	99.8	99.8
FgYqAAAAAA==	554	0.0	0.0	8.1	0.0	2.0	0.0	61.3	13.4	36.4	0.1	0.1
4wAE+gABAAA=	480	0.4	7.0	0.4	5.2	0.2	0.0	0.0	0.5	1.0	0.0	0.1
FgIAAQAAAAA=	29	0.9	0.0	0.2	0.1	0.1	0.0	0.0	1.0	1.0	0.0	0.0
FwADKgAAAA==	29	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.1	0.1
4wAAAAAAAAA=	19	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0
R0VUIC8gSFQ=	11	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
FwACKgAAAAA=	10	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
EwAAAAAAAAA=	8	0.4	0.0	0.1	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
FwACKgAAAAAAAAAAAAAAAAAA==	7	0.0	0.0	0.1	0.0	0.0	0.0	1.6	0.0	0.0	0.0	0.0

DNS

Table E.6 - Top 10 packets used with DNS

Data	Atts.	MV	SN	Gr. A	Gr. B	Gr. C	Gr. D	Gr. 1	Gr. 2	Gr. 3	Gr. 4	Gr. 5
AAYBAAABAAAAAAAAAB3ZlcnNpb24EYmluZAAAEAAD	127	0.0	0.0	2.0	0.1	1.7	31.6	31.6	23.1	22.0	16.9	16.1
ZDE6YWQyOmlkMjA6YWJjZGVmZ2hpajAxMjM0NTY3ODk2OnRhcmdldDIwOm1ub3BxcnN0dXZ3eHl6MTIzNDU2ZTE6cTk6ZmluZF9ub2RlMTp0MjphYTE6eTE6cWU=	82	0.0	0.0	0.0	0.0	0.8	22.0	15.8	17.9	16.0	10.8	10.7
BwABAAABAAAAAAAABmdvb2dsZQNjb20AAAEA	75	0.0	0.0	28.0	3.0	14.9	7.2	7.0	7.7	10.0	10.8	10.7
BwABAAABAAAAAAAABmdvb2dsZQNjb20AAAEAAQwAAQ==	61	0.0	0.0	17.3	1.0	19.8	12.9	8.8	15.4	14.0	9.2	7.1
BwABAAABAAAAAAAABmdvb2dsZQNjb20AAAEAAQ==	57	0.0	0.0	5.3	0.5	28.1	20.1	17.5	25.6	26.0	20.0	23.2
BwABAAABAAAAAAAABWFwcGxlA2NvbQAAAQABAQwAAQ==	38	0.0	0.0	12.0	0.1	14.0	8.1	7.0	12.8	10.0	7.7	3.6
BwABAAABAAAAAAAABWFwcGxlA2NvbQAAAQABAQ==	38	0.0	0.0	1.3	0.1	20.7	12.4	10.5	20.5	12.0	18.5	14.3
BwABAAABAAAAAAAABWFwcGxlA2NvbQAAAQAB	37	0.0	0.0	20.7	0.5	8.3	6.7	5.3	2.6	4.0	3.1	3.6
BwABAAABAAAAAAAABmdvb2dsZQNjb20AAAEAAQAB	36	0.0	0.0	6.0	1.4	12.4	8.6	5.3	0.0	4.0	0.0	1.8
BwABAAABAAAAAAAABmdvb2dsZQNjb20AAAEAAQAMAAE=	36	0.0	0.0	8.7	1.0	10.7	5.3	12.3	12.8	10.0	6.2	17.9

Glossary

API
Application programming interface, a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or another service.

Botnet
A botnet is a network of (Internet) connected devices that are infected with malicious software. They are controlled by a group or individual without the awareness of the owner of the device.

Booter Services
A booter service provides a DDoS attack as a service. Booter services may make use of compromised servers to generate their traffic - but this is not always the case. In theory, a booter service could execute massive attacks by using just a single server in a network that allows for address spoofing. A customer orders a DDoS attack of a specified duration and quality and usually pays for these services using anonymous currencies such as Bitcoin. It is not surprising that these attacks are so frequent, as these booter services are incredibly cheap [40,63]. A DDoS attack with a short duration of less than an hour, against medium protected targets, only costs $5-20. Attacks with longer durations against better protected targets are more expensive, between $10-1000.

