University of Brasília (UnB)

Institute of Exact Sciences

Department of Computer Science

Paulo Victor França de Souza

Amplified Reflection Attacks Over TCP Exploiting Middleboxes

Advisor

Prof. João José Costa Gondim

Brasília

2025

Dedication

I dedicate this work to my family, whose support was essential throughout my journey, to everyone who encouraged, supported, and contributed to my education during my undergraduate studies, and to the information security community.

Acknowledgements

I am deeply grateful to my mother, Ester, and my father, Clécio, for their love, support, and inspiration throughout my journey at the University of Brasília (UnB). My sincere gratitude also goes to Isabelle, a special person who has always been by my side, offering unconditional support and companionship throughout my years of study. I express my gratitude to Professor João José Costa Gondim, whose guidance was essential in choosing the topic and directing it, as well as to the Institutional Scientific Initiation Scholarship Program (PIBIC), for the financial support that made this work possible. I am grateful to my friends, the people I met at my program's Academic Center, and all those I met throughout university, who encouraged me and made this journey easier, with laughter, conversations, and valuable advice. I am also deeply grateful to the university environment at UnB, where the various student initiatives organized university parties that were essential to my academic experience, helping me maintain emotional balance by reconciling study time with leisure time, whether as a DJ, which contributed to my development in this area, or simply enjoying moments with people.

Abstract

First described in 2021 and confirmed in real-world cases, the emerging technique of TCP-based amplified reflection attacks exploiting middleboxes represents a growing concern within the cybersecurity community. This malicious approach leverages intermediary devices, such as firewalls and proxies, which can be improperly used as unintentional amplification vectors in distributed denial-of-service (DDoS) campaigns. This work presents the theoretical foundations of DoS/DDoS attacks, the TCP protocol, and the role of middleboxes in this context, along with a review of previous research and known CVEs from vendors such as Forcepoint, Fortinet, and Palo Alto. Focusing on experimentation, practical tests were carried out using attack code and a laboratory environment composed of pfSense firewalls (with pfBlockerNG, Squid, and SquidGuard) and FortiGate, enabling the assessment of the technique's effectiveness in different scenarios and the behavior of the devices under attack. Additionally, a scan of the Brazilian IP space was conducted using the ZMap tool with customized TCP packets to identify vulnerable middleboxes. Based on the collected data, mitigation measures, study limitations, and future directions are discussed, including ongoing improvements to the attack code in collaboration with its original developer. The main contributions of this work are to deepen the understanding of a still underexplored threat and to strengthen defense strategies in modern networks.

Keywords: amplified reflection attacks, middleboxes, DoS, DDoS, denial of service, networks, cyber security

Sumário

Lista de Figuras

List of Tables

List of Abbreviations and Acronyms

ACK Acknowledgement.

ACL Access Control List.

AI Artificial Intelligence.

BSD Berkeley Software Distribution.

CDF Cumulative Distribution Function.

CIDR Classless Inter-Domain Routing.

CISA Cybersecurity and Infrastructure Security Agency.

CLDAP Connectionless Lightweight Directory Access Protocol.

CNA Captive Network Assistant.

CoAP Constrained Application Protocol.

CSV Comma-Separated Values.

CTIR Gov Centro de Prevenção, Tratamento e Resposta a Incidentes Cibernéticos de Governo.

CVE Common Vulnerabilities and Exposures.

CVSS Common Vulnerability Scoring System.

CWE Common Weakness Enumeration.

DDoS Distributed Denial of Service.

DHCP Dynamic Host Configuration Protocol.

DLP Data Loss Prevention.

DMZ Demilitarized Zone.

DNS Domain Name System.

DoS Denial of Service.

DPI Deep Packet Inspection.

FIN Finish.

FTP File Transfer Protocol.

GB GygaByte.

Gbps Gigabits per second.

HTML Hypertext Markup Language.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

ICMP Internet Control Message Protocol.

IDS Intrusion Detection System.

IoT Internet of Things.

IP Internet Protocol.

IPS Intrusion Prevention System.

IPv4 Internet Protocol version 4.

LAN Local Area Network.

