Development and Study of a DNS Amplification Attack in a Virtual Environment

Van Pevenaeyge Amaury

Master thesis submitted under the supervision of
Debatty Thibault

in order to be awarded the Degree of
Master in Cybersecurity
Cryptanalysis and Forensics

Academic year
2023 – 2024

I hereby confirm that this thesis was written independently by myself without the use of any sources beyond those cited, and all passages and ideas taken from other sources are cited accordingly.

22/07/2024

Title: Development and Study of a DNS Amplification Attack in a Virtual Environment

Author: Van Pevenaeyge Amaury
Master in Cybersecurity – Cryptanalysis and Forensics
Academic year: 2023 – 2024

Abstract

This Master's thesis investigates the dynamics and impacts of DNS amplification attacks within a virtual environment. DNS amplification, a significant threat in the cybersecurity domain, exploits DNS protocol vulnerabilities to generate large volumes of traffic, potentially crippling target infrastructures. This study aims to provide a comprehensive understanding of these attacks and evaluate various mitigation strategies.

An optimized virtual laboratory was configured to simulate DNS amplification attacks under controlled conditions. This environment was designed to be highly vulnerable, facilitating detailed observations of attack mechanisms and impacts. Custom scripts were developed to accurately reproduce attack scenarios, measure amplification rates, and monitor DNS traffic on the victim side.

The analysis involved multiple series of measurements to compare findings with existing scientific literature and explore various factors influencing amplification rates. Mitigation techniques such as optimal DNS server configurations and rate limiting mechanisms were evaluated for their effectiveness in countering amplification attacks.

Future work suggestions encompass optimizing the developed scripts, enhancing the web monitoring interface, and applying machine learning algorithms to dynamically adjust mitigation measures based on legitimate user behavior and network load. The study also proposes expanding the virtual laboratory to simulate attacks on comprehensive enterprise networks and deploying these scenarios within the Royal Military Academy's Cyber Range, CyRange.

This thesis significantly contributes to the field of cybersecurity by offering practical solutions and insights for protecting systems against DNS amplification attacks, providing a foundation for future research and professional training in this critical area.

Keywords: DNS amplification, virtual environment, measurement, amplification factors

Preface

This Master's thesis in cybersecurity explores the complex dynamics and practical implications of a DNS amplification attack in a virtual environment. The aim of this work is to contribute to the understanding and prevention of this type of attack, which is becoming increasingly common and a threat to network security.

The initiative for this project was born of a deep interest in computer security and a desire to push back the current limits of our knowledge of cybersecurity. The research and development carried out in the course of this work required scientific rigor and a robust methodology for simulating and analyzing attacks in a controlled environment.

In the course of this dissertation, we discuss the theoretical foundations of the DNS protocol, the amplification mechanisms of DDoS attacks and possible mitigation measures. Through in-depth experimentation and analysis, we aim to offer fresh insights and practical solutions for strengthening system resilience against these threats.

This dissertation is the fruit of a collective effort and a personal commitment, aimed at making a significant contribution to the scientific community and to the practice of cybersecurity. I hope that readers will find in this document relevant information and food for thought for their own work and projects.

Acknowledgements

First of all, I'd like to thank the Royal Military Academy for allowing me to write this thesis. I would also like to thank my promotor, Professor Thibaut Debatty, for his help, availability and guidance in the preparation of this thesis.

I would also like to thank my internship supervisor Anthony Hannouille and the project manager of my internship team, Laurent Praxel, for pushing me to do my best to complete this thesis.

And finally, I'd like to thank my friends and family for supporting me, pushing me to excel and cheering me up during the many difficult moments of this thesis.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ADDoS	Amplified Distributed Denial-of-Service
ARP	Address Resolution Protocol
BAF	Bandwidth Amplification Factor
CnC	Command and Control
CPU	Central Processing Unit
DDoS	Distributed Denial-of-Service
DMZ	Demilitarized Zone
DNS	Domain Name System
DNSSEC	Domain Name System Security Extensions
DoS	Denial-of-Service
DRDoS	Distributed Reflection Denial-of-Service
ECDSA	Elliptic Curve Digital Signature Algorithm
FQDN	Fully Qualified Domain Name
HDD	Hard Disk Drive
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
ISC	Internet System Consortium
ISP	Internet Service Provider
KSK	Key-Signing Key
MTU	Maximum Transmission Unit
PAF	Packet Amplification Factor
RAM	Random Access Memory
RFC	Request For Comments
RR	Resource Record
RSA	Rivest–Shamir–Adleman
SHA	Secure Hash Algorithm
TCP	Transmission Control Protocol
TLD	Top-Level Domain
TTL	Time-to-Live
UDP	User Datagram Protocol
VDI	Virtual Disk Image
VHD	Virtual Hard Disk
VM	Virtual Machine
ZSK	Zone-Signing Key

Chapter 1
Introduction

1.1 Motivations

Cybersecurity has become a crucial issue in today's society, where computer networks and information systems are ubiquitous. DNS amplification attacks, in particular, represent a serious threat, as they exploit protocol flaws to generate huge volumes of traffic, which can paralyze entire infrastructures, including critical infrastructures such as hospitals. This study is part of an effort to gain a deeper understanding of the mechanisms, impacts and mitigation measures of these attacks, with the aim of reinforcing the security of IT systems.

There are many reasons for this research project. On the one hand, it aims to respond to a growing concern among IT industry players about the increase and sophistication of DDoS attacks. On the other hand, this dissertation aims to make a significant academic contribution by exploring innovative solutions and developing practical tools for the study of DNS amplification attacks, enabling us to learn how to detect and prevent these attacks. In addition, this dissertation is also intended to serve as an educational resource, enabling other students and professionals to gain a better understanding of this attack and how it works, and to encourage future work aimed at constantly improving knowledge of this threat and ways of protecting against it.

1.1.1 Context

The DNS (Domain Name System) is a fundamental element of the Internet infrastructure, enabling the resolution of domain names into IP addresses and vice versa. However, this same infrastructure can be exploited for amplification attacks, where small DNS queries can generate much larger responses, flooding targets with unwanted traffic. The DNS amplification attack is a particular form of DDoS (Distributed Denial of Service) attack, which takes advantage of the vulnerabilities of misconfigured DNS servers.

Historically, several notable DDoS attacks have highlighted the vulnerabilities of DNS systems. For example, in 2016, the DDoS attack against Dyn, a DNS management company, caused major disruptions to websites such as Twitter, Netflix, and Reddit. This attack mainly exploited vulnerabilities in security cameras and other IoT devices to generate massive traffic. Another notable attack was that of 2013, which targeted Spamhaus, an anti-spam organization, and is considered one of the largest DNS amplification DDoS attacks ever recorded, with traffic peaks of up to 300 Gbps. These incidents show just how devastating DNS amplification attacks can be, and underline the importance of ongoing research in this field.

In this context, the need for in-depth study and experimentation in a controlled environment is imperative. By simulating these attacks in a virtual environment, it is possible to understand the underlying mechanisms, carry out a series of measurements to assess the impact of such attacks, and point the way to possible mitigation measures against them.

This approach contributes to overall network security by offering solutions that can be applied in real-life contexts.

1.1.2 Problem statement

Despite advances in security, DNS amplification attacks remain a significant threat. They exploit inadequate DNS server configurations and the inherent functionality of the DNS protocol to generate massive attacks. The main problem lies in the difficulty of preventing and detecting these attacks before they cause significant damage. What's more, current mitigation solutions are not always sufficient or suitable for all situations.

This thesis focuses on the study of DNS amplification attacks in a virtual environment, highlighting the technical challenges and specific vulnerabilities associated with such attacks. It also aims to assess the strength of such attacks, and to initiate the study of various existing mitigation methods. This dissertation is the starting point for a series of other projects aimed at improving our knowledge of this attack and inventing or improving mitigation methods against it.

1.2 Project statement & contributions

The main objective of this project is to develop a detailed understanding of DNS amplification attacks and to evaluate some mitigation strategies proposed by various experts in the field. The specific contributions of this dissertation are as follows:

1. **Development of a virtual laboratory environment**: Setting up a virtual infrastructure to simulate and analyze DNS amplification attacks under controlled conditions. The infrastructure was configured to be as vulnerable as possible in order to optimally observe attack mechanisms and their impact. In addition, the DNS infrastructure has been configured in such a way as to reproduce and observe the results found in the scientific literature.

2. **Development of attack, measurement and monitoring scripts**: Scripts have been developed to reproduce this type of attack, to perform the various amplification measures and to monitor DNS traffic on the victim's side. These scripts enable faithful reproduction of attack scenarios and accurate data analysis.

3. **Analysis of amplification mechanisms**: Detailed study of the different types of DNS queries and their amplification potential. Several series of measurements were carried out to provide a variety of comparisons, including comparisons with the scientific literature and analysis of various factors influencing amplification rates.

4. **Evaluation of mitigation techniques**: Analysis of the effectiveness of certain mitigation measures, such as the optimal configuration of DNS servers and the implementation of mechanisms to limit amplification factors such as limiting the number of responses or minimizing the size of these responses.

5. **Suggestions for improvement**: Open up different avenues for possible improvement of this thesis, in particular by detailing the various works that could result from this thesis.

1.3 Organization of this document

This report is structured to guide the reader through the various stages of the study. Chapter 1 introduces the motivations, context, issues and contributions of the project. Chapter 2 presents a review of the scientific literature on the DNS protocol, DDoS attacks and existing mitigation methods. Chapter 3 describes the implementation of the virtual environment and DNS services used for the experiments. Chapter 4 details the implementation of the attack, measurement and monitoring scripts, and presents the results obtained. Chapter 5 explores some mitigation methods for DNS amplification attacks and analyzes their effectiveness. Chapter 6 suggests future work and avenues for improvement. Finally, Chapter 7 summarizes the main results and contributions of the dissertation.

Chapter 2
Literature review, state of the art (SotA)

2.1 DNS Protocol Review

First of all, it's worth taking a look at the purpose of the DNS protocol, the architecture behind it and the mechanisms behind it.

2.1.1 Purpose and structure of the DNS protocol

The DNS protocol translates domain names or host names into IP addresses. The domain name space, a major component of DNS, has a tree structure made up of nodes and leaves. Each node and leaf in the domain name space has a label and a set of information. This information is contained in records called Resources Records. It is possible for a node or leaf to contain no resource records. The tree is subdivided into zones starting at the root zone. [27]

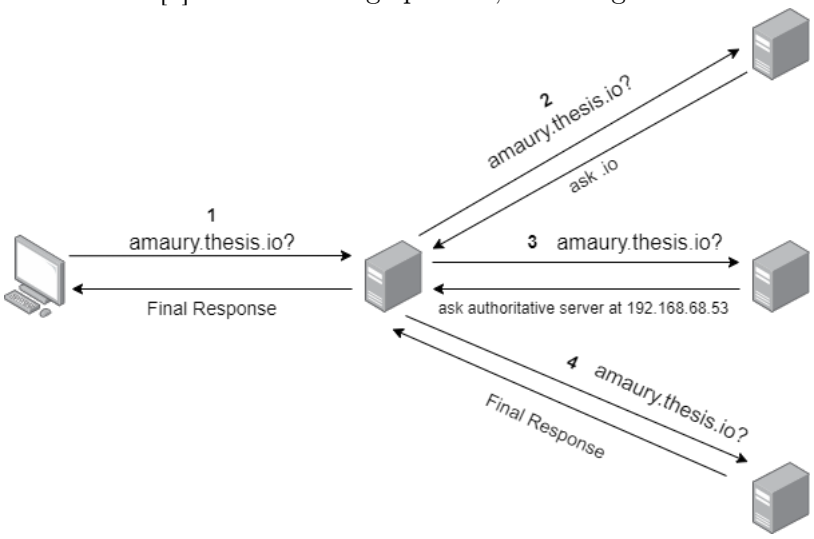
Definition

A DNS zone is a specific portion of the DNS namespace and is managed by an organization or administrator. This enables decentralized resource management. [6, 27]

DNS is in fact a distributed database using the client-server model. The nodes mentioned above are in fact name servers.[1] It is these name servers that provide information on a domain or host name. A very important name server is the authoritative name server. This name server knows the information relating to the zone it manages, and is authorized to respond to DNS requests for the zone it manages. [27]

2.1.2 Domain name resolution

Domain name resolution [5] is a multi-stage process, involving several DNS servers.



[1] https://en.wikipedia.org/wiki/Domain_Name_System

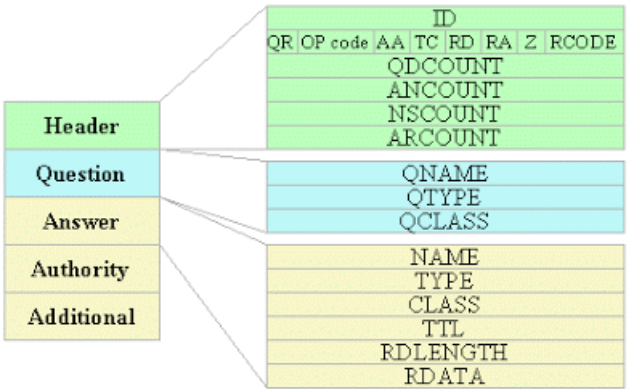
1. First of all, the customer queries his DNS resolver, generally by making a recursive request for the domain name he's looking for.
2. This DNS resolver then queries each DNS server required to obtain the final answer. It will start by querying the root name server, which is the reference server for other DNS servers containing the specific information.
3. If the information is not contained in the root name server, the latter will reply to the DNS resolver to query the TLD name server corresponding to the requested domain name. This TLD name server will send the DNS resolver a list of authoritative name servers for the requested domain name.
4. With this list of authoritative name servers for the requested domain, the DNS resolver will be able to query the correct DNS name server containing the information for the requested domain name, and thus obtain the information requested by the customer.
5. Once the information has been obtained, the DNS resolver sends it back to the client.

Side note

The client generally makes a recursive request to its DNS resolver, but the DNS resolver in turn makes iterative requests to other DNS servers to avoid overloading the other DNS servers.

2.1.3 DNS packet details

Before going any further and detailing the operation of the DNS protocol's request-response mechanism, it is necessary to detail the format of DNS messages.



All communications within the DNS protocol follow a format called a message. This message is made up of different sections, as can be seen in the previous image. [28]

The Header section is present in all messages. The Question section contains fields describing a question posed to a name server. Finally, the 3 other sections follow the same format: a set of resource records, which can of course be empty. The Answer section contains, as its name suggests, all RRs responding to the question asked. The Authority section contains RRs pointing to the authoritative name server(s) for the requested zone. And finally, the Additional section contains additional RRs relating to the request, but these RRs are not necessarily answers to the question posed. [28]

Side note

Details of the formats of the Header and Question sections and of the Resource Records making up the Answer, Authority and Additional sections can be found in RFC 1035. [28]

2.1.4 Transport mechanism

Messages are generally sent via the UDP protocol on server port 53. Initially, these messages were limited to a size of 512 bytes (excluding IP and UDP headers). However, the introduction of the EDNS0 extension made it possible to specify message sizes larger than 512 bytes. Messages exceeding the size limit were truncated (Truncated bit set to 1), meaning that transmission of the message was retried via TCP. As data sizes became increasingly large, the EDNS0 extension made it possible to extend this size and avoid this retransmission mechanism via TCP having to be constantly implemented. However, as mentioned in RFC 6891, this extension can also create a number of vulnerabilities, including a denial-of-service attack, which will be discussed in more detail later in this report. [28]

Side note

As mentioned in RFC 6891, the EDNS extension is a hop-by-hop extension, which means that its use must be negotiated between all servers involved in the resolution process. [14]

Messages can also be sent via TCP on server port 53. TCP is generally used when Truncated bit is enabled. This protocol is also used for zone file transfers between the primary name server and the secondary name server. [28]

Definitions

The **primary name server** is the authoritative server hosting the master copy of data for the DNS zone for which it is authoritative. This server is also responsible for the management and direct updates of its zone. [7]

Secondary name servers obtain read-only copies of zone files from the primary name server. These secondary name servers enable data redundancy and load balancing by also responding to requests for the zone concerned. [7]

2.2 DNSSEC Protocol Overview

Before starting, it's very important to note that the details in this section are based on Cloudflare's article detailing the DNSSEC protocol. [8]

On September 10, 2014 researchers at Carnegy Mellon University published an article in which they detailed having discovered that emails destined for Yahoo!, Hotmail and Gmail servers were traveling through unauthorized mail servers, in other words that a kind of Man-In-The-Middle was in place. The DNS protocol vulnerability exploited here is that authentication information is not verified by the domain name system before accepting a response. [33]

Definitions

A **Man-In-The-Middle Attack** is a technique whereby an attacker secretly inserts himself between the communications of two parties, intercepting and potentially modifying the data exchanged.

The solution to this problem is provided by the DNSSEC protocol. This protocol secures the authenticity of DNS protocol information. To fully understand how the DNSSEC protocol works, it's essential to detail several components: the new DNS record types added, the RRsets, the various keys, the Delegation Signer records and the chain of trust. Once these details have been covered, domain name resolution with DNSSEC can be explained.

2.2.1 Added DNS record types

The DNSSEC protocol strengthens the security of the domain name system by integrating cryptographic signatures into the DNS records already in place. DNS name servers retain these digital signatures along with the usual record types such as A, AAAA, MX, CNAME, TXT, etc. By checking the signature associated with a DNS record, it is possible to verify that the origin of this record is indeed its authoritative name server, and that it has therefore not been modified by a Man-in-the-Middle attacker. To simplify signature verification, DNSSEC introduces several new DNS record types:

- **RRSIG:** a cryptographic signature of the corresponding RRset.
- **DNSKEY:** a public signing key.
- **DS:** cryptographic hash of a DNSKEY record
- **NSEC and NSEC3:** for explicit denial of DNS record existence. With DNSSEC, DNS zones can include cryptographic proofs, in the form of NSEC (Next Secure) or NSEC3 (Next Secure version 3), which explicitly specify which records exist between two consecutive records in a secure DNS zone. These proofs provide irrefutable evidence that a certain DNS record does not exist in a given zone. Thus, "explicit denial of existence" means that the DNS server can verifiably prove that a specific record does not exist in the DNS zone.
- **CDNSKEY and CDS:** when a child zone requests updates for one or more DS records in the parent zone.