Character Generation (CHARGEN)
The Character Generation protocol originated in 1983 and was mostly used as a network debugging tool. The official specification[31] for this service specifies the following: “CHARGEN listens for UDP datagrams on port 19. Whenever a packet is received, the server will respond with a random number of characters, and the data of the received packet is discarded.”

Again, the length of the response could be any arbitrary value. The most popularly deployed implementations of the CHARGEN protocol use a round-robin scheme of printable ASCII characters and then shift the characters to the left one by one. For example, consecutive packets would like the following:

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefgh..
"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghi..
#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghij..
```

The typical amplification factor of these services is around 358, according to Bohte [43].

Domain Name Service (DNS)
The domain name system is also a complex protocol. Internet routing works on IP addressing, as previously discussed, but this is not very feasible for humans to remember. If an Internet user would like to google something, it would be cumbersome to enter its IP address of 172.217.168.195. It is not user-friendly if users would need to remember all the dozens of IP addresses in numerical format to reach a website. To tackle this problem and make visiting websites more user friendly, users started to make use of a host file. The host file contained names and an IP address where this name could be found. Thus, users could enter an address such as google.nl, and the computer would translate this automatically to the correct IP address. This was a good solution at first, but when the Internet grew, manually maintaining these lists became prone to errors and was no longer feasible.

After some iterations, in 1987, the Domain Name Service system was released that solved this problem. Without going into too much detail, the DNS system converts human-readable addresses to internet addresses automatically, without the need to manually update it. The typical amplification factor is 28 to 54 according to Bohte [43].

Latency
Latency is a time delay in data exchange caused by the physical limitations of the medium in which information is transmitted. For example, while transmitting information between the Netherlands and the United States, the latency will be at least the amount of milliseconds light will take to travel between these locations.

[31] https://tools.ietf.org/html/rfc864

Multi-vector Attack

Some attacks use only a single type of vulnerable service in an attack. For example, they use many vulnerable weather services that were used an an example in the background chapter. In multi-vector attacks, adversaries use services of different types in their attack. For example, they not only use vulnerable weather services in an attack but also vulnerable servers that normally provide the current time.

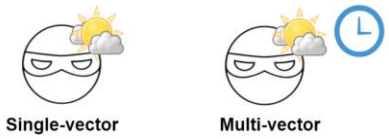


Figure G.1 - Single and multi-vector attacks

Network Time Protocol (NTP)

While almost any electronic device has a battery and a clock, they may have different accuracies and are known to drift away from the actual time. It is not feasible to use an atom clock for all those devices. However, synchronizing the time of those electronic devices over the network to the time of an atom clock would solve this problem. This is precisely the purpose of the network time protocol (NTP).

The NTP protocol is much more sophisticated than both QOTD and the CHARGEN protocol. First released in 1985[32], the NTP is a protocol and provides methods to synchronize time with nanosecond precision between different nodes on the Internet.

Unfortunately, a feature in one of the software packages implementing this protocol resulted in unintended side effects making the servers running the software vulnerable to DRDoS attacks. This implementation, the ntpd package for Linux systems, contained a debugging tool to request a list of maximum 600 hosts with which the NTP server communicated recently [64]. This feature was implemented as a 'mode 7' feature. This mode, specified in the NTP specification from 2010[33], was reserved for 'private use'. The feature was implemented and enabled by default for all installations in 2010. Shortly after, the request to remove the feature was already made as a bug report[34]. It had, however, gone unnoticed as a severe issue.

In December 2013, adversaries started exploiting this unintended feature on a large scale. Cloudflare [64] reported attacks of over 400 gigabit per second on some of its customers. The attack at the time was so large that it congested parts of the network in Europe. By that time, the bug was marked as a major vulnerability. At that moment, changes were applied so that the feature was no longer enabled by default. However, today, five years later, Bohte [43] estimated that there are still roughly 1.4 million open amplifiers available as the configuration of many of these NTP servers have not been updated.

The request to trigger such a large response is straightforward. The request needs only to be 8 specific static bytes that request the monlist. The typical amplification varies between 38x but could - depending on the monlist - even be around 500x. Bohte [43] reported a typical amplification factor of 557.

Protocol

A protocol is a set of rules and procedures for transmitting data between electronic devices. It ensures that the different electronic devices understand each other when communicating data. It can be compared to a language. It is a set of grammar and words to exchange information.

Payload

The payload is the part of transmitted data that is the actual intended message. Headers and metadata are sent only to enable payload delivery.