2.2.2 RRsets

The first step in implementing the DNSSEC protocol is to group DNS records of the same type into a set of RRs called an RRset. For example, if a zone contains three records of type AAAA with the label www.example.com, these records will be grouped into a single AAAA RRset. It is this RRset that will be signed, and will then get an associated RRSIG record.

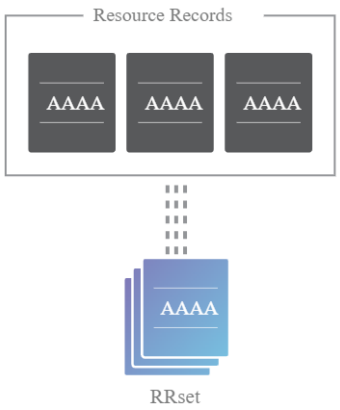


Figure 2.1: RRset illustration [8]

2.2.3 Zone-Signing Keys and Key-Signing Keys

The DNSSEC protocol introduces 2 types of signing keys: zone-signing keys and key-signing keys. Each zone is provided with a zone-signing key pair called ZSK. The private ZSK will sign each RRset of the zone concerned, while the public ZSK will verify the signature of these RRsets. Each RRSIG obtained as a result of the RRset signing process, thanks to the private ZSK, is stored in the corresponding name server. Each RRSIG allows to specify that the corresponding DNS records belong to it, that their origin is the name server holding this RRSIG, and what these records should look like.

The DNS zone administrator will also publish its public ZSK within its name server in a DNSKEY resource record. When a DNSSEC resolver requests a particular record, the name server will return the RRSIG associated with that record, along with the zone's DNSKEY record to obtain the zone's public ZSK. It is this triplet [RRset, RRSIG and public ZSK] that validates the DNS response.

However, this public ZSK must also be validated to ensure that it has not been compromised. This public ZSK is validated by introducing a key-signing key pair. This works in a similar way to signing a DNS zone. The private KSK will be used to sign the zone's public ZSK, producing an RRSIG for the DNSKEY record corresponding to this public ZSK. The name server will also publish its public KSK in a DNSKEY-type RR at the same time as signing this public KSK using the private KSK, thus producing an RRSIG. Resolvers are now able to validate the public ZSK registration thanks to the public KSK registration.

2.2.4 Delegation Signer Records

All that remains now is to establish trust in the rest of the DNS hierarchy. The Delegation Signer record will establish this trust. This record is in fact the hash, produced by the child zone, of the DNSKEY record containing the zone's public KSK. This record is then transmitted to the parent zone so that it can publish this DS record. To verify the public KSK of a child zone, the resolver will hash this KSK and compare this hash with the DS record supplied by the parent zone. If the two hashes match, then the public KSK is validated and therefore trustworthy.

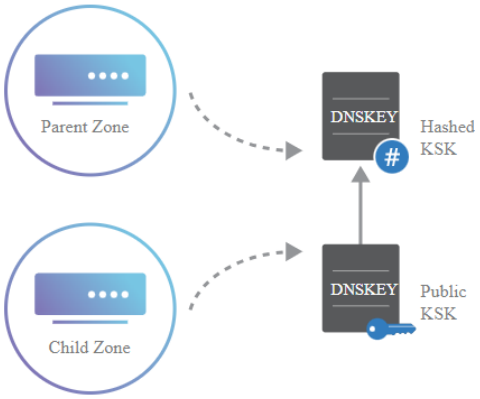


Figure 2.2: Delegation Signer Record illustration [8]

2.2.5 The chain of trust

The DNSSEC protocol therefore introduces the notion of a chain of trust through the various keys and DS records. To establish this chain of trust, it is necessary to repeat the validation process described above, right down to the root zone. It's important to note that there is no DS record for the root zone, so there's a ceremony called the Root Signing Ceremony [9] to produce the RRSIG record that can be used to verify the root name server's public KSK and ZSK.

Side note

The illustration of the chain of trust mechanism can be found in the appendix.

2.2.6 Name resolution and validation with DNSSEC

Now that all the elements of the DNSSEC protocol have been introduced, it's time to explain how DNSSEC-based name resolution works in practice.

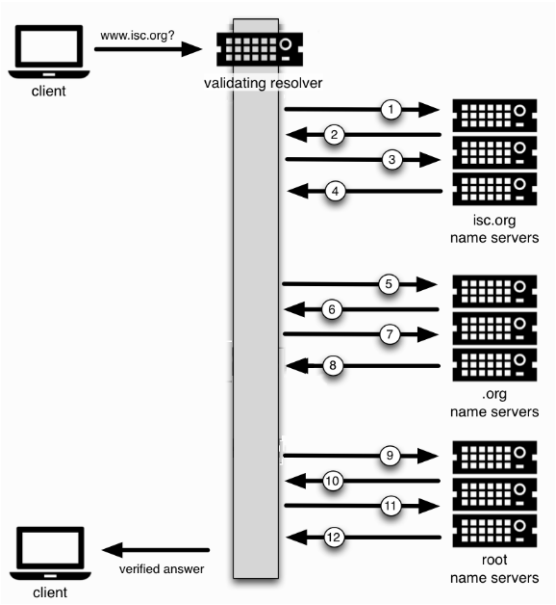


Figure 2.3: Name resolution and validation with DNSSEC [22]

1. When the validating resolver receives a DNS request with the DNSSEC flag set, meaning that the client wants DNSSEC responses, this resolver will follow the standard DNS protocol to find the name server for isc.org and will ask it to send a DNSSEC request to obtain the type A record from www.isc.org.

2. The isc.org name server supporting the DNSSEC protocol will respond with the type A record and the corresponding RRSIG so that the validation process can proceed.

3. The validating resolver will ask isc.org for its cryptographic keys in order to validate the RRSIG obtained.

4. The isc.org name server will respond with DNSKEY records and corresponding RRSIGs. The validating resolver will now be able to verify the responses received in step 2.

5. The validating resolver will ask the .org name server for all the information needed to verify its isc.org child.

6. The .org name server will respond with the corresponding DS record and the corresponding RRSIG. The validating resolver can now calculate the hash of isc.org's public KSK and compare it with the DS record returned by .org. If the 2 match, this will prove the authenticity of isc.org.

7. The validating resolver will ask .org for its DNSKEY records in order to verify the signatures it received in step 6.

8. .org will respond with its DNSKEY records and associated RRSIGs. The validating resolver can now verify the answers obtained in step 6.

9. The validating resolver will now ask the root zone for all the information needed to verify the authenticity of .org.

10. The root name server will then return the corresponding DS record and the associated RRSIG. The validating resolver will repeat the same process as in step 6 to validate the authenticity of .org.

11. The validating resolver will request DNSKEY records from the root zone in order to validate the information received in step 10.

12. The root name server will respond with the associated DNSKEY and RRSIG records. The validating resolver, which trusts the root zone thanks to the Root Signing Ceremony concept attesting to the root zone's identity, will validate the responses received in step 10 and consequently validate the entire chain of trust, which will at the same time validate the validity of the response. The validating resolver will then send the verified response to the client.

2.3 Distributed Denial-of-Service Attack (DDoS)

Before describing a DDoS attack in detail, it's important to define a DoS attack. A DoS attack is an attack designed to alter or even render unavailable the services of a machine or network. The attack does not affect the data itself, but rather its availability. Most DoS attacks aim to saturate the target's bandwidth or connectivity. To achieve this, the

attacker will generally send such a flow of traffic or requests that it will consume all available network resources, preventing legitimate users from accessing the network or machine normally. [16]

A DDoS attack is a distributed DoS attack, i.e. one launched from multiple devices. During this type of attack, the attacker will generally use a set of machines to launch DoS attacks from each of these machines.

This type of attack has particularly flourished since the emergence of botnets. Botnets are a collection of infected machines known as bots or zombies, controlled by an attacker. The creation of such a botnet has become the activity of choice for attackers since the emergence of the IoT and all connected devices. Typically, attackers will attempt to infect as many machines as possible using malware such as Mirai [2]. They then control and coordinate DDoS attacks from a platform known as the Command and Control Center. These botnets enable attackers to launch large-scale DDoS attacks by exploiting the resources of infected machines, thus saving the attacker a great deal of resources. [15, 34] Obviously, the larger the botnet, the more powerful the attack will be, as the network of bots will be able to send a large number of requests or generate a huge network flow. [3, 16]

In these DDoS attacks, we can find several classes of attacks, including the Flooding Attack, the Reflection Attack, the Coremelt Attack, the Land Attack, the Amplification Attack and many others. Among all these attack classes, the best known are probably the flooding attack, the reflection attack and the amplification attack. [15]

Flooding attacks consist in sending a very large amount of traffic to a victim using bots, also known as zombies. [15] These flooding attacks include such popular attacks as:

- **SYN flood attack:** This attack consists of repeatedly sending a large number of SYN packets. The attacker is attempting to exploit a vulnerability in the TCP protocol. When the server receives a SYN packet, it responds with a SYN/ACK packet, allocating the resources required for the connection while waiting for the final stage of the connection, i.e. the reception of an ACK packet. As a result, the server will receive a large number of SYN packets from the attacker, and will allocate resources for each requested connection without ever finalizing them, thus consuming all its available resources. [11]

[2] https://github.com/jgamblin/Mirai-Source-Code

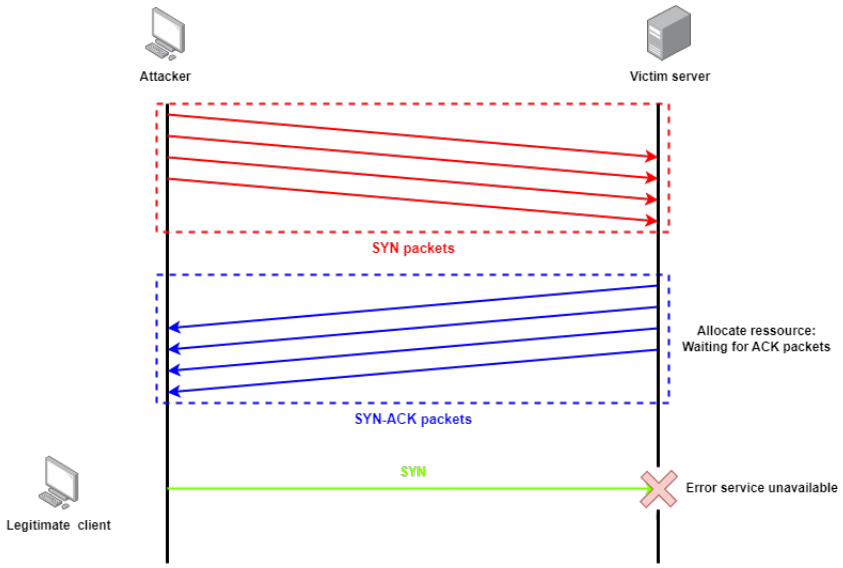


Figure 2.4: SYN Flood

- **HTTP flood attack:** This attack aims to overload a server by sending a large number of HTTP requests. There are 2 types of HTTP flood attack: HTTP GET attack and HTTP post attack. The idea of the HTTP GET attack is to flood the server with GET requests. This type of request typically asks the server to return a specific resource to the client. This mass of GET requests will therefore cause the HTTP server to consume large amounts of resources, thereby overloading the server. The idea behind the HTTP POST attack is similar to that of the HTTP GET attack, but with POST requests. POST requests are used to send data to the server. The server must then manage and process this data. As a result, this mass of POST requests will consume a huge amount of resources on the server side, overloading it. [10]

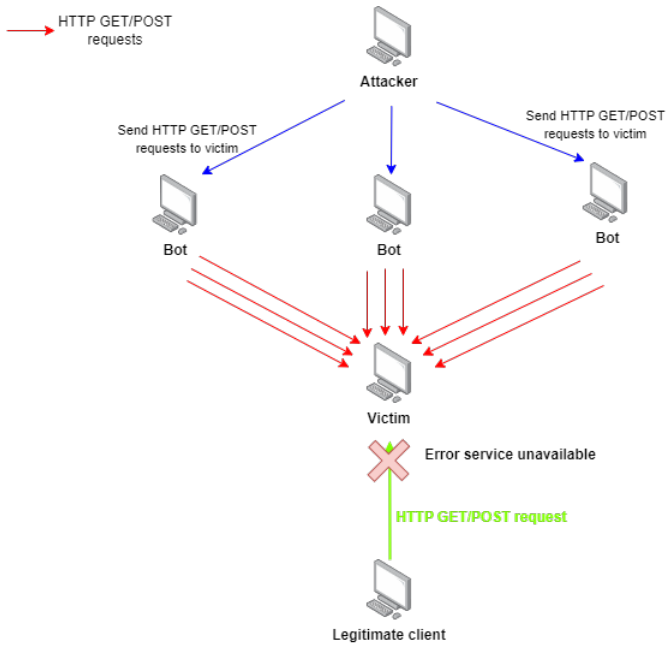


Figure 2.5: HTTP Flood

- **UDP flood attack:** This attack simply involves sending a large number of UDP packets. Since the UDP protocol does not require a connection to be established, the attacker can easily send a huge number of packets via his botnet. When receiving a UDP packet on a specific port, the server will consume resources by checking whether a program is running on the specified port and replying to the client with an ICMP packet specifying whether the destination is reachable or not. It is this verification-response mechanism that will cause the server to overload when faced with a large number of UDP requests. [12]

A reflection attack is a special type of DoS attack. During a reflection attack, the attacker will launch a DoS or DDoS attack by spoofing the victim's IP address in order to direct the attack towards the victim. Next, the attacker will typically direct the destination of the requests to a server acting as a reflector. This server will then direct all responses to the victim, as its address will have been usurped by the attacker. The difference between reflection attacks and classic DDoS attacks is that reflection attacks don't target the destination directly, but rather use the destination to reflect all the requests that can be sent to it. The target here is really the spoofed source IP address.

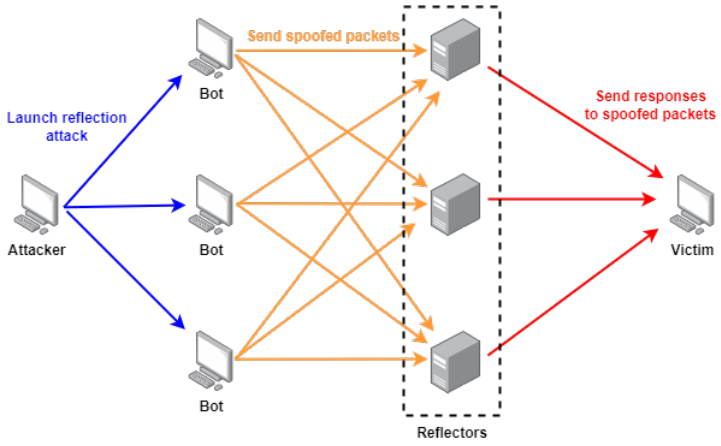


Figure 2.6: Reflection Attack

It's very important not to confuse reflection attacks with amplification attacks. In fact, the difference between these two attacks is that reflection attacks are not amplified and therefore have an amplification factor equal to one, but this will be detailed in the next section.

2.4 Amplification DDoS attack

As mentioned above, amplification attacks use the same mechanism as reflection attacks, with the difference that amplification attacks use protocols that amplify network traffic. What is meant here by amplifying network traffic is that the size of the responses sent from the server to the victim will be larger than the size of the requests sent by the attacker to the server. The server will therefore be called an amplifier instead of a reflector. To add even more detail, reflectors will only allow IP address spoofing on the part of the attacker, whereas amplifiers will allow the victim's IP address to be spoofed at the same time as sending responses of a much larger size than the requests initially received. [23, 32]

Before launching a DDoS amplification attack, an attacker will first seek to collect a list of amplifiers enabling the attack to be carried out. Next, the attacker will gather information about these amplifiers, such as the amplifier's maximum bandwidth, whether there is a waiting mechanism between responses or whether a blacklisting mechanism is in place. Once all this information is in hand, the attacker can launch his attack. [23]

Amplification attacks are often carried out via the UDP protocol, since this makes it easy to spoof the victim's IP address. The TCP protocol is actually not preferred, as it has a three-way handshake mechanism initially in place to validate the identity of the entity behind the communication. However, the article of Ismail and al [23] based on the article of Kührer and al [24] shows that the TCP protocol can be used to carry out amplification attacks, and in particular highlights the protocols using TCP that can be used to carry out this type of attack.

Table 2 – List of TCP based protocols used in ADDoS (Kührer et al., 2014b).

Protocol	Port(s)
FTP	21
HTTP	80
IMAP	143
IPP	631
IRC	6667
MySQL	3306
NetBIOS	137
NNTP	119
POP3	110
SIP	5060
SMTP	25
SSH	3389
Telnet	23

Figure 2.7: TCP based protocols for amplification attacks [23]

The article of Ismail and al [23], again based on the article of Kührer and al [24], highlights UDP-based protocols for amplification attacks. These are the protocols generally favored for amplification attacks, since the UDP protocol allows attackers to hide by usurping the victim's IP address.

Table 1 – UDP protocol (Rossow, 2014).