[32] https://tools.ietf.org/html/rfc958
[33] https://tools.ietf.org/html/rfc5905
[34] https://bugs.ntp.org/show_bug.cgi?id=1532

Quote of the Day (QOTD)

The Quote of the Day protocol was used to send daily quotes to terminals and also served as a network debugging tool. The protocol originates from 1983 and is now long considered a legacy protocol as it is no longer used. It has since been replaced by much more robust network debugging tools. However, many servers in the wild still run a service with this outdated protocol.

The official specification[35] of the protocol specifies that the server listens for UDP datagrams on port 17. Whenever a packet is received, the server will respond with a packet containing a quote. The received data is ignored. While the specification specifies the returned data to be a quote, the actual data could be anything. According to the specification, the length should be less than 512 bytes, but the length could be anything. As such, the amplification factor could vary from one to a lot. Bohte [43] reported that the typical amplification factor of the service is 140.3.

Routing Information Protocol version 1 (RIPv1)

When a router receives a packet, it needs to determine which outgoing link to use to send the packet to. Ultimately, the packet must reach its destination. Routers use routing tables to determine what the next hop in the network should be to forward the packet to, to eventually reach their destination.

The Routing Information Protocol (RIP) is used to exchange routing information between gateways and other hosts. The standard for the routing information protocol was released in 1988[36], but by then it was already used for six years. The routing information for each host contains addresses that it can route and a cost (metric) for it to reach the specific network. In the first implementation of this protocol, each host would distribute its routing table every thirty seconds to its neighbors. But a host can also request its neighbors for them to send back a part of their current routing table.

However, there is a specific case defined in the RFC *"If there is exactly one entry in the request, with an address family identifier of 0 (meaning unspecified), and a metric of infinity (i.e., 16 for current implementations), this is a request to send the entire routing table. In that case, a call is made to the output process to send the routing table to the requesting port."* This means that a large response may be triggered by crafting a specific packet. If the routing tables are large, this can provide an enormous amplification factor. An amplification factor of above 100 is not unusual. Bohte [43] reported a typical amplification factor of 131.

Routing table

A routing table consists of entries that describe how to reach certain destinations in a network.

Simple Service Discovery Protocol (SSDP)

Universal Plug and Play (UPnP) is a set of protocols that allow devices connected to a network to easily discover each other's presence and establish functional network services for data sharing, communication, and entertainment. For example, UPnP is used to find, use, and connect your mobile phone to a chromecast to play Netflix on your TV. The discovery part of UPnP is covered by the Simple Service Discovery Protocol[37] (SSDP). SSDP ensures no central server is required to let different devices discovered each other on the network.

SSDP was designed in 1999, much later than the other described protocols. It works with the HTTPU protocol, which is similar to HTTP but runs on the UDP protocol. The protocol consists of three types of messages; one is a discovery request, one is a response, and one is an active announcement. Only the request and response messages are relevant for our service, as no active announcement happens on the Internet.

When a service becomes active, it sends out a NOTIFY on a local network. Other devices then become aware of the presence of the service. A client that just connected his device may also want to discover other devices on the network by sending out a M-SEARCH request. These devices may then respond with a 'HTTP/1.1 200 OK' message, or the device sends out a new NOTIFY request. As all messages are sent on the network broadcast address, all devices see all the individual packets of any type.

This works well for home or local networks, but should never be run for networks connected to the Internet without a firewall in between. However, devices are sometimes directly connected to the Internet, and firewalls are also not always

[35] https://tools.ietf.org/html/rfc865
[36] https://tools.ietf.org/html/rfc1058
[37] https://tools.ietf.org/html/draft-cai-ssdp-v1-03

appropriately configured to block SSDP messages from leaving the network. This means that an adversary could send a M-SEARCH request from a spoofed source, and the device will respond with a longer NOTIFY or OK message back. Depending on the configured service and the number of services running on a network, the amplification factor will vary. The typical amplification factor reported by Bohte was around 30. This is a bit lower than the rest of the protocols. This can be explained by the fact that the query to get a response is also larger than with the other protocols.

Below are included an example request and response query that is used in a home network. Note that the response contains a location to a rootDesc.xml that provides more information about the UPnP service.