Category	Protocol	Port(s)	Description
Network Services	SNMP v2	161	Monitoring network-attached devices
	NTP	123	Time synchronization
	DNS	53	(Primarily) Domain name resolution
	NetBios	137	Name service protocol of NetBios API
	SSDP	1900	Discovery of UPnP-enabled hosts
Legacy Protocols	CharGen	19	Legacy character generation protocol
	QOTD	17	Legacy "Quote-of-the-day" protocol
P2P File Sharing	BitTorrent	Any	BitTorrent's Kademlia DHT implementation
	Kad	Any	eMule's Kademlia DHT implementation
Multiplayer Games	Quake 3	27,960	Games using the Quake 3 engine
	Steam	27,015	Games using the Steam protocol
P2P Based Botnets	ZAv2	164	P2P-based rootkit
	Sality	Any	P2P-based malware dropper
	Gameover	Any	P2P-based banking trojan

Figure 2.8: UDP based protocols for amplification attacks [23]

When we talk about amplification attacks, there is one measure that is important in the choice of the protocol to be used in the attack. This measure is called the amplification rate and depends on a number of factors, including the target's protocol environment and all the measures it has taken to defend itself against these attacks. In reality, the amplification rate is divided into 2 amplification rates: the Bandwidth Amplification Factor (BAF) and the Packet Amplification Factor (PAF). According to article of Ismail and al [23], article

of Rossow and Görtz [32] and the article of Anagnostopoulos and al [3], the size of packet headers should not be taken into account when calculating amplification rates. Their justification is that this size may change if the protocol used evolves. However, a choice will be made later in this thesis on this subject. The calculations for these amplification rates are as follows:

- The Bandwidth Amplification Factor is the ratio between the size of the UDP payload sent by the amplifier to the target and the size of the UDP payload sent by the attacker to the amplifier.
- The Packet Amplification Factor is the ratio between the number of packets sent by the amplifier to the target and the number of packets sent by the attacker to the amplifier.

In addition, Rossow and Görtz's article [32] presents a study of these two amplification rates for UDP-based protocols enabling these amplification attacks.

Protocol	BAF *all*	BAF 50%	BAF 10%	PAF *all*	Scenario
SNMP v2	6.3	8.6	11.3	1.00	*GetBulk* request
NTP	556.9	1083.2	4670.0	3.84	Request client statistics
DNS_{NS}	54.6	76.7	98.3	2.08	ANY lookup at author. NS
DNS_{OR}	28.7	41.2	64.1	1.32	ANY lookup at open resolv.
NetBios	3.8	4.5	4.9	1.00	Name resolution
SSDP	30.8	40.4	75.9	9.92	*SEARCH* request
CharGen	358.8	n/a	n/a	1.00	Character generation request
QOTD	140.3	n/a	n/a	1.00	Quote request
BitTorrent	3.8	5.3	10.3	1.58	File search
Kad	16.3	21.5	22.7	1.00	Peer list exchange
Quake 3	63.9	74.9	82.8	1.01	Server info exchange
Steam	5.5	6.9	14.7	1.12	Server info exchange
ZAv2	36.0	36.6	41.1	1.02	Peer list and cmd exchange
Sality	37.3	37.9	38.4	1.00	URL list exchange
Gameover	45.4	45.9	46.2	5.39	Peer and proxy exchange

Figure 2.9: Amplification factors for some UDP based protocols [32]

The article of Ismail and al [23] also produced a taxonomy of ADDoS. It states that this type of attack can be divided into two broad categories: flow multiplication attacks and payload magnification attacks. Each of these categories then contains a series of known attacks, such as the Smurf attack, or a series of known protocols, such as TCP and UDP. In this article, you can also find a table from the article Kührer et al showing the number of amplifiers and amplification rates for the TCP protocol.

Table 6 – Result of TCP based ADDoS by Kührer et al. (2014b).

	SYN/ACK		PSH		RST	
Protocol	# Amplifiers	AF	# Amplifiers	AF	# Amplifiers	AF
FTP	2,907,279	22x	274	103x	5,577	53,927x
HTTP	421,487	60x	241	147x	3,411	432x
NetBIOS	8,863	54x	64	71x	3,087	78,042x
SIP	16,496	1,596x	2	696x	6,306	32,411x
SSH	81,256	80x	391	57x	5,889	29,705x
Telnet	2,112,706	28x	2,353	3,272x	4,242	79,625x

Figure 2.10: Amplification factors for some TCP based protocols [23, 24]

And a final point to note about these amplification attacks is that they can be carried out using a botnet, but also without a botnet. The advantage of the botnet here is that the attacker will use the botnet's resources to carry out the attack, sending all the requests

where he would use his own resources without a botnet. However, a botnet requires a certain architecture and specific mechanisms.

2.5 DNS Amplification Attack

A DNS amplification attack is a particular type of amplification attack. In this attack, the attacker takes advantage of the fact that the size of DNS queries is smaller than the size of DNS responses. The attacker will, through a botnet or not, use one or more machines to act either as DNS servers or DNS forwarders, sending DNS responses back to the victim. This type of attack is particularly popular with attackers for a number of reasons. Firstly, it allows the attacker to be "anonymous", since the UDP protocol can be used to spoof the victim's IP address. Secondly, as DNS is used worldwide, it provides a large attack surface and a wide choice of victims. And finally, if the victim has little or no protection against this attack, the DNS amplification attack is fairly easy to carry out.

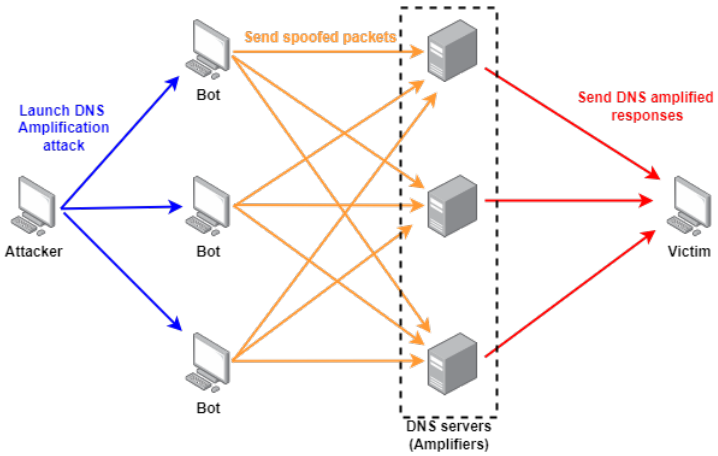


Figure 2.11: DNS Amplification Attack

In an article by Anagnostopoulos and al [2], you can find a schematic of the typical architecture of a DNS amplification attack. In this schematic, it is interesting to note that an attacker could typically set up an authoritative DNS server containing a large number of DNS records, with the aim of generating a very large response. In this architecture, a botnet would be used to send DNS queries to several recursive DNS servers, potentially including Open DNS servers. These queries would typically contain a request to resolve the domain name set up by the attacker. The DNS servers receiving these requests would then query the attacker's DNS server to resolve the domain name, eventually returning an amplified response to the victim.

In the article by Anagnostopoulos and al [2], it's also possible to find a very important point about the DNS amplification attack. The adaptation of the EDNS0 extension has made it possible to create responses much larger than the initial 512-byte limit. This extension therefore considerably increased the amplification rates of the DNS amplification attack and consequently considerably increased the impact of this attack. This article also mentions that an attacker will generally place a very large TXT DNS record (around 4KB) in order to maximize the size of the DNS response and therefore maximize the amplification rate of the attack. Still in the same article, the authors present the attack scenario ac-

companied by pseudo code using the Scapy library of the Python programming language. Next, the authors describe the results of executing their attack scenario in 3 countries: Greece, Ireland and Portugal. In particular, these results concern the percentage of open forwarders per country in regards to the size of response they return. These results also concern the impact on performance for the victim.

In another article by Anagnostopoulos and al [3], the authors explain their process of discovering Open DNS Resolvers and DNS forwarders, which is a crucial step in the development of a DNS amplification attack. Indeed, as seen previously, one of the first steps in this type of attack is to collect a set of amplifiers to be used in the attack in question. In this study, it is important to note that the authors once again carried out their study in Greece and Portugal, but this time replaced Ireland with Singapore. The authors also carry out a study of amplification rates for different types of queries with different parameters. This article is also interesting in that it mentions that the DNSSEC extension considerably increases the Banwidth Amplification Factor. This is because the DNSSEC extension introduces signatures and public keys into DNS responses, thereby increasing the size of these responses and thus the BAF. A final important point made by this article is that the scientific literature, in particular the article about the influence of TLDs in a DNS amplification attack [1], reveals that the use of Top-level domains would once again increase the BAF. Finally, like many scientific studies, this article studies BAF for different query types.

Finally, the article by van Rijswijk-Deij and al [31], highlights and studies the fact that using the DNSSEC extension in a DNS amplification attack considerably increases the BAF, particularly with ANY queries. This increase in BAF comes, as mentioned above, from the fact that the DNSSEC extension will add a series of DNS records including signatures (RRSIGs) and public keys (DNSKEY). This addition of DNS records will increase the impact of ANY queries. Indeed, this type of query will return all the DNS records available for the requested domain name, and the response will therefore include all the signatures and keys for the zone. This will considerably increase the size of DNS responses, the BAF and consequently the impact of the attack.

2.6 Mitigation mesures against DNS Amplification attacks

Side note

In this section, when any type of DNS server is mentioned, it will be referring to DNS relay servers, or in other words the DNS servers used to amplify this attack, as shown in Figure 2.11.

In addition, the victim here can be another server of any type, an individual machine or an entire network.

As mentioned in an article written by Cloudflare [4], a single individual or company has limited defense capabilities against DNS amplification attacks. Indeed, the effects of such an attack will be felt across the entire infrastructure surrounding the actual target of the attack. Apart from defense solutions offered by external services, the main mitigation strategies are preventive measures. In fact, Cloudflare offers two preventive measures against this type of attack: reduce the total number of open DNS resolvers and check the source IP address. It's important to note that not leaving your DNS server open to the

Internet is a best practice documented by BIND9. Indeed, reducing the number of open DNS servers makes it more difficult for attackers to find amplifiers to carry out their attack. Source IP addresses will be verified by Internet Service Providers (ISPs). Cloudflare's logic here is that a DNS packet sent from inside a network with a spoofed source IP address that makes it look like an external host will surely constitute a spoofed packet and can therefore be dropped. Finally, Cloudflare has solutions to mitigate DNS amplification attacks. In fact, its Anycast[3] network and other solutions[4] enable the weight of the attack to be spread over numerous datacenters, and thus over a larger surface area, to better absorb the load of this type of attack.

Fortinet's article [17] details 10 ways in which their FortiDDoS solution can mitigate a DNS amplification attack and thus protect your DNS infrastructure.

First of all, the authors advise against allowing DNS responses that have no corresponding DNS query, i.e. DNS responses that have not been requested. A classic DNS exchange consists of sending a DNS request from a DNS resolver to a DNS server, followed by a response from the DNS server to the DNS resolver. A legitimate DNS response will therefore always be preceded by a DNS request. Based on this principle, their solution analyzes whether each DNS response has an associated DNS query.

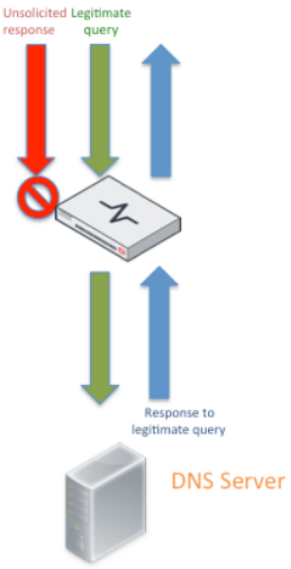


Figure 2.12: Blocking unsolicited DNS responses [17]

Their second recommendation is to drop retransmissions that are too fast. Indeed, a legitimate DNS client will never send the same DNS query many times consecutively within a short time interval. Their logic is therefore that if the same DNS query arrives too quickly from the same IP address, such DNS queries will simply be dropped.

[3] https://blog.cloudflare.com/a-brief-anycast-primer/
[4] https://www.cloudflare.com/fr-fr/ddos/

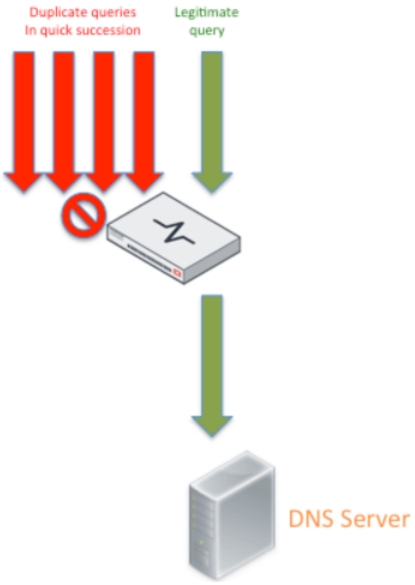


Figure 2.13: Drop Quick Retransmissions [17]

Secondly, the authors also recommend applying the Time-to-Live (TTL) principle, i.e. not accepting the same requests too quickly if the DNS server has recently sent the response to this request. Indeed, if a DNS client receives a DNS response to its query, this client will cache it thanks to the Time-to-Live specified in the response, and will therefore normally not need to request this information again, since it will already contain it in cache.

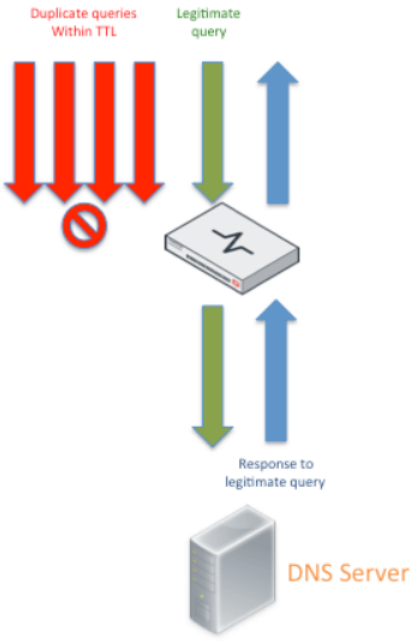


Figure 2.14: Enforce TTL [17]

It is also recommended not to accept DNS requests or DNS responses that are abnormal. Indeed, as this type of attack is most often carried out via scripts, these may contain bugs and therefore build incorrect or non-standard DNS packets. Such malformed packets should therefore be dropped.

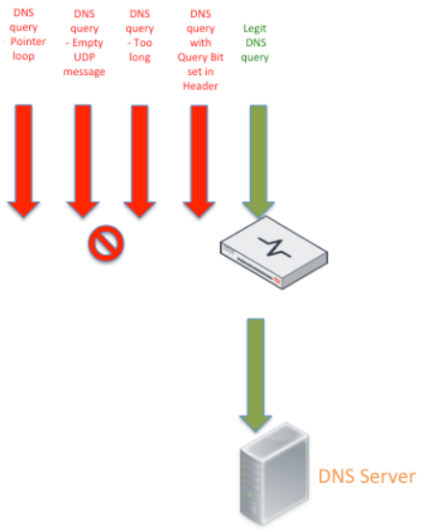


Figure 2.15: Drop Anomalous DNS Packets [17]

Their solution forces the DNS client to prove that it is who it claims to be, and therefore that it has not been spoofed. To do this, they will force retransmissions via TCP. Fortinet also recommends using the Access Control Lists mechanism. In fact, a very important feature of this mechanism is its ability to filter traffic by IP address, and thus block IP addresses for which you do not wish to authorize traffic.

Articles from the Internet System Consortium (ISC) [20, 21] and Imperva [18] also recommend adopting best practices for configuring DNS servers to make them less vulnerable and more robust against this type of attack. Examples of best practices include limiting recursive queries to authorized IP addresses, or limiting the size and rate of DNS responses. The Imperva article also mentions setting up rate-limiting mechanisms to limit the volume of DNS traffic. An article by America's Cyber Defense Agency (CISA) [13] also supports the implementation of best practices for configuring DNS servers, implementing mechanisms such as source IP address verification, and advising the implementation of Response Rate Limiting mechanisms, while cautioning that this mechanism could also impact legitimate traffic.

The article by MacFarland et al [25] mentions that the US CERT (United States Computer Emergency Readiness Team) recommends the use of this Response Rate Limiting mechanism with a limit of 5 identical responses per second for the same origin. This article also mentions that the US CERT agrees that this mechanism could also impact legitimate traffic and therefore cause unanswered DNS responses.

Finally, RFC8482 [19] recommends minimizing responses to ANY DNS queries. Cloudflare implements this mechanism and details part of its implementation in its article [26] on the subject. The idea would be to provide a special HINFO record mentioning RFC8482 and possibly the title of this RFC.

Definition

A DNS Record HINFO (Host Information) is used to store information about a host's hardware and software characteristics.

Chapter 3
Configuration of virtual lab environment and DNS services

The main problem addressed in this thesis is the difficulty of preventing and detecting DNS amplification attacks before they cause significant damage. Current mitigation solutions are not always sufficient or suitable for all situations, hence the need for this in-depth research. The development of a particularly vulnerable virtual laboratory environment is essential to understand the underlying mechanisms of these attacks and assess the effectiveness of different mitigation strategies.

For the purposes of this thesis, it was crucial to set up an infrastructure for simulating and analyzing DNS amplification attacks under controlled conditions. This infrastructure was configured to be as vulnerable as possible, in order to optimally observe attack mechanisms and their impact. In addition, it has been designed to reproduce and observe results found in the scientific literature, thus providing a solid basis for empirical comparisons and validating theoretical observations through practical experimentation.

The state of the art highlighted the importance of understanding the DNS protocol and the vulnerabilities exploited by DDoS attacks, particularly amplification attacks. Notable incidents such as the attacks on Dyn in 2016 and Spamhaus in 2013 have demonstrated the seriousness of these threats and the potentially devastating impact of DNS amplification attacks. By reproducing these attack scenarios in a virtual environment, it is possible to explore amplification factors in depth and assess the effectiveness of proposed mitigation measures.

The detailed configuration of this virtual environment and DNS services is therefore a crucial step in addressing the issues raised and achieving the objectives defined in this thesis.

3.1 Set up the virtual laboratory environment

3.1.1 Hypervisor selection

Before deploying any virtual environment, it was necessary to choose a hypervisor to run the entire laboratory. VirtualBox was chosen for several reasons.

Firstly, as VirtualBox is open source software, it was obvious that it would be a good choice for the development of a master's thesis. What's more, its active community means that the software is well maintained and responsive to bug fixes.

Secondly, VirtualBox offers a range of additional features compared with VMware Workstation Player. VirtualBox enables Software Virtualization. VirtualBox naturally supports more Host and Guest Operating Systems, such as Solaris, FreeBSD and macOS,

where VMware requires VMware Fusion. VirtualBox also enables snapshots to be taken, whereas VMware only offers this functionality with the paid version of the product. VirtualBox supports more Virtual Disk Formats such as VDI, VHD and HDD. VirtualBox also offers, free of charge, all the Virtual Network Models needed to develop any type of lab, whereas you'd have to pay to access such features with VMware. And finally, VirtualBox natively offers a wide range of integrations such as Vagrant and Docker, whereas VMware requires an additional conversion utility for more VM types. [29]

And finally, as VirtualBox is widely used in the student community, it seemed a wise choice for the development of a laboratory for observing and experimenting with a DNS Amplification attack.

3.1.2 Topology and Set up of virtual environments

3.1.2.1 Laboratory Topology

Before going into detail about setting up the laboratory environment, we need to describe its topology. This virtual laboratory will contain 3 virtual machines:

- A machine representing the DNS server to be used in the attack. It is through this DNS server that all DNS queries intended to overload the victim will pass. This server will also be the subject of amplification rate studies in the remainder of this thesis.
- A machine representing the attacker, which will be used to run the script launching DNS queries used to measure amplification rates, as well as the attack script. This machine will also act as a DNS client.
- A machine representing the victim of the attack, which will be used to observe DNS traffic via a web interface. This machine will also act as a DNS client.

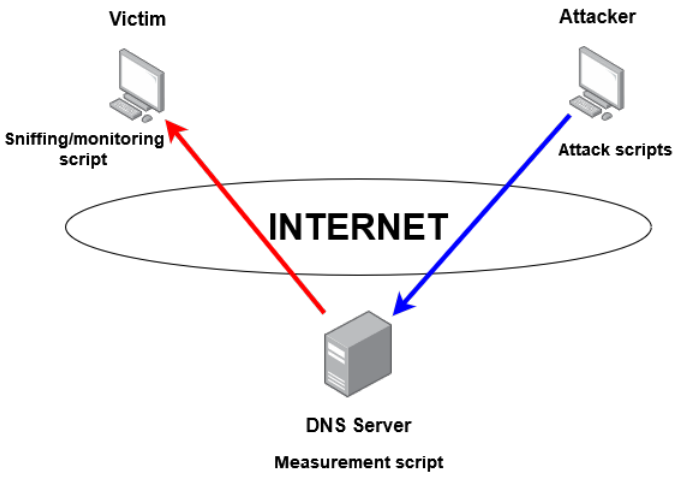


Figure 3.1: Laboratory Topology

3.1.2.2 Laboratory set-up

Given the popularity and ease of use of the Ubuntu 22.04 LTS Jammy Jellyfish operating system, this was chosen as the operating system for the various virtual machines. This

choice also seemed judicious in view of the students' familiarity with this operating system. Once this choice of operating system had been made, it was necessary to make choices concerning the virtual machine configuration. These choices were as follows:

- RAM: 2GB.
- CPU: 1.
- VDI: normal 25GB.
- Network Interface: one single network interface in bridge access mode. In the advanced settings, the interface type is Intel PRO/1000 MT Desktop (82540EM), the promiscuity mode is set to Allow All and the connected cable box is checked.

Once the virtual machines had been correctly configured, Guest Additions were installed to ensure that there would be no future problems with any action, and above all to obtain additional functionalities such as automatic screen adjustment and the creation of a shared folder on the virtual machine to serve as a DNS server. To achieve this, several installation steps[1] had to be followed:

1. Installing requirements.
2. Go to the Devices tab and select "Insert Guest Additions CD Image", which mounts the Guest Additions image in the virtual machine.
3. Go to the folder containing the Guest Additions image.
4. Install VirtualBox Guest Additions.
5. Restart the virtual machine.

```
sudo apt update
sudo apt install build-essential linux-headers-$(uname -r) -y
cd /media/<username>/VBox_Gas_7.0.12/
sudo ./VBoxLinuxAdditions.run
sudo reboot
```

Once these Guest Additions had been installed, a shared folder[2] was created for direct access to the project code.

The next step was to configure a static IP address for the virtual machine containing the DNS server. The IP of a DNS server is never supposed to change. To achieve this, we had to follow a number of steps:

1. Modify the netplan configuration file with the configuration shown in the following image.
2. Apply changes.

```
sudo nano /etc/netlan/01-network-manager-all.yaml
sudo netplan apply
```

[1] https://www.linuxtechi.com/install-virtualbox-guest-additions-on-ubuntu/
[2] https://carleton.ca/scs/tech-support/troubleshooting-guides/creating-a-shared-folder-in-virtualbox/

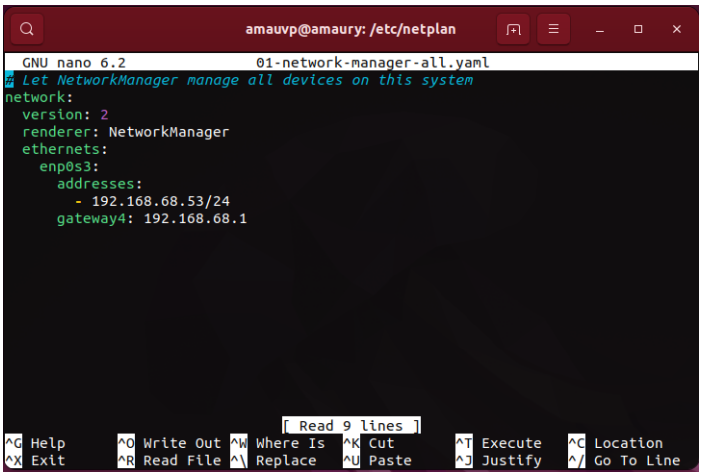


Figure 3.2: Netplan Configuration

The final step in configuring each virtual machine was to install Python and the Scapy library. It was also necessary to install the Flask and Flask-SocketIO libraries on the virtual machine playing the role of the victim. For the Python installation, version 3.9 of this programming language was chosen, as it is recognized as stable, particularly with the Scapy library. It was therefore necessary to follow six steps[3] in order to achieve optimum installation of Python 3.9.

1. Install all necessary dependencies.
2. Download the configuration file, unzip the archive and move to the unzipped folder.
3. Run the configuration script and compile the result.
4. Install Python 3 binaries.

```
sudo apt install build-essential zlib1g-dev libncurses5-dev libgdbm-dev
↪ libnss3-dev libssl-dev libreadline-dev libffi-dev libsqlite3-dev wget
↪ libbz2-dev -y
wget https://www.python.org/ftp/python/3.9.10/Python-3.9.10.tgz
tar -xvf Python-3.9.10.tgz
cd Python-3.9.10/
./configure --enable-optimizations
make
sudo make altinstall
```

And finally, in order to develop the script for sniffing DNS traffic and displaying this traffic in a web interface, it was necessary to install the Flask and Flask-SocketIO libraries on the virtual machine playing the role of the victim.

```
pip install scapy
pip install Flask
pip install flask-socketio
```

[3] https://vegastack.com/tutorials/how-to-install-python-3-9-on-ubuntu-22-04/

3.2 Set up the DNS services

The second crucial step in this project was to configure a DNS server as well as the clients of this DNS server. The choice made here was to do this with BIND9, which is very famous in the field. Installation was based on a tutorial[4].

The first step was to configure the hostname and Fully Qualified Domain Name (FQDN) of the DNS server.

```
sudo hostnamectl set-hostname amaury.thesis.io
echo '192.168.68.53    amaury.thesis.io ns1' | sudo tee -a /etc/hosts
```

The second step was to install the BIND9 services and configure their default options to work only with IPv4 addresses.

```
sudo apt install bind9 bind9utils bind9-doc dnsutils
echo 'OPTIONS="-u bind -4"' | sudo tee -a /etc/default/named
sudo systemctl restart named
sudo systemctl status named
```

The third step was to configure the DNS server itself. It's very important to note that the architecture here was intended to be simple, i.e. there would be just one Master DNS server with no Slave servers. This DNS server was also configured in such a way as to make it completely vulnerable to a DNS Amplification attack. The first step was to configure all DNS server options in the */etc/bind/named.conf.options* file. DNS server configuration is as follows:

- In order to make the DNS server vulnerable to the desired attack, it was necessary to make the server accept large UDP packets as input and output. This was made possible by the **edns-udp-size**, **max-udp-size** and **nocookie-udp-size** options. These options control the size of packets received and set the maximum size of UDP responses sent, respectively.
- Recursion was enabled by setting the **recursion** option to *yes*. It was also specified that any IP address could accept recursive DNS queries, thanks to the **allow-recursion** and **allow-recursion-on** options.
- It has been specified that any machine can send DNS requests to the server using the **allow-query** option.
- It was also necessary to specify the IPv4 address on which the server would listen for DNS queries, using the **listen-on** option.
- Zone transfer has been disabled as no slave server is in place. This was done using the **allow-transfer** option.

[4] https://www.howtoforge.com/how-to-setup-dns-server-with-bind-on-ubuntu-22-04/

- Google's DNS servers 8.8.8.8 and 8.8.4.4 have been defined as **forwarders** using the option with the same name. A forwarder is a server that resolves a domain name if the local DNS server is unable to do so.
- Since the first phase of this lab is to observe the DNS Amplification attack without deploying the DNSSEC protocol, DNSSEC validation has been disabled using the **dnssec-validation** option.

```
 GNU nano 6.2
options {
        directory "/var/cache/bind";

        edns-udp-size 8192;
        nocookie-udp-size 8192;
        recursion yes;
        allow-recursion { any; };
        allow-recursion-on { any; };
        allow-query { any; };
        allow-query-cache { any; };
        listen-on { 192.168.68.53; };
        allow-transfer { none; };
        max-udp-size 8192;
        dnssec-validation no;

        forwarders {
                8.8.8.8;
                8.8.4.4;
        };
        forward only;

        //listen-on-v6 { any; };
};
```

Figure 3.3: Configuration of /etc/bind/named.conf.options

After configuring the DNS server options, it was necessary to configure the domain name zones. These 2 zones are called the Domain Transfer Zone and the Reverse Lookup Zone. The DNS server configuration simply involved specifying the location of the zone files, i.e. the file containing all the zone's Resource Records. The subtlety here lay in the creation of custom zone files with the aim of reproducing results observed in the scientific literature. The zone file of most interest to us was the Domain Transfer Zone.

Side note

The contents of the zone files can be found in the appendix.

```
zone "amaury.thesis.io" {
        type master;
        file "/etc/bind/zones/db.amaury.thesis.io";
};

zone "68.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/zones/db.192.168.68";
};
```

Figure 3.4: Configuration of /etc/bind/named.conf.local

And finally, the last step in this configuration was to configure the DNS resolver of the attacking and victim virtual machines. This configuration consisted in ensuring that the DNS server previously configured was used as a priority when DNS requests were made by the attacking and victim machines. A Google DNS server was configured as the second DNS server to be contacted if DNS resolution could not be achieved via the laboratory's DNS server.

```
sudo unlink /etc/resolv.conf
sudo nano /etc/resolv.conf #See the configuration in the following figure
sudo apt install dnsutils bind9-utils
```

```
nameserver 192.168.68.53
nameserver 8.8.8.8
search amaury.thesis.io
```

Figure 3.5: Configuration of /etc/resolv.conf

Chapter 4
Implementation & Results

4.1 DNS Query Script for Amplification Rate Calculation

Before going into detail about the scripts, their purpose and results, it's important to note that they have been implemented in Python. The choice of this programming language was simply made out of familiarity and deeper knowledge of this programming language. What's more, this language has a highly reputed library for manipulating and sending network packets. This is the Scapy library. Scapy can be used to build network packets of any type, such as ARP, DNS, TCP, etc. This library can also be used to send these previously constructed packets. Finally, this library can be used to sniff the network to collect the various packets passing through it, which can then be manipulated and analyzed. The strength of this library will become clearer as we go on to describe the various scripts.

Side note

The source code for the entire project can be consulted on the GitHub repository[a] for this thesis.

[a] https://github.com/Amauvp/dns-amplification.git

The first step in the implementation phase was to create a script to send all the main types of DNS requests. The aim of this script is to be able to study the amplification rates on the vulnerable DNS server. The purpose of this study of amplification rates is to determine the type(s) of queries that will be particularly interesting and impactful for the attacker. In other words, the aim is to select the types of requests that will cause the most damage on the victim's side, i.e. the types of requests that will most quickly overload the victim's bandwidth. This script was initially designed to send DNS queries without including the DNSSEC security extension. It was essential that this script respect the basics of a DNS Amplification attack, i.e. usurpation of the victim's IP address, sending DNS queries to the vulnerable DNS server and the target domain name.

```
import argparse
from scapy.all import *
import time

def sendQueries(dnsSource, dnsDestination, queryName, duration):
    """
    Send DNS queries to a DNS server
    :param dnsSource: Source IP
    :param dnsDestination: Destination IP
    :param queryName: Query name
    :param duration: Duration in seconds
    """
    queryTypes = ["ALL", "A", "AAAA", "CNAME", "MX", "NS", "SOA", "TXT"]
```

Listing 1: sendQueries function specification and DNS types declaration

This script will therefore have as parameters the victim's IP address (source IP address), the IP address of the vulnerable DNS server (destination IP address), the target domain name and finally the time during which the queries will be sent, i.e. the measurement time. In the implementation of this script, before building the DNS packets and sending them, the different types of DNS queries of interest to us in the measurements were declared. These are as follows:

- **A** type is used to obtain IPv4 addresses relating to the requested domain name. The associated DNS response will therefore contain all the IPv4 addresses associated with the domain name.

- **AAAA** type is used to obtain IPv6 addresses relating to the requested domain name. The associated DNS response will therefore contain all the IPv6 addresses associated with the domain name.

- The **CNAME** type is used to obtain the canonical name (also known as the alias) of the requested domain. The associated DNS response will therefore contain a domain name. For example, the DNS request for the domain name *www.amaury.thesis.io* will return a **CNAME record** containing the domain name *amaury.thesis.io*.

- The **MX** type is used to obtain DNS records for mail servers in the requested domain. DNS responses will contain a list of mail servers with their priorities associated with their names.

- The purpose of the **NS** type is to find authoritative name servers for the requested domain. DNS responses will therefore contain a list of authoritative nameservers for the requested domain. For example, if the DNS request is for the domain name *amaury.thesis.io*, the associated response might contain the authoritative DNS server name *ns1.amaury.thesis.io*.

- **SOA** type provides Start of Authority information for the requested domain. This information is fundamental to the efficient management of DNS zones, as it provides precise details of the authority, synchronization parameters and critical administrative information required for DNS services to function correctly. The DNS response associated with this type will therefore contain all related information, such as the name of the primary server, the Serial Number of the zone, the Refresh Interval, the Retry Interval, the Expire Interval, the Minimum TTL, and so on.

- **TXT** type is used to retrieve the textual information associated with the requested domain name. The associated DNS response will therefore contain all textual records for the requested domain name. This textual data can be of any type. It can include domain-specific data, service definitions, arbitrary texts, etc.

Side note

It's important to note that when the DNSSEC extension is used, one or more RRSIG records will be added to each DNS response.

- **ANY** meta type is used to obtain all available information concerning the domain name specified in the DNS query. The response to this query will contain all available DNS records for the specified domain. For example, the response will contain all records of type A, AAAA, MX, NS, SOA, CNAME, TXT. If the DNSSEC security extension is used, DNS responses will also contain all records related to this extension, i.e. DNSKEY, RRSIG, DS, etc. records.

Definition

A DNS meta type is a special query or command that is not itself a specific record type, but is instead used to obtain global or complete information on the DNS records available for the given domain. The ANY type is an example of a DNS meta type, as it can retrieve all the DNS record types associated with a domain name in a single query.

Then, the various DNS packets are built up in a loop until the measurement time timer expires. The IP layer will contain the victim's spoofed IP address as the source IP address and the target DNS server's IP address as the destination IP address. The UDP layer will contain a random port number as the source port and port number 53 as the destination port. And finally, the DNS layer will contain an incremental number as the identifier or packet number, the rd (Recursion Desired) flag set to 1 to specify that DNS recursion is enabled, the target domain name and the type of queries. This DNS layer also contains a series of flags:

- The **QR** (Query/Response) flag, set to 0, indicates that the packet is a DNS request.
- The **CD** (Checking Disabled) flag set to 1 indicates that the resolver requires the DNS server to disable authenticity checking for the DNS request.
- The **AR** (Additional Records) flag set to DNSRROPT(rclass=8192) is intended to include DNS options specific to the DNS query. The value mentioned here and thus implemented in the script is intended to indicate that responses can be up to 8192 bytes in size. The aim is to increase the size of DNS responses as much as possible without truncating them.

```
packetNumber = 0
endTime = time.time() + duration

while time.time() < endTime:
    for queryType in queryTypes:
        packetNumber += 1
        dnsQuery = IP(src=dnsSource, dst=dnsDestination) / UDP(sport=RandShort(),
        ↪  dport=53) / DNS(id=packetNumber, rd=1, ad=0, cd=1,
        ↪  qd=DNSQR(qname=queryName, qtype=queryType), ar=DNSRROPT(rclass=8192))
```

Listing 2: Crafting DNS queries

Once the packet has been built, it is simply sent using Scapy's *send()* function. To launch this script, simply open a command terminal on the attacker's side, move to the folder containing the entire project and enter the following command:

```
sudo python3.9 attack/sendQuery.py -s <victimAddress> -d 192.168.68.53 -q
↪  amaury.thesis.io -t 60
```

It's important to note that this script can already constitute a DNS Amplification attack. Indeed, this script respects all the codes of such an attack and will therefore amplify the DNS responses returned. However, an enhancement will be made to the attack script itself to send the most impactful DNS query type(s) through several processes in order to maximize the number of DNS queries sent and thus increase the impact of the attack.