(REQUEST)

M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
USER-AGENT: UPnP/1.0 DLNADOC/1.50 Platinum/1.0.4.2
MAN: "ssdp:discover"
ST: ssdp:all
MX: 3

(RESPONSE)

NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age=60
LOCATION: http://172.16.0.1:5000/rootDesc.xml
SERVER: OpenWRT/OpenWrt UPnP/1.1 MiniUPnPd/1.9
NT: urn:schemas-upnp-org:service:Layer3Forwarding:1
USN: uuid:762da019-715a-4375-79bb-94b582cd9361::urn:schemas-upnp-org:service:Layer3Forwarding:1
NTS: ssdp:alive
OPT: "http://schemas.upnp.org/upnp/1/0/"; ns=01
01-NLS: 1569488741
BOOTID.UPNP.ORG: 1569488741
CONFIGID.UPNP.ORG: 1337

Subnet Attacks

Some attacks merely target a single address, for example, a website, a broadband internet connection, or a game server. However, there are also attacks that target multiple addresses in the same network. Figure G.2 shows a schematic example of a network of a bank. Attacking a single address could mean that adversaries attack the webserver (www) of the bank, potentially leading to a website that is no longer available. In that same network, other services of a bank could also be hosted, for example, a database server that contains customer information, and a payment server that processes transactions. With a subnet attack, all these targets are targeted simultaneously.

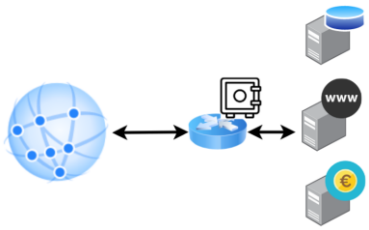


Figure G.2 - Subnet attacks

Bibliography

[1] Craig Timberg. A flaw in the design the internet's founders saw its promise but didn't foresee users attacking one another. https://www.statista.com/statistics/319732/daily-time-spent-online-device/, June 2019. Accessed: 2020-1-27.

[2] Daily time spent with the internet per capita worldwide from 2011 to 2021, by device.https://www.statista.com/statistics/319732/daily-time-spent-online-device/, June 2019. Accessed: 2020-1-27.

[3] Simon Kemp. DIGITAL 2019:GLOBAL INTERNET USE ACCELERATES. https://wearesocial.com/blog/2019/01/digital-2019-global-internet-use-accelerates. Accessed: 2019-1-27.

[4] David Kravents. U.N. report declares internet access a human right.https://www.wired.com/2011/06/internet-a-human-right/, June 2011. Accessed: 2019-10-8.

[5] Business Roundtable. Growing business dependence on the internet. Technical report, 2007.

[6] ENISA. Threat landscape report. Technical Report 2018, January 2019

[7] CloudFlare. What is a DDoS attack? https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/,.Accessed: 2020-1-27.

[8] Arbor Networks. Worldwide Infrastructure Security Report. Technical Report 10, 2014.

[9] DDoS monitor insight. https://ddosmon.net/insight/. Accessed: 2019-09

[10] Gary C. Kessler. Defenses against distributed denial of service attacks. November 2000.

[11] F. Lau, S.H. Rubin, M.H. Smith, L. Trajkovic. Distributed denial of service attacks. In ieee international conference on systems, man and cybernetics, 2000

[12] FBI. Today's FBI fact and figures. April 2003. https://web.archive.org/web/20070326115414/http://www.fbi.gov/libref/factsfigure/factsfiguresapri2003.htm Accessed August 2019.

[13] Matt Richtel. Canada arrests 15-Year-Old in web attack. April 2000. https://www.nytimes.com/2000/04/20/business/canada-arrests-15-year-old-in-web-attack.html Accessed August 2019.

[14] State of IP Spoofing, Center for Applied Internet Data Analysis. https://spoofer.caida.org/summary.php Accessed 2019-09

[15] Jose Nazario. DDoS attack evolution, March 2008.

[16] Jose Nazario. Politically motivated denial of service attacks, December 2018.

[17] Damien McGuinness. How a cyber attack transformed estonia. https://www.bbc.com/news/39655415, April 2017. Accessed: 2020-1-26.

[18] Adrian Croft, Peter Apps. *NATO websites hit in cyber attack linked to crimea tension. March 2014.* https://www.reuters.com/article/us-ukraine-nato/nato-websites-hit-in-cyber-attack-linked-to-crimea-tension-idUSBREA2E0T320140316. Accessed August 2019.