4.2 DNS Server Amplification Rate Calculation Script

The purpose of this script is to calculate amplification factors at DNS server level. In the context of a DNS Amplification attack, the amplification factor to be calculated will be the Bandwidth Amplification Factor, as this is the most relevant amplification factor for this attack. To achieve this, the script comprises 4 functions:

1. ***packetHandler***: this function filters all sniffed packets to retain only DNS packets. More specifically, this function will only retain DNS packets (requests and responses) concerning the domain name *amaury.thesis.io*, as well as the IP addresses of the victim and the DNS server used for the attack. This script will also store in two global dictionaries (one for queries and one for responses) the size and type of DNS queries and responses that may have been retained. The sizes and types of DNS packets will then be used to calculate amplification factors.

```
if IP in packet and UDP in packet and DNS in packet:
    if packet[IP].src == srcIP and packet[IP].dst == dstIP and
    ↪  packet[DNS].qr == 0 and str(packet[DNS].qd.qname.decode()) ==
    ↪  qname + '.':
        queryInfo = [packet[DNSQR].qtype, len(packet[DNS])]
        queriesInfo[packet[DNS].id] = queryInfo

    elif packet[IP].src == dstIP and packet[IP].dst == srcIP and
    ↪  packet[DNS].qr == 1 and str(packet[DNS].qd.qname.decode()) ==
    ↪  qname + '.':
        responseInfo = [packet[DNS].qd.qtype, len(packet[DNS])]
        responsesInfo[packet[DNS].id] = responseInfo
```

Listing 3: The packetHandler function

2. ***sniffPackets***: as its name suggests, this function sniffs network traffic passing through the machine's network interface. To do this, this function calls Scapy's *sniff()*[1] function, directly applying the *packetHandler* function to each packet captured.

```
sniff(filter=f"udp and port 53", timeout=25, prn=lambda x:
↪  packetHandler(x, srcIP, dstIP, qname))
```

Listing 4: The sniffPackets function

[1] https://scapy.readthedocs.io/en/latest/api/scapy.supersocket.html

3. ***calculateAmplificationFactors***: this function calculates the amplification factor for each request/response pair and stores the result in a data structure for each pair. The calculation is based directly on the Bandwidth Amplification Factor formula described in the state-of-the-art section previously described (See section 2.4). The amplification factor is therefore equal to the ratio between the size of the DNS response and the size of the corresponding DNS request.

```
results = {'255': [], '1': [], '28': [], '5': [], '15': [], '2': [],
↪  '6': [], '16': []}
for queryId in queriesInfo:
    for responseId in responsesInfo:
        if queryId == responseId:
            amplificationFactor = responsesInfo[responseId][1] /
            ↪  queriesInfo[queryId][1]
            results[str(queriesInfo[queryId][0])].append(amplificationF
            ↪  actor)

return results
```

Listing 5: The calculateAmplificationFactors function

4. ***meanAmplificationFactor***: as its name suggests, this function calculates the average amplification factor for each query type. To do this, this script will, for each query type, calculate the ratio between the sum of all amplification factors previously calculated for the query type concerned and the number of amplification factors contained in the data structure, or in other words, the number of query/response pairs observed.

```
meanFactors = {}
for queryType in allFactors:
    meanFactors[queryType] = sum(allFactors[queryType]) /
    ↪  len(allFactors[queryType])

return meanFactors
```

Listing 6: The meanAmplificationFactor function

To run this script, simply open a command terminal on the laboratory's DNS server, navigate to the folder containing the entire project and execute the following command:

```
1 sudo python3 server/calculateAF.py -s <victimAddress> -d 192.168.68.53 -q
  ↪  amaury.thesis.io
```

Once the script has been executed, the results will be displayed in the terminal and written to JSON files to keep track of them. These results include: the number of DNS requests, the number of DNS responses, the distribution of the different types of DNS requests and the amplification factors for each type of DNS request.

4.3 DNS Traffic Sniffing Script: Web Interface for Real-Time Monitoring and Machine Performance Metrics

The purpose of this script is to display DNS traffic passing through the victim's network interface. The web interface used to display this traffic is based on what can be done in

the Wireshark interface, to give a fairly clear and quick overview of DNS traffic. This web interface also displays the CPU usage percentage, RAM usage percentage and bandwidth consumption of the machine. The purpose of displaying both DNS traffic and machine performance is to observe the impact of the DNS Amplification attack directly on the victim machine.

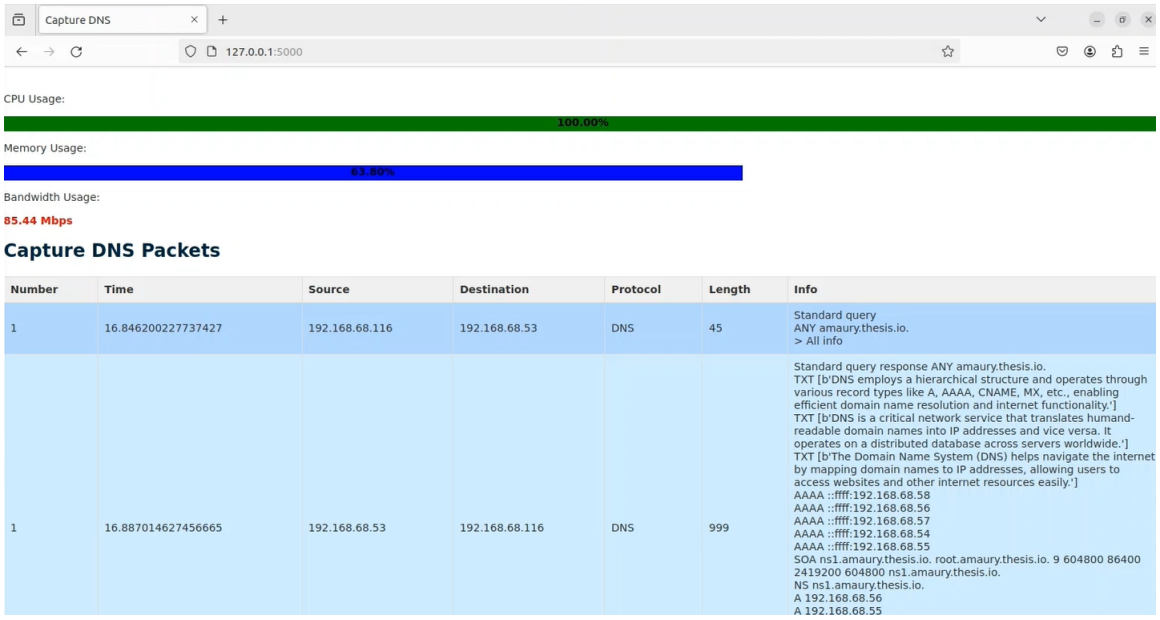


Figure 4.1: Web interface appearance

To implement all these functionalities, several libraries were used. Firstly, the *Flask*[2] library was used to implement the web interface. The choice of this library was motivated by the fact that it is extremely easy to use, lightweight and offers good performance for small to medium-sized applications. Secondly, the *Flask-SocketIO*[3] library was used to send data relating to DNS packets and machine performance in real time to the web interface. The choice of this library was motivated by the fact that Flask-SocketIO provides Flask applications with a low-latency and bi-directional communication channel. Moreover, the client-side can use the library to connect to a socket of its choice, so Flask-SocketIO is not restrictive for the client-side. And finally, the last crucial library in this implementation is Python's native *threading*[4] library. This allows network traffic to be sniffed, machine performance to be calculated and the Flask application to be run simultaneously in real time. On the server side, the implementation includes 2 major functions:

1. **packetHandler**: the purpose of this function is to be called by Scapy's *sniff()* function in order to filter DNS traffic so as to retain only DNS packets concerning the *amaury.thesis.io* domain. This function will then retrieve all relevant data from these DNS packets, such as the packet number, the time at which the packet was captured, the source IP address of the packet, the destination IP address of the packet, the packet size and the type of DNS packet (ANY, A, AAAA, etc.). All the data extracted from the packets is stored in a dictionary, which is then sent through the socket to be processed on the client side for display. When the DNS packet is a

[2]https://flask.palletsprojects.com/en/3.0.x/
[3]https://flask-socketio.readthedocs.io/en/latest/
[4]https://docs.python.org/3/library/threading.html#module-threading

DNS query, a character string is constructed to specify that the packet is a standard query, specifying the type of query and the domain name requested. When the DNS packet is a response, this character string will specify that the packet is a response, again specifying the type of response and the domain name associated with the response. However, this character string will also contain all the information relating to all the DNS records in the response, so that each DNS record can be displayed on the web interface. It's important to note that additional DNS information and records have not been processed here, to avoid overloading the web interface display, which will already be quite large by displaying all non-additional DNS records. This character string will in turn be stored in the dictionary containing the other data. Finally, this function will also store in the dictionary a summary of the DNS packet, which can be displayed on request. And finally, as mentioned above, the dictionary containing all the DNS packet data is sent across the socket to be processed and displayed by the client-side JavaScript code.

2. **getPerformances**: as its name suggests, this function is used to obtain machine performance data. More specifically, this function will store in a dictionary the percentage of CPU utilization, the percentage of RAM utilization and the bandwidth utilization. The data will then be sent through the socket to be displayed on the client side by the JavaScript code.

```
while True:
    cpuPercent = psutil.cpu_percent()
    memoryPercent = psutil.virtual_memory().percent
    bandwidth = (psutil.net_io_counters().bytes_sent +
    ↪  psutil.net_io_counters().bytes_recv) * 8/1000000
    performanceData = {'CPU': cpuPercent, 'memory': memoryPercent,
    ↪  'bandwidth': bandwidth}
    socketio.emit('performanceData', performanceData)
    time.sleep(2)
```

Listing 7: The getPerformances function

To launch this script, simply open a command terminal on the victim's side, move to the folder containing the entire project and enter the following command:

```
1 sudo python3 victim/viewTraffic.py
```

4.4 Analyzing Results Without DNSSEC

Before detailing the various amplification factors observed during this measurement phase, it is interesting and important to detail several points to be taken into consideration in order to best interpret these results. First of all, the BAF calculation depends directly on the size of DNS queries and the size of DNS responses. Query size depends on a number of factors, including the domain name requested. As shown in the article by R. van Rijswijk-Deij and al [31], very small domain names will have a smaller query size than very large domain names. This article highlights this through an example, mentioning that a one-character-long domain name will have a query size of 34 bytes

versus 96 bytes for a 63-character-long domain name. The article of Anagnostopoulos and al [1] supports this finding, arguing that TLDs are particularly interesting for this type of attack, given their very short domain names and the fact that the list of different TLDs is publicly available. For the purposes of this thesis, the domain name *amaury.thesis.io* is 16 characters long, which will undoubtedly have an impact on the size of queries and therefore on the amplification factors observed. More precisely, the query size for this laboratory's domain name is 45 bytes. Response sizes will depend on the DNS records available in the zone. Indeed, some zones will have few DNS records and therefore a relatively small response size. On the other hand, other zones will contain a large number of records, resulting in a large DNS response size. Generally speaking, an attacker will create a very large DNS zone, thanks in particular to TXT records, in order to maximize the size of the zone and therefore the size of DNS responses. This was described in detail in section 2.5 and in particular in the article by Anagnostopoulos and al [2].

For the purposes of this thesis, the zone was arbitrarily created with the aim of creating a relatively large zone in order to attempt to reproduce the results that may have been observed in the article by S. Ismail [23].

It is therefore very important to note that the results observed in this thesis will not correspond exactly to everything that has been observed in the scientific literature, as this depends directly on the size of the queries, the domains queried, the size of the responses and potentially on the different measurement environments. Furthermore, few scientific articles have measured these amplification factors for each type of DNS query, detailing the measurement environment, the measurement scripts and the different DNS zones studied. This is the main contribution of this thesis.

DNS query types	Length of DNS packets	
	Length of DNS query	Length of DNS response
A	45	109
AAAA	45	185
CNAME	45	90
MX	45	158
NS	45	79
SOA	45	90
TXT	45	607
ANY	45	999

Table 4.1: DNS query types and the length of DNS packets

The Table 4.1 details the DNS query and response sizes for the main query types detailed above. As mentioned above, the size of DNS queries is constant at 45 bytes. Response sizes, on the other hand, vary considerably, as expected. In this laboratory, response sizes range from 79 bytes to 999 bytes. As observed in the scientific literature and as

expected from the purpose/content of each type of DNS query, the ANY and TXT types have the largest response sizes. There are 2 reasons for this. The first reason is that, for the purposes of this thesis, the size of each TXT record has been maximized in order to obtain the largest possible response size for this type of DNS query. However, several TXT records could have been added in order to obtain the largest possible response size for this type of query, as detailed in the article by Anagnostopoulos and al [2]. The few TXT records therefore serve more as a proof of concept to demonstrate the power of this type of DNS record. The second reason explains why ANY queries have the highest amplification factor and the highest response size. This is because, as explained above, the aim of this type of query is to obtain all the information or DNS records available for the domain name requested. In the context of this thesis, the response size for a TXT query is already large. This size, combined with the size of all the other records, will make the ANY response as large as possible.

DNS query types	Amplification factors
A	2.4
AAAA	4.1
CNAME	2.0
MX	3.5
NS	1.8
SOA	2.0
TXT	13.5
ANY	22.2

Table 4.2: DNS query types and their amplification factors

With these DNS query and response sizes, it is now possible to detail the amplification factors for the different types of DNS queries. The amplification factors for this laboratory range from 1.8 to 22.2 when the DNSSEC extension is not used. Once again, it's interesting to note that ANY and TXT query types have the highest amplification factors. This observation is a direct consequence of the observation concerning the size of responses to these 2 DNS query types. It is also interesting to observe that the amplification factor corresponding to the ANY query type approaches the lower bound mentioned in the article of Ismail and al [23] and in the article of Rossow and Görtz [32]. However, the latter does not correspond exactly to this lower bound. In order to reach this lower bound, it would have been possible to add a few DNS records such as TXT records to increase the size of ANY responses and thus the corresponding amplification factor. It is also possible to compare the amplification factors of this laboratory with those observed in the article of R. van Rijswijk-Deij and al [31]. First of all, it is interesting to observe that the amplification factors in this article vary greatly from one field to another. This can be explained by the fact that query and response sizes are highly dependent on the domains queried and the content of the various DNS zones. It is rather difficult to interpret the results presented in this article, given the wide variation in amplification factors in the various graphs. However, by trying to pick out the peaks in these graphs to extract the most frequent amplification factors, it is possible to extract a kind of norm for the different types of DNS queries. It

is this standard that will be used for comparison, and the various peaks will represent the results column in Table 4.3. Secondly, it's interesting to note that some of the results of this thesis may correspond to the peaks observed in the graphs of this article. Indeed, similar results can be observed for query types A, AAAA and MX. On the other hand, some results diverge sharply, such as those for NS, TXT and ANY query types. These observations once again highlight the importance of detailing the size of DNS queries and responses, the configuration of the environment, the content of DNS zones and the various measurement scripts.

DNS query types	Amplification factors	
	Laboratory results	Article results
A	2.4	2-4
AAAA	4.1	3-4
CNAME	2.0	/
MX	3.5	2-5
NS	1.8	3-4
SOA	2.0	/
TXT	13.5	3
ANY	22.2	6

Table 4.3: DNS query types and their amplification factors from laboratory and scientific literature results

4.5 DNS Amplification Attack Script

Based on the results detailed previously, an attack script has been implemented. The logic of this script is exactly the same as that of the first script (See section 4.1). However, this script will only send ANY DNS requests, as this type of request has the highest amplification factor. This script will build ANY DNS requests and send them through the number of processes specified in the parameters. The greater the number of processes specified in the parameters, the more the victim will be saturated with DNS requests. However, it's important to note that, since the attacker here is an isolated attacker and doesn't use a botnet, he will very quickly consume the entire resources of his machine if the number of processes he requests is too high. On the other hand, this script is advantageous when the attacker is using a botnet, as the load will be distributed across the entire botnet. Indeed, the resources used will be those of the bots and not those of the attacker, which is particularly interesting in the scenario of such an attack.

This script could also be improved by alternating the sending of ANY and TXT DNS requests. However, the amplification factor of TXT requests is highly dependent on the DNS zone configuration. If the attacker were to configure his DNS zone for TXT queries in the way it has been configured in this laboratory, or even more powerfully, alternating the sending of these types of queries would be particularly relevant. Indeed, this would enable the attacker to try avoiding defense mechanisms such as BIND9's Response Rate

Limiting, which will be detailed in the next chapter of this thesis.
To launch this script, simply open a command terminal on the attacker's side, move to the folder containing the entire project and enter the following command:

```
sudo python3.9 attack/dnsAttack.py -s <victimAddress> -d 192.168.68.53 -q
↪  amaury.thesis.io -t 60 -n <Number of process>
```

4.6 DNSSEC Deployment

In order to deploy the DNSSEC extension, BIND9 provides documentation[5] on how to do this. Their basic tutorial allows you to deploy DNSSEC almost automatically. However, for the purposes of this thesis, this tutorial didn't work, so it was necessary to deploy the DNSSEC extension manually, using a mix of information from the BIND9 documentation and a DigitalOcean[6] tutorial.

The first step in deploying DNSSEC is to generate the various key pairs, i.e. the pairs (public ZSK, private ZSK) and (public KSK, private KSK). These are the keys that will be used to sign the zone and create DNSSEC-related DNS records. To generate these keys, simply navigate to the directory containing the BIND files and run the following commands:

```
dnssec-keygen -a ECDSAP256SHA256 amaury.thesis.io
dnssec-keygen -a ECDSAP256SHA256 -f KSK amaury.thesis.io
```

Here the choice of digital signature algorithm was ECDSA[7] Curve P-256 with SHA-256. The choice of this algorithm was simply made because it was the algorithm mentioned for manual key generation in the BIND9 documentation. However, a Cloudflare article[8] has highlighted a number of interesting points in the adoption of this signature algorithm, which may help to justify future results in terms of amplification factors. The most interesting point in the context of an Amplification attack, as mentioned by Cloudflare, is that the RSA key size required for a certain level of security is considerably higher than the ECDSA key size. Indeed, to achieve 128-bit security, the RSA key size needs to be 12 times larger than the ECDSA key size. However, Cloudflare mentions that this 12-fold factor is not the norm. Despite the fact that this 12-fold factor is not the norm, it's worth noting that using RSA instead of ECDSA potentially returns larger answer sizes, thereby increasing amplification factors. Last but not least, Cloudflare also mentions a major drawback of ECDSA. According to Cloudflare, this signature algorithm offers poorer performance than RSA when it comes to signature validation. According to the article, ECDSA is 6.6 times slower than RSA when it comes to signature validation.

[5] https://bind9.readthedocs.io/en/v9.18.14/dnssec-guide.html
[6] https://www.digitalocean.com/community/tutorials/how-to-setup-dnssec-on-an-authoritative-bind-dns-server-2
[7] https://cryptobook.nakov.com/digital-signatures/ecdsa-sign-verify-messages
[8] https://www.cloudflare.com/dns/dnssec/ecdsa-and-dnssec/

After generating the key pairs, it was necessary to include the public ZSK and KSK in the zone file, so that these keys could be taken into account when signing the zone. To do this, simply modify the zone file and include the following 2 lines at the end of this zone file:

```
$INCLUDE Kamaury.thesis.io.zsk.key #Replace zsk with the digits of the generated
↪ key
$INCLUDE Kamaury.thesis.io.ksk.key #Replace ksk with the digits of the generated
↪ key
```

Once this step has been completed, the zone can be signed using the following command:

```
dnssec-signzone -A -3 $(head -c 1000 /dev/random | sha1sum | cut -b 1-16) -N
↪ INCREMENT -o amaury.thesis.io -t db.amaury.thesis.io
```

Once the zone had been signed, it was necessary to modify the *named.conf.local* file to indicate that the reference file for the zone was now the signed zone file.

```
zone "amaury.thesis.io" {
        type master;
        file "/etc/bind/zones/db.amaury.thesis.io.signed";
};

zone "68.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/zones/db.192.168.68";
};
```

Figure 4.2: Configuration of /etc/bind/named.conf.local with DNSSEC

It was also necessary to modify the *named.conf.options* file to enable DNSSEC validation.

```
options {
        directory "/var/cache/bind";

        edns-udp-size 8192;
        nocookie-udp-size 8192;
        recursion yes;
        allow-recursion { any; };
        allow-recursion-on { any; };
        allow-query { any; };
        allow-query-cache { any; };
        listen-on { 192.168.68.53; };
        allow-transfer { none; };
        max-udp-size 8192;
        dnssec-validation yes;


        forwarders {
                8.8.8.8;
                8.8.4.4;
        };
        forward only;

        //listen-on-v6 { any; };
};
```

Figure 4.3: Configuration of /etc/bind/named.conf.options with DNSSEC

The last step was to restart the BIND9 services with the following command:

```
service bind9 reload
```

It's important to note that DS records were not configured here, as this would require the domain name to be purchased/rented. Furthermore, as the zone is a self-signed zone

and the environment is a laboratory environment, there was no need to check the chain of trust for DNSSEC deployment. However, this will have a direct impact on the size of DNS responses, as DS records will not be present in these responses.

4.7 Adapting Scripts for DNSSEC Protocol

In order to adapt the various scripts to the DNSSEC extension, it was necessary to modify them. The main modifications involved processing the various DNS records relating to the DNSSEC extension, i.e. the RRSIG, DS, DNSKEY, NSEC and NSEC3 records.

4.7.1 Adapting The DNS Query Script for Amplification Rate Calculation

The first script to be modified was the script for sending the different types of DNS requests, in order to measure the amplification factors of these types of requests. First of all, a parameter was added to the script call. The purpose of this parameter is to specify whether the use of the DNSSEC extension has been requested by the user. When this is the case, two modifications have been made to this script:

1. The first change is to add the relevant DNS query types to the tables of query types to be sent when the use of the DNSSEC extension is requested. For the purposes of this thesis, the types added are RRSIG, DNSKEY and NSEC. Indeed, when we look at the signed zone, it quickly becomes apparent that we can't observe the NSEC3 and DS types. It would therefore be irrelevant to calculate amplification factors for these query types, which is why they have not been added to the table of DNS query types.

2. The second change is to set the CD (Checking Disabled) flag to 0 when the use of the DNSSEC extension is requested by the user. This ensures that the resolver checks DNSSEC signatures to ensure the integrity and authenticity of DNS records.

```
queryTypes = ["ALL", "A", "AAAA", "CNAME", "MX", "NS", "SOA", "TXT"]

if use_dnssec:
    dnssecRecordTypes = ["RRSIG", "DNSKEY", "NSEC"]
    queryTypes.extend(dnssecRecordTypes)
```

Listing 8: Adding DNSSEC-related DNS records

```
if use_dnssec:
    dnsQuery[DNS].cd = 0
    dnsQuery[DNS].qr = 0
    dnsQuery[DNS].ar = DNSRROPT(rclass=8192)
else:
    dnsQuery[DNS].cd = 1
    dnsQuery[DNS].qr = 0
    dnsQuery[DNS].ar = DNSRROPT(rclass=8192)
```

Listing 9: Adapting the various flags

4.7.2 Adapting The DNS Server Amplification Rate Calculation Script

The modification made to this script lies, firstly, in the addition of the same parameter as the previous script, to ask the user if they want the DNSSEC extension to be used. Where required, DNSSEC-related DNS query types have simply been added to the dictionary storing the various amplification factors. Next, the number of DNSSEC-related query-response pairs sniffed by the script was displayed when this extension is used.

4.7.3 Adapting The DNS Traffic Sniffing Script

The most important script to modify was the one used to sniff DNS traffic on the victim machine. Indeed, the complexity of the modifications here lay in correctly processing the various DNS records relating to the DNSSEC extension so that they could be correctly displayed in the web interface. The other difficulty in this modification phase was to factorize all the code in this script, which was beginning to be totally incomprehensible due to the repetition of the same portions of code and successive conditions. As regards the processing of DNS records relating to the DNSSEC extension, these were simply processed using the relevant attributes that Scapy was able to obtain. The logic here was to select only those attributes that could be observed in the result of the dig command. As regards code factorization, two functions have been created. The first function retrieves the DNS query type identified for a number and associates it with a textual query type to make it more comprehensible in the web interface. The purpose of the second function is to extract information from various DNS records that have a few special fields and therefore cannot be processed like other DNS records. These record types are SOA, MX, DS, RRSIG, NSEC, DNSKEY and NSEC3.

```
if record.type == 6:  # SOA
        return f"{record.mname.decode()} {record.rname.decode()} {record.serial} {record.refresh}
        ↪  {record.retry} {record.expire} {record.minimum}"
    elif record.type == 15:  # MX
        return f"{record.ttl} {record.preference} {record.exchange.decode()}"
    elif record.type == 43:  # DS
        return f"{record.keytag} {record.algorithm} {record.digesttype} {record.digest.decode()}"
    elif record.type == 46:  # RRSIG
        return f"{record.typecovered} {record.algorithm} {record.labels} {record.originalttl}
        ↪  {record.expiration} {record.inception} {record.keytag} {record.signersname.decode()}
        ↪  {base64.b64encode(record.signature).decode('ascii')}"
    elif record.type == 47:  # NSEC
        return f"{record.nextname.decode()} {record.typebitmaps}"
    elif record.type == 48:  # DNSKEY
        return f"{record.flags} {record.protocol} {record.algorithm}
        ↪  {base64.b64encode(record.publickey).decode('ascii')}"
    elif record.type == 50:  # NSEC3
        return f"{record.hashalg} {record.flags} {record.iterations} {record.salt.decode()}
        ↪  {record.hashalg} {record.nexthashedownername.decode()} {record.typebitmaps}"

    return ''
```

Listing 10: Processing of specific records

4.7.4 Adapting DNS Server And Victim Configurations

In order to be able to sniff all DNS packets and avoid their fragmentation, it was necessary to make a slight modification to the DNS server configuration and to the configuration of the victim machine. The use of the DNSSEC extension considerably increased the size

of the DNS responses to be sniffed. This increase in response size caused DNS responses to become fragmented. Before modifying the configuration of the DNS server and the victim machine, an attempt was made to manage the fragmented DNS packets within the code itself. However, this attempt failed due to the fact that handling fragmented packets with Scapy is quite complex. Indeed, some layers cannot be rebuilt or are very complex to rebuild. Following this failure, it was decided to modify the DNS server and victim configuration to avoid packet fragmentation, so that DNS responses could be sniffed in one piece. To achieve this, the MTU (Maximum Transmission Unit) was set at 2800, which defines the maximum size of data packets that can be sent over a network interface without needing to be fragmented. To achieve this configuration, we simply ran the following commands and set up the following configurations:

```
cd /etc/netplan
sudo nano 01-network-manager-all.yaml
```

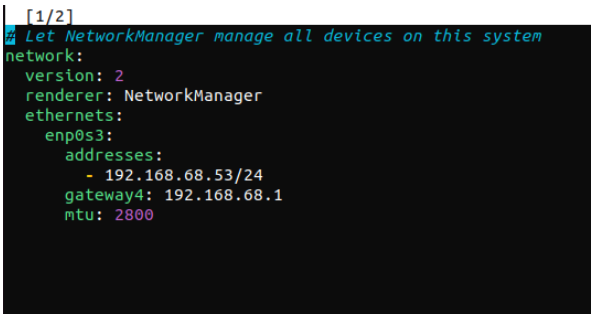


Figure 4.4: MTU configuration of the DNS server

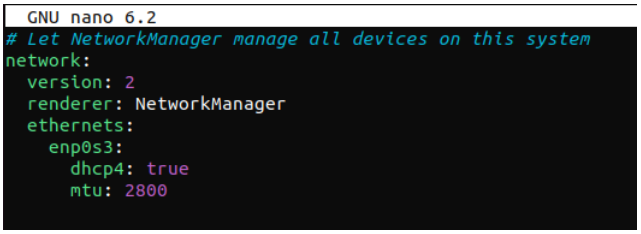


Figure 4.5: MTU configuration of the victim machine

The last step in this configuration is to apply the netplan configuration changes with the following command:

```
sudo netplan apply
```

4.7.5 Adapting The DNS Amplification Attack Script

As far as the modification of this script is concerned, the same parameter as in the previous scripts has been added to specify once again whether the user wishes to use the DNSSEC extension. Then, the only change made was to set the CD (Checking Disabled) flag to 0, once again requiring the DNS resolver to check DNS record signatures. However, another

modification could be made to this script. Depending on the amplification factors observed when using DNSSEC, other types of DNS requests could be sent in addition to ANY requests. This would have the advantage of evading defensive measures such as Response Rate Limiting.

4.8 Analyzing Results With DNSSEC

As regards this phase of measuring the amplification factors of different types of DNS queries using the DNSSEC extension, it is also important to take into account all the factors, previously detailed in section 4.4, that can influence these measurements. As a reminder, this laboratory's DNS zone was signed using the ECDSA Curve P-256 with SHA-256 digital signature algorithm. This choice of algorithm obviously has an influence on the results obtained in this laboratory (see Section 4.6).

DNS query types	Length of DNS packets	
	Length of DNS query	Length of DNS response
A	45	221
AAAA	45	297
CNAME	45	357
MX	45	606
NS	45	303
SOA	45	202
TXT	45	719
RRSGIG	45	1053
DNSKEY	45	429
NSEC	45	200
ANY	45	2658

Table 4.4: DNS query types and the length of DNS packets using DNSSEC

The table 4.4 shows how the use of the DNSSEC extension greatly increases response sizes for each of the DNS query types. For some query types, response sizes have literally exploded. These include CNAME, MX, NS and ANY. For CNAME, MX and NS, the size of DNS responses has increased by a factor of 4. For the ANY type, response size increased by a factor of 2.2, from 999 bytes without DNSSEC to 2658 bytes with DNSSEC. It's also interesting to note that response size for the TXT request type didn't increase significantly, rising from 607 bytes without DNSSEC to 719 bytes with DNSSEC.

Two types of DNS records become potentially interesting for a DNS Amplification attack. Indeed, the response size for the RRSIG and DNSKEY query types is relatively high, which will undoubtedly produce a potentially interesting amplification factor for this type of query.

DNS query types	Amplification factors
A	4.9
AAAA	6.6
CNAME	7.9
MX	13.5
NS	6.7
SOA	4.5
TXT	16
RRSIG	23.4
DNSKEY	9.5
NSEC	4.4
ANY	59.1

Table 4.5: DNS query types and their amplification factors using DNSSEC

As expected, the increase in response size for each type of DNS query has caused an increase in the various amplification factors. The amplification factors for this lab now range from 4.4 to 59.1 when the DNSSEC extension is used. It is now interesting to note that the DNS query types with the highest amplification factors are MX, TXT, RRSIG and ANY. The amplification factor for the ANY type approaches and even exceeds the upper bound mentioned in the paper of Ismail and al [23] and in the paper of Rossow and Görtz [32]. It is again interesting to compare the laboratory results of this thesis with the results drawn up in the paper of R. van Rijswijk-Deij and al [31]. Once again, the amplification factors in this article when the DNSSEC extension is used vary greatly from one domain to another. This is again due to the various factors that can influence the size of DNS responses, but also to the way in which DNS zones have been signed. It is rather difficult to interpret the results presented in this article, given the wide variation in amplification factors in the various graphs. However, by trying to pick out the peaks in these graphs to extract the most frequent amplification factors, it is possible to extract a kind of norm for the different types of DNS queries. It is this standard that will be used for comparison, and the various peaks will represent the results column in Table 4.6. In this article, the amplification factor for the ANY type is lower than the amplification factor observed in this thesis. For types A, AAAA , NS and TXT, the amplification factors observed in this thesis match some of the peaks observed in the article. And finally, this laboratory's amplification factors for MX, DNSKEY and NSEC types do not match what can be observed in this article. It is also important to note that this article mentions outliers for which amplification factors can exceed 100. The article of Anagnostopoulos and al [3] also mentions an amplification factor that can exceed 230 for ANY query types. However, it's important to note that these outliers are not the norm and, once again, amplification factors vary widely from one domain to another.

The most important information to be drawn from these results is that the ANY query type does indeed have the highest amplification factor. Other query types may also be of interest for a DNS Amplification attack, such as TXT, RRSIG and MX.

DNS query types	Amplification factors	
	Laboratory results	Article results
A	4.9	5, 10 and 15
AAAA	6.6	5 and 15-18
CNAME	7.9	/
MX	13.5	6-8
NS	6.7	6 and 9
SOA	4.5	/
TXT	16	6 and 16-18
RRSIG	23.4	/
DNSKEY	9.5	13-28
NSEC	4.4	15
ANY	59.1	40

Table 4.6: DNS query types and their amplification factors from laboratory and scientific literature results using DNSSEC

4.9 Signing The Zone With RSA

As the Cloudflare article[9] mentions that the use of the RSA digital signature algorithm could potentially produce interesting answer sizes, it was interesting to carry out a measurement phase by first signing the area of this laboratory with the RSA algorithm. The article by R. van Rijswijk-Deij [31] mentions that the most common key sizes are 1024-bits and 2048-bits long. In this thesis, only the 1024-bit key size will be tested for performance reasons. Indeed, in the laboratory environment, the 2048-bit key size could be much too greedy and would not bring any new results than the 1024-bit one. It's also important to note that there are 2 RSA signature algorithms: RSASHA256 and RSASHA512. For the purposes of this thesis, the RSASHA256 signature algorithm has been chosen to match the ECDSA Curve P-256 with SHA-256 algorithm initially selected.
To do this, it is necessary to generate the RSA key pairs using the following commands:

```
1 dnssec-keygen -a RSASHA256 -b 1024 amaury.thesis.io
2 dnssec-keygen -a RSASHA256 -f KSK -b 1024 amaury.thesis.io
```

After the key pair generation, the zone is signed exactly as explained in section 4.6.

4.10 Analyzing Results With DNSSEC and RSA Signed Zones

Before analyzing the various results obtained, it's important to note that in order to avoid packet fragmentation once again, the MTU had to be increased to 4000 using the following commands:

[9] https://www.cloudflare.com/dns/dnssec/ecdsa-and-dnssec/

```
cd /etc/netplan
sudo nano 01-network-manager-all.yaml
sudo netplan apply
```

It's interesting to note that, as the Cloudflare article mentions, the size of DNS responses has indeed exploded, which has also caused the amplification factors of the various DNS query types to explode. Indeed, amplification factors now range from 5.9 to 80.2. Comparing these results with those obtained previously with the ECDSA Curve P-256 with SHA-256 signature algorithm, it is interesting to note that the RSA algorithm does indeed offer higher amplification factors, as Table 4.7 shows. For example, the amplification factor for the ANY query type increases from 59.1 to 80.2. The amplification factor for the MX query type rises from 13.5 to 19.2. And finally, the amplification factor for the RRSIG query type rises from 23.4 to 36.2. An attacker would therefore be well advised to sign his zone with the RSA algorithm, or to target a DNS zone signed with this algorithm.

DNS query types	Amplification factor	
	ECDSA	RSA
A	4.9	6.3
AAAA	6.6	8.0
CNAME	7.9	11.8
MX	13.5	19.2
NS	6.7	9.6
SOA	6.7	5.9
TXT	16	17.4
RRSIG	23.4	36.2
DNSKEY	9.5	15.4
NSEC	4.4	11.8
ANY	59.1	80.2

Table 4.7: DNS query types and their amplification factors for ECDSA and RSA

Finally, it is important to note that amplification factors also depend on the type of digital signature algorithm used to sign the zone. The extremely high amplification factors reported in the scientific literature are therefore possible depending on the zone configuration, the environment configuration and the way the zone was signed.

Important remark

The remainder of the same will be processed with the signed zone using the ECDSA Curve P-256 with SHA-256 digital signature algorithm for performance reasons, to make it less cumbersome in the laboratory environment.