[19] Telegram. We're currently experiencing a powerful DDoS attack, telegram users in the americas and some users from other countries may experience connection issues. https://twitter.com/telegram/status/1138768124914929664, June 2019. Accessed: 2020-1-27.

[20] Pavel Durov. IP addresses coming mostly from china. historically, all state actor-sized DDoS(200-400 gb/s of junk) we experienced coincided in time with protests in hong kong (coordinated on @telegram). this case was not an exception.https://twitter.com/durov/status/1138942773430804480, June 2019. Accessed: 2019-12-6.

[21] Charles Arthur. WikiLeaks under attack: the definitive timeline. Januari 2010. https://www.theguardian.com/media/2010/dec/07/wikileaks-under-attack-definitive-timeline Accessed August 2019.

[22] 'tis the season of DDoS – WikiLeaks edition. *Pada Security. December 2010*. Accessed 2019-07 https://www.pandasecurity.com/mediacenter/news/tis-the-season-of-ddos-wikileaks-editio/

[23] Anonymous hackers 'cost PayPal £3.5m'. https://www.bbc.com/news/uk-20449474,November 2012. Accessed: 2020-1-28.

[24] Max Pritchard. DDoS attack timeline: Time to take DDoS seriously. https://activereach.net/newsroom/blog/time-to-take-ddos-seriously-a-recent-timeline-of-events/, April 2018. Accessed: 2020-1-7

[25] Huib Modderkolk. In gesprek met jelle s. (18), die met ddos-aanvallen de belastingdienst en banken zou hebben platgelegd. https://www.volkskrant.nl/wetenschap/in-gesprek-met-jelle-s-18-die-met-ddos-aanvallen-de-belastingdienst-en-banken-zou-hebben-platgelegd~b3887d7b/,2018. Accessed: 2019-7.

[26] Wout Funnekotter, Tweakers.net June 2018. Een ddos'er betrapt. Hoe de aanvaller tegen de lamp liep https://tweakers.net/reviews/6031/een-ddoser-betrapt-hoe-de-aanvaller-tegen-de-lamp-liep.html Accessed Juli 2019.

[27] Darren Pauli. Teen UK hacker pleads guilty after earning $385k from DDoS tool.https://www.theregister.co.uk/2016/11/02/teen_uk_hacker_pleads_guilty_after_earning_385k_from_ddos_tool/, November 2016. Accessed: 2019-7.

[28] Mark Hendrikman. Ubisoft beschuldigt nederlandse minderjarige in rechtszaak tegen ddos-aanbieders. https://tweakers.net/nieuws/162546/ubisoft-beschuldigt-nederlandse-minderjarige-in-rechtszaak-tegen-ddos-aanbieders.html, January 2020. Accessed: 2020-1-21

[29] NETSCOUT. NETSCOUT's 14th annual worldwide infrastructure security report. Technical Report 14, 2019

[30] Inside 'the attack that almost broke the internet'. https://krebsonsecurity.com/2016/08/inside-the-attack-that-almost-broke-the-internet/, August 2016. Accessed:2020-1-22

[31] Matthew Prince. The DDoS that almost broke the internet. https://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet/, March 2013. Accessed: 2020-4-7.

[32] Nicky Woolf. DDoS attack that disrupted internet was largest of its kind in history, experts say. https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet, October 2016. Accessed: 2020-1-10.

[33] Ethan Ciel. Here are the sites you can't access because someone took the internet down. https://splinternews.com/here-are-the-sites-you-cant-access-because-someone-took-1793863079, October 2016. Accessed: 2020-1-10.

[34] Lee Mathews. Angry gamer blamed for most devastating DDoS of 2016. https://www.forbes.com/sites/leemathews/2016/11/17/angry-gamer-blamed-for-most-devastating-ddos-of-2016/#14221a252dac,November 2016. Accessed: 2020-1-10.

[35] Paul Roberts. Exclusive: Mirai attack was costly for dyn,data suggests. https://securityledger.com/2017/02/mirai-attack-was-costly-for-dyn-data-suggests/, February 2017. Accessed:2020-1-10

[36] DreamHost caught in alt-right crossfire. https://www.datacenterdynamics.com/news/dreamhost-caught-in-alt-right-crossfire/, August 2017. Accessed: 2020-1-10.

[37] Tomer Shani.This DDoS attack unleashed the most packets per second ever. here's why that's important. https://www.imperva.com/blog/this-ddos-attack-unleashed-the-most-packets-per-second-ever-heres-why-thats-important/,April 2019. Accessed: 2020-1-10.