Chapter 5
Exploring Mitigation Methods for DNS Amplification Attacks

In this chapter, various mitigation methods against a DNS Amplification attack will be investigated and evaluated in order to observe what can be put in place by an individual to try and protect against this type of attack. The methods covered in this chapter are: the best practices for a DNS server configuration, the mechanism for minimizing response size and in particular the size of responses to ANY queries, the Response Rate Limiting mechanism, and the use of TCP protocol for DNS name resolution instead of UDP.

5.1 Best DNS Server Configuration Practices

As the laboratory's DNS server is an authoritative DNS server for its zone, an article from the Internet Systems Consortium (ISC) [20] was consulted to analyze the best practices that could be implemented to achieve the best possible configuration. The first best practice given by this article is to configure the DNS server on a dedicated machine, mainly to ensure that other services are not impacted in the event of an attack, or for various performance reasons. The second best practice is to separate authoritative DNS servers from recursive DNS servers. More concretely, this means deciding which DNS servers will play the role of authoritative DNS servers and which will play the role of recursive DNS servers. Authoritative servers will only return information relating to the zone for which they are authoritative. For the purposes of this thesis, recursion should be disabled on the DNS server, as it is authoritative for the zone it serves, the zone of this laboratory. And finally, the last major best practice to be implemented according to this article is to configure the Response Rate Limiting mechanism. The implementation of this mechanism will be detailed in another section of this chapter. Based on these best practices, the only thing to do on this laboratory's DNS server is to disable recursion and see if this changes anything in the results observed.

```
  GNU nano 6.2
options {
        directory "/var/cache/bind";

        edns-udp-size 8192;
        nocookie-udp-size 8192;
        recursion no;
        allow-recursion { none; };
        allow-recursion-on { none; };
        allow-query { any; };
        allow-query-cache { any; };
        listen-on { 192.168.68.53; };
        allow-transfer { none; };
        max-udp-size 8192;
        dnssec-validation yes;

        forwarders {
                8.8.8.8;
                8.8.4.4;
        };
        forward only;

        //listen-on-v6 { any; };
};
```

Figure 5.1: Configuration with best practices of /etc/bind/named.conf.options

Unfortunately, after running a test to see if this reduced the number of responses sent or the amplification factors observed, this test showed that following best configuration practices made no difference. This result was expected. Indeed, simply respecting best DNS server configuration practices does not protect against a DNS Amplification attack.

5.2 Minimizing Response Size

The first technique for minimizing the impact of a DNS Amplification attack is a piece of advice found in RFC8482 [19]. This article advises minimizing the size of ANY responses through 3 main behaviors:

1. The responder could choose a subset of DNS records to return when it receives an ANY request.

2. The responder could send back a HINFO record in order to send back a synthesized response. Cloudflare [26], for example, has implemented this mechanism by specifying in this record that according to RFC8482 the size of ANY responses is minimal. However, in discussions with the BIND9 community, a certain Mark Andrews from ISC highlighted what was really behind this RFC and why it had been written. In his words: "This RFC was written for database servers where getting an ANY response is actually difficult. Cloudflare was using a response pattern that most people thought wasn't really correct, but wasn't broken enough to say: Don't do this. If their customers were happy with that behavior, then okay. This RFC was written to allow them to keep doing what they were doing without having to fight the fact that they weren't RFC-compliant. It wasn't written to say that this is the way to respond to ANY". His words make it clear that it wouldn't really be right to implement the same mechanism that Cloudflare has deployed to minimize the size of ANY responses.

3. The resolver can try to return the DNS records they think are most suitable/probable for the requester.

To minimize the size of responses, BIND9 has implemented several mechanisms. Firstly, the *minimal-responses*[1] option controls whether the server can add additional records to the authority and additional data sections. Setting this option to *yes* has the effect of "adding records to the authority and additional data sections only when such records are required by the DNS protocol", according to the BIND documentation. Next, the *minimal-any*[2] option attempts to minimize the size of ANY responses by returning as few RRsets as possible. This option will therefore be set to *yes* in order to minimize the size of ANY responses.

[1] https://bind9.readthedocs.io/en/v9.18.25/reference.html#namedconf-statement-minimal-responses
[2] https://bind9.readthedocs.io/en/v9.18.25/reference.html#namedconf-statement-minimal-any

```
 GNU nano 6.2
options {
        directory "/var/cache/bind";

        minimal-responses yes;
        minimal-any yes;
        edns-udp-size 8192;
        nocookie-udp-size 8192;
        recursion no;
        allow-recursion { none; };
        allow-recursion-on { none; };
        allow-query { any; };
        allow-query-cache { any; };
        listen-on { 192.168.68.53; };
        allow-transfer { none; };
        max-udp-size 8192;
        dnssec-validation yes;

        forwarders {
                8.8.8.8;
                8.8.4.4;
        };
        forward only;

        //listen-on-v6 { any; };
};
```

Figure 5.2: Minimizing Response Size: Configuration of /etc/bind/named.conf.options

During the test phase of this mitigation method, several very interesting results emerged. Table 5.1 shows the new query and response sizes observed. It can be seen that some DNS response sizes have been drastically reduced. The most impressive reduction is in the size of ANY responses. These responses go from a size of 2658 bytes without the use of countermeasures to a size of 202 when the two options detailed above are used. RRSIG responses undergo the same significant reduction in size. Indeed, the size of this type of response drops from 1053 bytes without the use of this mitigation measure to 157 bytes when using this mitigation technique. And finally, the last type of response to undergo a reduction in size is the MX type. This type of response goes from a size of 606 bytes without the use of these two options to a size of 222 bytes. These three response types have therefore seen their size reduced by a factor of 13, 7 and 3 respectively. Unfortunately, the other DNS response types did not experience any reduction in size. For some response types, this is still not too problematic. On the other hand, it is still a problem for the TXT response type, which is still very large and now has the largest response size. However, by simply analyzing this reduction in size, it is possible to deduce that this mitigation measure has a positive impact on protection against this type of attack. In addition, it remains interesting to observe the impact of this technique on the various amplification factors.

DNS query types	DNS responses length	
	Before mitigation	After mitigation
A	221	221
AAAA	297	297
CNAME	357	357
MX	606	222
NS	303	303
SOA	202	202
TXT	719	719
RRSGIG	1053	157
DNSKEY	429	429
NSEC	200	200
ANY	2658	202

Table 5.1: DNS query types and the length of DNS responses before and after mitigation

The same observation can be made for the different amplification factors of the ANY, MX and RRSIG query types. In fact, the reduction in response size for these types leads to a proportional decrease in their amplification factor. The amplification factor for ANY queries drops from 59.1 to 4.5, corresponding to a reduction of a factor of 13, as previously observed. Next, the amplification factor for the RRSIG type drops from 23.4 to 3.5, which again corresponds to a reduction of a factor of 7. And finally, the amplification factor for the MX type suffers the same fate. In fact, this amplification factor fell by a factor of 3, from 13.5 to 4.9. However, the same observation as for the response sizes of other query types can be made for the amplification factors. Indeed, the amplification factors for the other query types remain unchanged, as do their sizes. This invariability therefore still poses a problem for some query types, such as TXT or DNSKEY queries, which retain the highest amplification factors. However, an analysis of the various observations on this mitigation method shows that it is indeed effective and delivers the expected results. It is important to note, however, that a single mitigation method will not be sufficient to effectively protect against this type of attack. It is therefore necessary to investigate the Response Rate Limiting mechanism.

DNS query types	Amplification factors	
	Before mitigation	After mitigation
A	4.9	4.9
AAAA	6.6	6.6
CNAME	7.9	7.9
MX	13.5	4.9
NS	6.7	6.7
SOA	4.5	4.5
TXT	16	16
RRSIG	23.4	3.5
DNSKEY	9.5	9.5
NSEC	4.4	4.4
ANY	59.1	4.5

Table 5.2: DNS query types and their amplification factors before and after mitigation

5.3 Response Rate Limiting Mechanism

Before going into detail about the Response Rate Limiting mechanism, it's important to note that all the information in this section is based on two ISC articles[3][4], as well as on the BIND9 reference manual also written by the ISC[5].

The Response Rate Limiting mechanism is a DNS protocol enhancement that serves as a mitigation technique against the DNS Amplification attack. It is very important to note that the deployment of this mechanism is only recommended for authoritative DNS servers. This mitigation measure could therefore make sense in the context of this thesis, given that the DNS server in this laboratory is authoritative for its zone. The principle

[3]What is the Response Rate Limiting Feature in BIND?
[4]Using the Response Rate Limiting Feature
[5]The BIND 9 Administrator Reference Manual

of this mechanism is to avoid excessive UDP responses through various mechanisms such as the number of responses per second or dropping responses to the same queries too frequently.

The response rate limiting mechanism is based on a system of "credits" or "tokens". For each unique combination of response and customer identity, a virtual "account" is assigned, accumulating a defined quantity of credits every second. A considered answer reduces this account by one credit. If the account is negative, responses are either abandoned or truncated. Responses are tracked within a moving time window, set at 15 seconds by default, but adjustable via the "window" option. If the number of credits allocated to a response category is set to 0, then these responses are not subject to rate limiting.

In the context of this thesis, only a few options will be investigated in order to configure this mechanism and observe interesting behaviors. These are the following options:

- **window**: this parameter specifies the time period during which responses are tracked. The default value is 15 seconds, but it can be set between 1 and 3600 seconds.
- **responses-per-second**: this option limits the number of non-empty responses for a valid domain name and a certain DNS record type. The default value is 0, i.e. no limit is imposed.
- **slip**: the purpose of this option is to limit the size of frequently observed responses or to drop responses that are too frequently observed. The default value of this option is 2. According to the documentation, this means that "every other UDP request without a valid server cookie to be answered with a small response". The value of this option can vary between 0 and 10. According to the documentation, a value set to 0 will result in "no small responses are sent due to rate limiting. Rather, all responses are dropped".
- **all-per-second**: this option limits UDP responses of any type. It is important to note that the value of this option must be at least 4 times greater than the value of the responses-per-second option. The value of this option is similar to the throughput limitation offered by firewalls, but is often lower.

For the purposes of this thesis, the *window* parameter will be set to its default value, and will not be modified thereafter, in order to monitor the other parameters much more precisely. Each option will then be investigated individually to observe its effects.

5.3.1 First Option: responses-per-second

Four configurations were tested in order to study the effects of this option on the number of responses sent by the DNS server in relation to the number of queries it receives. The results of the various tests can be seen in Table 5.5. It's interesting to note that for the first configuration, no difference was found. However, for the other configurations, a significant change can be observed. In fact, the number of responses sent equals approximately half the number of queries received by the DNS server.

responses-per-second configuration	Number of DNS queries	Number of DNS responses
5	2852	2852
4	2758	1504
3	2899	1497
2	3017	1521

Table 5.3: responses-per-second configurations and their corresponding DNS queries and responses

By also analyzing the amplification factors corresponding to each of these configurations, the same observation can be made (see Table 5.4). Indeed, for the first configuration, the amplification factors remain unchanged. By contrast, for the other configurations, amplification factors are drastically reduced. If, in addition to these highly constraining configurations, response size minimization is applied, these amplification factors could potentially be further reduced.

Amplification factors	responses-per-second configuration			
	5	4	3	2
A	4.9	1.7	1.2	1.1
AAAA	6.6	1.8	1.3	1.1
CNAME	7.9	2.0	1.4	1.1
MX	13.5	2.8	1.7	1.2
NS	6.7	1.9	1.3	1.1
SOA	4.5	1.5	1.2	1.0
TXT	16	3.4	1.9	1.2
RRSIG	23.4	4.8	2.3	1.3
DNSKEY	9.5	2.4	1.6	1.1
NSEC	4.4	1.6	1.25	1.0
ANY	59.1	11.9	4.4	2.3

Table 5.4: Amplification factors for different responses-per-second configurations

However, it is very important to note that such configurations can be extremely restrictive for legitimate customers. As discussed with Professor Debatty, a Machine Learning algorithm would be needed here to adapt such a configuration to the behavior of legitimate users in real time.

5.3.2 Second Option: slip

The slip option was then investigated to see if, on its own, it could mitigate this type of attack. Once again, four configurations were tested. The values for these configurations were 0, 1, 2 and 10 respectively. However, no consequences could be observed at the end of these tests. As a result, this option alone does not appear to influence the impact of the attack.

5.3.3 Third Option: all-per-second

With regard to the all-per-second option, three configurations were tested in order to observe the impact of this option on the attack's impact. No effect was observed on amplification factors. On the other hand, the effect of this option was more on the number of responses sent by the DNS server, as shown by the results in Table 5.5. Indeed, as the value of this option increases, the number of responses sent by the server decreases proportionally. For example, for a value of 20, the number of responses decreased by a factor of 108. For a value of 50, the reduction was very slight. And finally, for value 100, no decrease could be observed.

all-per-second configuration	Number of DNS queries	Number of DNS responses
20	2909	27
50	2680	2497
100	2756	2756

Table 5.5: all-per-second configurations and their corresponding DNS queries and responses

5.3.4 General Conclusion

It's very important to note that the interpretation of the results could be biased due to the implementation of the sniffing script running on the DNS server. Indeed, traffic sniffing may have an indeterminate behavior for each of the options modified here. For example, some packets might not pass through certain conditions of this script and therefore not be taken into account in the observed results.

However, the overall conclusion for this mitigation solution remains unchanged. Indeed, such a rate-limiting mechanism should be studied for implementation through a Machine Learning mechanism, so that the configuration is the most user-friendly and the least restrictive for users.

5.4 DNS Traffic using TCP

As mentioned in the state of the art (see Section 2.6), some solutions can force the retransmission of DNS packets using the TCP protocol and not UDP. In other words, some solutions would force the use of the TCP protocol for domain name resolution. In the context of this thesis, this mitigation solution was not implemented. However, some advantages and some major disadvantages have been examined.

There are two main advantages to using the TCP protocol:

1. **IP address spoofing prevention**: Since the TCP protocol requires a three-way connection also known as a three-way handshake, this makes it more difficult for attackers to spoof the victim's source IP address or falsify other source IP addresses, thereby reducing the risk of spoofing attacks.

2. **Reduced amplification**: As seen in the script details above, UDP requests are fairly easy to build and send, allowing attackers to amplify requests without too much

effort. TCP, on the other hand, requires an established connection, as explained in the previous point, making packet amplification more difficult.

Three main disadvantages were also identified:

1. **Server overload**: Establishing and maintaining TCP connections is more costly in terms of server resources (CPU, memory) than UDP requests, which can increase the load on DNS servers. Indeed, resources are allocated to each TCP connection attempt, which is more costly than UDP being connectionless. This could also give rise to other types of attack, such as DRDoS. Indeed, if the attacker sends a large number of DNS requests via TCP through a botnet, establishing all the connections could overload the server's available resources. The victim here would therefore be the targeted DNS server.

2. **Increased latency**: The connection establishment process known as three-way handshake introduces additional latency into communication, which can slow down DNS name resolutions.

3. **Bandwidth consumption**: TCP connections consume more bandwidth due to control and acknowledgement mechanisms, increasing overall network traffic.

Finally, in a discussion with an incident responder[6] from the Easi company[7], another major drawback was identified. During a security incident at a company, an attacker had extracted the company's data via TXT records. However, as the DNS traffic passed under TCP, it was encrypted and therefore impossible to decrypt. As a result, it was impossible to establish which data had been extracted. The attack described here is called a DNS Tunneling attack.

Definition

"Tunneling is a protocol that allows for the secure movement of data from one network to another. It is an interconnection strategy that is used when the source and destination networks of the same type are connected through a network of different types. Tunneling is also known as port forwarding." [30]

[6] https://linkedin.com/in/jan-marc-munuku
[7] https://easi.net/en

Working of DNS Tunneling attack [30]

The DNS Tunneling attack is often used by attackers to exfiltrate internal data in a corporate network.
A DNS Tunneling attack uses a client-server model that hijacks the DNS protocol to transmit malware and other types of data. The process begins with the attacker registering a domain name. This domain name is configured to point to a server controlled by the attacker, where a tunneling application is running.

Once the domain name has been established, the attacker infects a target system, often located behind a corporate firewall, with malware. Since DNS queries are usually allowed through firewalls, the infected system can send DNS queries to the DNS resolver. The DNS tunneling client then sends a query to the DNS server, which then follows the classic DNS name resolution scheme. In the final phase of the DNS resolution process, the DNS resolver directs the query to the IP address controlled by the attacker, where the tunneling server program is running. This establishes a connection between the tunneling client software on the victim machine and the tunneling server program on the attacker's infrastructure, via a tunnel through the recursive resolver.

This DNS tunnel makes it possible to encode and transmit data from other protocols within DNS queries, facilitating data exfiltration or other malicious actions, such as command and control (CnC) communications or the installation of additional malware. The absence of a direct connection between attacker and victim makes DNS tunnel detection more complex. In theory, as long as data is encoded correctly and does not exceed UDP packet size limits, any type of data can be sent through DNS queries.

Chapter 6
Future work

In the context of future work, several avenues of improvement and exploration can be envisaged to deepen and enrich the study of DNS amplification attacks. The following proposals aim to improve the tools developed, diversify experimental scenarios and extend the analysis of mitigation and detection measures.

1. **Improving the various scripts**: A first step would be to optimize the code of the scripts developed for attack simulation and data collection. These improvements could include optimizing data processing algorithms, reducing resource consumption and improving execution speed. More powerful code would enable more complex experiments to be carried out and more accurate results to be obtained.

2. **Improved web interface**: The web interface used to monitor DNS traffic and victim machine performance could benefit from an improved design. A more intuitive and efficient interface would make it easier to observe and analyze data in real time. Additional features, such as interactive graphics and customizable dashboards, could also be integrated to enrich the user experience.

3. **Configuration variation**: It would be beneficial to carry out configuration variations in the virtual environment, particularly concerning the DNS server and the victim machine. By modifying various parameters such as DNS server configuration, DNS zone configuration, network configurations and security policies, it would be possible to observe a wider range of amplification factors and results. These variations would enable more detailed comparisons and a better understanding of the impact of each configuration on the attack.

4. **Broader study of mitigation measures**: A thorough analysis of all possible mitigation measures against DNS amplification attacks is essential. This study should include a detailed assessment of the effectiveness of each measure in different scenarios. The aim is to identify the most robust mitigation strategies and propose recommendations for their implementation.

5. **Detection measures**: In addition to mitigation measures, it is crucial to explore techniques for detecting DNS amplification attacks. By analyzing the effectiveness of different detection systems, we can develop preventive solutions capable of identifying and neutralizing attacks before they cause significant damage.

6. **Development of Machine Learning algorithms**: The application of Machine Learning could revolutionize the management of mitigation measures. By developing algorithms capable of learning and adapting to legitimate user behavior and network load, it would be possible to automatically configure mitigation measures for optimum efficiency. These algorithms could detect abnormal patterns and adjust protections in real time.

7. **Enhancing the virtual laboratory**: For a more realistic simulation, it would be appropriate to enhance the virtual laboratory by integrating elements of a complete

corporate network, including a demilitarized zone (DMZ), for example. This would make it possible to simulate more complex attacks and observe their impact on a more diversified network infrastructure.

8. **Deployment within the Royal Military Academy's Cyber Range, CyRange**: Finally, deployment of this attack and study scenario within the Royal Military Academy's Cyber Range, CyRange, would provide a realistic and secure training environment. This deployment would enable mitigation and detection strategies to be tested and validated in an operational setting, thus contributing to the training of future cybersecurity experts.

By pursuing these lines of research, it will be possible not only to deepen our understanding of DNS amplification attacks, but also to develop practical and effective solutions to protect IT systems against these growing threats.

Chapter 7
Conclusions

This Master's thesis in cybersecurity explored in depth the dynamics and implications of DNS amplification attacks in a virtual environment. Through a rigorous and methodological approach, it has been possible to develop a detailed understanding of these attacks, examining their mechanisms, their impact, and possible mitigation strategies.

The study began by presenting the motivations and background behind research into DNS amplification attacks. The growing threats posed by DDoS attacks, particularly those exploiting DNS vulnerabilities, underlined the importance of in-depth research in this area. Notable incidents, such as the attacks on Dyn in 2016 and Spamhaus in 2013, have demonstrated the seriousness of these threats.

In the first part of this thesis, a virtual laboratory environment was set up, designed to be particularly vulnerable, enabling optimal observation of attack mechanisms and their impacts. This infrastructure was configured to reproduce and observe results found in the scientific literature, providing a solid basis for empirical comparisons.

Specific scripts were then developed to faithfully reproduce the attack scenarios, perform various amplification measurements and monitor DNS traffic on the victim's side. These tools enabled a detailed analysis of the different types of DNS queries and their amplification potential, comparing the results with those in the literature and exploring various factors influencing amplification rates.

The final part of this study was the evaluation of mitigation techniques. The effectiveness of various prevention and response measures, such as the optimal configuration of DNS servers and the implementation of mechanisms to limit amplification factors, was analyzed. The results enabled us to identify the most robust strategies and propose recommendations for their practical implementation.

In addition, suggestions were made for future work, aimed at improving the tools developed, diversifying the experimental scenarios, and extending the analysis of mitigation and detection measures. These suggestions include optimizing the script code, improving the design of the monitoring web interface, varying the configurations of the virtual environment, and applying Machine Learning to automatically configure mitigation measures. Enhancing the virtual laboratory to simulate attacks against complete enterprise networks, and deploying these scenarios within the Royal Military Academy's Cyber Range, CyRange, were also suggested.

In conclusion, this thesis has made a significant contribution to the understanding and mitigation of DNS amplification attacks. The results obtained and the recommendations made offer concrete avenues for strengthening the security of IT systems in the face of these threats. This work will serve as a basis for future research and will contribute to

the training of cybersecurity professionals, enabling critical infrastructures to be better protected against DDoS attacks.

Bibliography

[1] Anagnostopoulos, M., Kambourakis, G., Gritzalis, S., Yau, D.K.: Never say never: Authoritative TLD nameserver-powered DNS amplification. In: NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium. pp. 1–9. IEEE (2018) [Cited on pages 17 and 35.]

[2] Anagnostopoulos, M., Kambourakis, G., Kopanos, P., Louloudakis, G., Gritzalis, S.: DNS amplification attack revisited. Computers & Security 39, 475–485 (2013) [Cited on pages 16, 35, and 36.]

[3] Anagnostopoulos, M., Lagos, S., Kambourakis, G.: Large-scale empirical evaluation of DNS and SSDP amplification attacks. Journal of Information Security and Applications 66, 103168 (2022) [Cited on pages 11, 15, 17, and 44.]

[4] Cloudflare: DNS Amplification DDoS Attack, `https://www.cloudflare.com/learning/ddos/dns-amplification-ddos-attack/` [Cited on page 17.]

[5] Cloudflare: DNS Server Types (Accessed December 16, 2023), `https://www.cloudflare.com/learning/dns/dns-server-types/` [Cited on page 4.]

[6] Cloudflare: DNS Zone (Accessed December 16, 2023), `https://www.cloudflare.com/learning/dns/glossary/dns-zone/` [Cited on page 4.]

[7] Cloudflare: Primary and Secondary DNS (Accessed December 16, 2023), `https://www.cloudflare.com/learning/dns/glossary/primary-secondary-dns/` [Cited on page 6.]

[8] Cloudflare: How DNSSEC Works (N/A), `https://www.cloudflare.com/dns/dnssec/how-dnssec-works/`, Accessed on March 31, 2024 [Cited on pages V, VI, 6, 8, 9, and 63.]

[9] Cloudflare: Root Signing Ceremony (N/A), `https://www.cloudflare.com/dns/dnssec/root-signing-ceremony/`, Accessed on March 31, 2024 [Cited on page 9.]

[10] Cloudflare: HTTP flood attack (nd), `https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/` [Cited on page 12.]

[11] Cloudflare: SYN Flood DDoS Attack (nd), `https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/` [Cited on page 11.]

[12] Cloudflare: UDP Flood DDoS Attack (nd), `https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/` [Cited on page 13.]

[13] Cybersecurity and Infrastructure Security Agency (CISA): DNS Amplification Attacks (2019), `https://www.cisa.gov/news-events/alerts/2013/03/29/dns-amplification-attacks`, accessed: 2024-05-20 [Cited on page 20.]

[14] Damas, J., Neves, F.: Extension Mechanisms for DNS (EDNS(0)). RFC 6891, RFC Editor (April 2013), Available Online: `https://datatracker.ietf.org/doc/html/rfc6891`RFC 6891 [Cited on page 6.]

[15] Dong, S., Abbas, K., Jain, R.: A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments. IEEE Access 7, 80813–80828 (2019) [Cited on page 11.]

[16] Douligeris, C., Mitrokotsa, A.: DDoS attacks and defense mechanisms: classification and state-of-the-art. Computer networks 44(5), 643–666 (2004) [Cited on page 11.]

[17] Fortinet: 10 Simple Ways to Mitigate DNS-Based DDoS Attacks (2016), https://www.fortinet.com/blog/threat-research/10-simple-ways-to-mitigate-dns-based-ddos-attacks, accessed: 2024-05-20 [Cited on pages VI, 18, 19, and 20.]

[18] Imperva: DNS Amplification. https://www.imperva.com/learn/ddos/dns-amplification/, accessed: 2024-05-20 [Cited on page 20.]

[19] Internet Engineering Task Force (IETF): Providing Minimal-Sized Responses to DNS Queries That Have QTYPE=ANY. https://datatracker.ietf.org/doc/html/rfc8482 (2019), accessed: 2024-05-20 [Cited on pages 20 and 48.]

[20] Internet Systems Consortium (ISC): BIND Best Practices: Authoritative, https://kb.isc.org/docs/bind-best-practices-authoritative, accessed: 2024-05-20 [Cited on pages 20 and 47.]

[21] Internet Systems Consortium (ISC): BIND Best Practices: Recursive, https://kb.isc.org/docs/bind-best-practices-recursive, accessed: 2024-05-20 [Cited on page 20.]

[22] Internet Systems Consortium (ISC): BIND 9 DNSSEC guide (N/A), https://bind9.readthedocs.io/en/latest/dnssec-guide.html, Accessed on March 31, 2024 [Cited on pages V and 9.]

[23] Ismail, S., Hassen, H.R., Just, M., Zantout, H.: A review of amplification-based distributed denial of service attacks and their mitigation. Computers & Security 109, 102380 (2021) [Cited on pages VI, 13, 14, 15, 35, 36, and 44.]

[24] Kührer, M., Hupperich, T., Rossow, C., Holz, T.: Hell of a handshake: abusing {TCP} for reflective amplification {DDoS} attacks. In: 8th USENIX Workshop on Offensive Technologies (WOOT 14) (2014) [Cited on pages VI, 14, and 15.]

[25] MacFarland, D.C., Shue, C.A., Kalafut, A.J.: Characterizing optimal DNS amplification attacks and effective mitigation. In: Passive and Active Measurement: 16th International Conference, PAM 2015, New York, NY, USA, March 19-20, 2015, Proceedings 16. pp. 15–27. Springer (2015) [Cited on page 20.]

[26] Majkowski, M.: RFC8482 - Saying Goodbye to "ANY" (2019), https://blog.cloudflare.com/rfc8482-saying-goodbye-to-any [Cited on pages 20 and 48.]

[27] Mockapetris, P.: Domain Names - Concepts and Facilities. RFC 1034, RFC Editor (November 1987), Available online: https://datatracker.ietf.org/doc/html/rfc1034 [Cited on page 4.]

[28] Mockapetris, P.: Domain Names - Implementation and Specification. RFC 1035, RFC Editor (November 1987), Available online: https://datatracker.ietf.org/doc/html/rfc1035 [Cited on pages 5 and 6.]

[29] PhoenixNAP: VirtualBox vs VMware: What's the Difference? (2021), https://phoenixnap.com/kb/virtualbox-vs-vmware [Cited on page 22.]

[30] Rajendran, B., et al.: Dns amplification & dns tunneling attacks simulation, detection and mitigation approaches. In: 2020 International Conference on Inventive Computation Technologies (ICICT). pp. 230–236. IEEE (2020) [Cited on pages 54 and 55.]

[31] van Rijswijk-Deij, R., Sperotto, A., Pras, A.: DNSSEC and its potential for DDoS attacks: a comprehensive measurement study. In: Proceedings of the 2014 Conference on Internet Measurement Conference. pp. 449–460 (2014) [Cited on pages 17, 34, 36, 44, and 45.]

[32] Rossow, C.: Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In: NDSS. pp. 1–15 (2014) [Cited on pages VI, 13, 15, 36, and 44.]

[33] Spring, J., Metcalf, L.: Probable Cache Poisoning of Mail Handling Domains. Carnegie Mellon University, Software Engineering Institute's Insights (blog) (2014), https://insights.sei.cmu.edu/blog/probable-cache-poisoning-of-mail-handling-domains/, Accessed: 2024-Mar-30 [Cited on page 6.]

[34] Vishwakarma, R., Jain, A.K.: A survey of DDoS attacking techniques and defence mechanisms in the IoT network. Telecommunication Systems 73, 3–25 (2020) [Cited on page 11.]

Appendix A

DNSSEC Protocol

A.1 Chain of trust mechanism

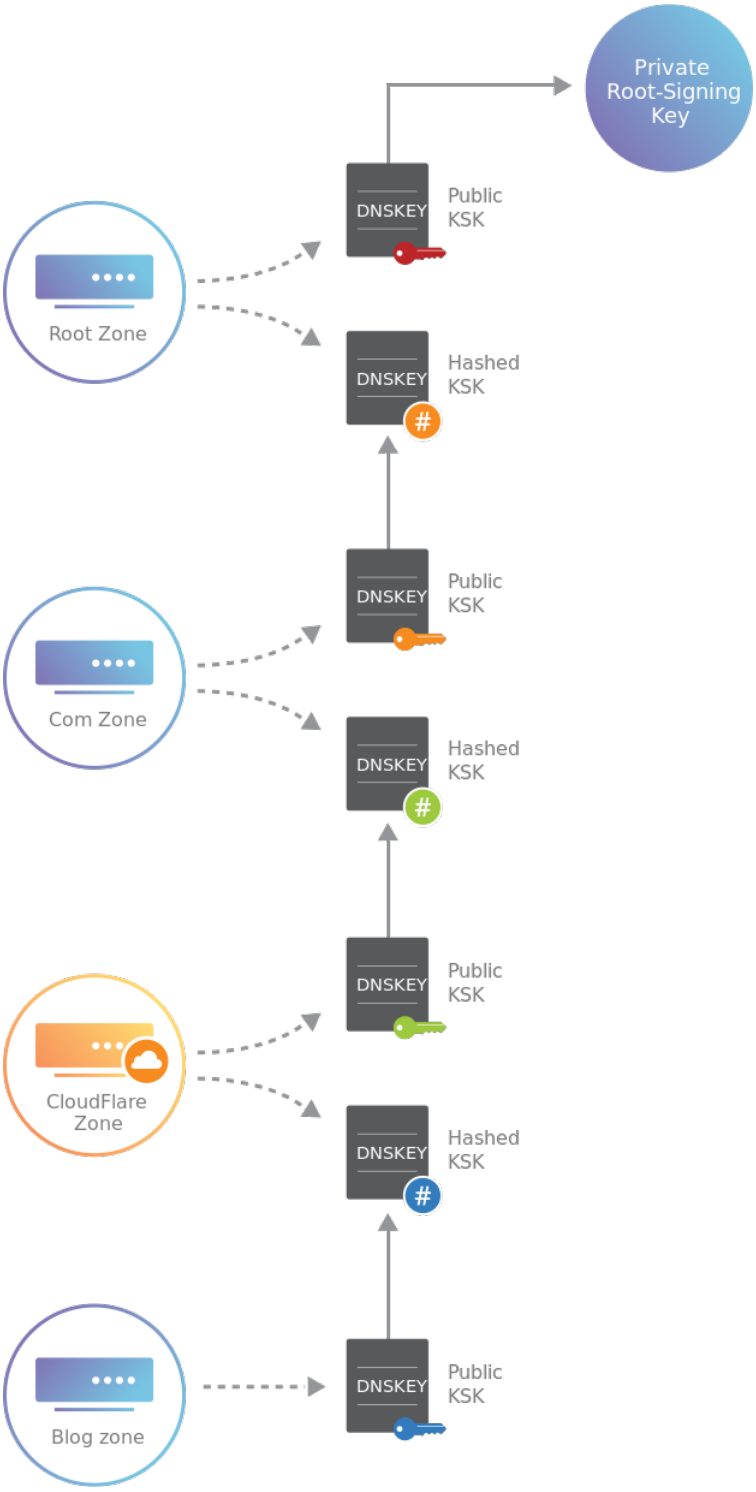


Figure A.1: Chain of trust mechanism [8]

Appendix B

DNS Server Configuration

B.1 Domain Transfer Zone

```
;
$TTL    604800
@       IN      SOA     ns1.amaury.thesis.io. root.amaury.thesis.io. (
                              6         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;

; NS record for name severs
@       IN      NS      ns1.amaury.thesis.io.

; A records for name servers
@       IN      A       192.168.68.53
ns1     IN      A       192.168.68.53

; Mail handler or MX record for the domain amaury.thesis.io
amaury.thesis.io.       IN      MX      10      mail.amaury.thesis.io.

; A records for domain names
amaury.thesis.io        IN      A       192.168.68.53
mail    IN      A       192.168.68.63
www     IN      A       192.168.68.53
mail2   IN      A       192.168.68.73
mail3   IN      A       192.168.68.83
; CNAME record
ftp     IN      CNAME   www.amaury.thesis.io.

; Addition of multiple records for different DNS query types
; A records
@       IN      A       192.168.68.54
        IN      A       192.168.68.55
        IN      A       192.168.68.56

; AAAA records
@       IN      AAAA    0000:0000:0000:0000:0000:ffff:c0a8:4436         ; 192.168.68.54 -> ipv6
        IN      AAAA    0000:0000:0000:0000:0000:ffff:c0a8:4437         ; 192.168.68.55 -> ipv6
        IN      AAAA    0000:0000:0000:0000:0000:ffff:c0a8:4438         ; 192.168.68.56 -> ipv6
        IN      AAAA    0000:0000:0000:0000:0000:ffff:c0a8:4439         ; 192.168.68.57 -> ipv6
        IN      AAAA    0000:0000:0000:0000:0000:ffff:c0a8:443a         ; 192.168.68.58 -> ipv6

; CNAME record
services        IN      CNAME   www.amaury.thesis.io.
blog            IN      CNAME   www.amaury.thesis.io.
app             IN      CNAME   www.amaury.thesis.io.
test            IN      CNAME   www.amaury.thesis.io.

; MX records
@       IN      MX      20      mail2.amaury.thesis.io.
        IN      MX      30      mail3.amaury.thesis.io.

; TXT records
@       IN      TXT     "DNS is a critical network service that translates humand-readable domain names into IP addresses and vice versa. It operates on a distributed database across servers worldwide."
        IN      TXT     "The Domain Name System (DNS) helps navigate the internet by mapping domain names to IP addresses, allowing users to access websites and other internet resources easily."
        IN      TXT     "DNS employs a hierarchical structure and operates through various record types like A, AAAA, CNAME, MX, etc., enabling efficient domain name resolution and internet functionality."
```

Figure B.1: Domain Transfer Zone Configuration

B.2 Reverse Lookup Zone

```
;
; BIND reverse data file for local loopback interface
;
$TTL    604800
@       IN      SOA     ns1.amaury.thesis.io. root.amaury.thesis.io. (
                              5         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;

; name servers - NS records
        IN      NS      ns1.amaury.thesis.io.

; PTR records
53      IN      PTR     ns1.amaury.thesis.io.
53      IN      PTR     amaury.thesis.io.
63      IN      PTR     mail.amaury.thesis.io.
53      IN      PTR     www.amaury.thesis.io.
73      IN      PTR     mail2.amaury.thesis.io.
83      IN      PTR     mail3.amaury.thesis.io.
```

Figure B.2: Reverse Lookup Zone Configuration