[38] CloudFlare. Famous DDoS attacks | the largest DDoS attacks of all time. https://www.cloudflare.com/learning/ddos/famous-ddos-attacks/ Accessed December 2019.

[39] NBIP. The impact of DDoS attacks on dutch enterprises, 2018.

[40] Symantec. Internet security threat report 2018. Technical Report 23, 2018.

[41] Arbor Networks. The risk vs. cost of enterprise DDoS protection. Technical report, 2012.

[42] Mike Stevens. How to scan whole internet 3.7 billion IP addresses in few minutes? https://www.securitynewspaper.com/2015/10/15/how-to-scan-whole-internet-3-7-billion-ip-addresses-in-few-minutes/

[43] Edgar Bohte. Evaluation of current state of amplification-based DDoS attacks. 2018.

[44] Lukas Krämert, Johannes Krupp, Daisuke Makita, Tomomi Nishizoe, Takashi Koide, Katsunari Yoshioka, Christian Rossow, Research in Attacks, Intrusions, and Defenses (2015) 615–636. AmpPot: Monitoring and defending against amplification DDoS attacks.

[45] C. Rossow, Proceedings 2014 Network and Distributed System Security Symposium (2014). Amplification hell: Revisiting network protocols for DDoS abuse

[46] ICANN. Available pool of unallocated IPv4 internet addresses now completely emptied. March 2011.

[47] Bryan Harris, Eli Konikoff, Phillip Petersen. Breaking the DDoS attack chain. 2013.

[48] Eric M. Hutchins, Michael J. Cloppert, Rohan M. Amin. Intelligence-Driven computer network Defense Informed by analysis of adversary campaigns and intrusion kill chains. Lockheed Martin Corporation.

[49] H J Griffioen. Scanners discovery of distributed slow scanners in telescope data. 2018.

[50] Marc Kuhrer, Thomas Hupperich, Christian Rossow, Thorsten Holz, in: Horst Gortz Institute for IT-Security, Ruhr-University Bochum. Exit from hell? reducing the impact of amplification DDoS attacks.

[51] L. Spitzner, Honeypots: Tracking Hackers, Addison Wesley, 2002.

[52] N. Provos, in: Proceedings of the 10th DFNCERT Workshop, 2003, pp. 1–7. Honeyd: A virtual honeypot daemon.

[53] X. Sun, Y. Wang, J. Ren, Y. Zhu, S. Liu, 2008 The 9th International Conference for Young Computer Scientists (2008). Collecting internet malware based on client-side honeypot.

[54] M.S. Zemene, M. Solomon Zemene, P.S. Avadhani, 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (2015). Implementing high interaction honeypot to study SSH attacks.

[55] Sille Kamoen. Honeytrack: Persistent honeypot for the internet of things, 2018.

[56] J. Krupp, M. Backes, C. Rossow, Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16 (2016). Identifying the scan and attack infrastructures behind amplification DDoS attacks.

[57] J. Krupp, M. Karami, C. Rossow, D. McCoy, M. Backes, Research in Attacks, Intrusions, and Defenses (2017) 427–449. Linking amplification DDoS attacks to booter services.

[58] Marek Majkowski. How to receive a million packets per second. CloudFlare, June 2015. https://new.blog.cloudflare.com/how-to-receive-a-million-packets/ Accessed Juli 2019.

[59] Razali, Wah. Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, lilliefors and Anderson Darling tests. Journal of Statistical Modeling and Analytics, 2(1):21–33, 2011

[60] Johan Mazel, Rémi Strullu. Identifying and characterizing ZMap scans: a cryptanalytic approach. ANSSI, August 2019.

[61] CVE-2002-0510 linux UDP. https://nvd.nist.gov/vuln/detail/CVE-2002-0510, June 2020. Accessed: 2002-8-12

[62] Linux kernel version history. https://en.wikipedia.org/wiki/Linux_kernel_version_history. Accessed: 2020-6-4

[63] Symantec. Internet security threat report 2019. Technical Report 2019, 2019

[64] NTP monlist network traffic amplification attacks. *Meinberg Security Advisor. January 2014.* https://www.meinbergglobal.com/english/news/meinberg-security-advisory-mbgsa-1401-ntp-monlist-network-traffic-amplification-attacks.htm. Accessed August 2019.