Mbps Megabits per second.

Mpps Million packets per second.

NAT Network Address Translation.

NGFW Next-Generation Firewall.

NTP Network Time Protocol.

NVD National Vulnerability Database.

OSI Open Systems Interconnection.

PSH Push.

RFC Request For Comments.

RST Reset.

SMTP Simple Mail Transfer Protocol.

SNMP Simple Network Management Protocol.

SSDP Simple Service Discovery Protocol.

SSH Secure Shell.

SYN Synchronize.

TCP Transmission Control Protocol.

TTL Time To Live.

UDP User Datagram Protocol.

UnB Universidade de Brasília.

URL Uniform Resource Locator.

VM Virtual Machine.

VoIP Voice over Internet Protocol.

VPN Virtual Private Network.

WAN Wide Area Network.

ZTNA Zero Trust Network Access.

Capítulo 1

Introduction

In recent years, cybersecurity professionals tasked with mitigating and responding to Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks have faced increasingly complex challenges. These challenges are driven by the rapid evolution of more sophisticated attack techniques, including new forms of DDoS attacks based on reflection and amplification [1].

Traditionally, amplified reflection attacks were strongly associated with the UDP protocol, known for not requiring connection establishment, which makes it particularly vulnerable to IP address spoofing and the generation of amplified responses [2]. However, these attacks can also be carried out using the TCP protocol. One way of exploiting this protocol was demonstrated in the 2021 study by Bock et al. [3], who showed that reflection and amplification attacks can occur by exploiting inadequate behavior of intermediate devices, known as *middleboxes*. In certain cases, these devices inspect and respond to TCP packets even without an established connection [4].

Middleboxes are widely used in modern networks for functions like traffic monitoring, filtering, and transformation, being common in *firewalls*, *proxies*, and intrusion prevention systems (IPS) [5]. When misconfigured or implemented outside the TCP protocol standards, these devices can become involuntary vectors for attacks. In this technique, the attacker sends TCP packets with spoofed source IP addresses (using the victim's IP), and the *middlebox* responds, believing it is communicating with a legitimate client. The result is an amplified response sent directly to the victim, hiding the attack's origin and significantly increasing the received traffic.

1.1 Justification

Despite the availability of advanced security solutions, such as Next-Generation Firewalls (NGFW) from Check Point, Forcepoint, Fortinet, and Palo Alto, some versions of these

technologies still have vulnerabilities that can be exploited by attackers. These loopholes allow the devices to function as traffic amplifiers, enabling attacks that would have previously been more difficult to execute over the TCP protocol. The impact of these attacks can be devastating for companies, resulting in significant financial losses, unavailability of essential services, and reputational damage. Large corporations that depend on continuous operations, such as financial services, e-commerce platforms, and government infrastructures, can suffer millions in losses in a matter of hours due to service disruption.

Several attacks exploiting *middleboxes* have already been documented, such as in March 2022, when Akamai experts began identifying several distributed denial of service (DDoS) campaigns against their clients, with a focus on sectors such as banking, travel, gaming, media, and web hosting. These attacks involved SYN flooding and high traffic volumes, reaching up to 11 Gbps and 1.5 million packets per second (Mpps) [4]. Upon examining the TCP packets involved, it was identified that the technique employed corresponded to the TCP *middlebox reflection attack*, presented by Bock et al. in 2021 [3]. To date, this was the only prominent academic work identified on this type of attack, which reinforces the relevance of the ongoing thesis, dedicated to deepening the understanding of this technique and examining its implications in information security contexts.

As described by Akamai, the first attacks in this series peaked at 50 Mbps, but those responsible for these campaigns have improved their capabilities and adjusted their tactics. More recent attacks, which used the same *middlebox* vector, reached peaks of 2.7 Gbps and 11 Gbps, with the 11 Gbps attack reaching 1.5 Mpps. Although attacks using this technique are still smaller in scale compared to other DDoS vectors, there are clear signs that their popularity and intensity are increasing [4]. This trend shows that attackers are progressively adopting the *middlebox reflection* technique as a new resource in their attack arsenals, broadening the range of tools used in denial of service campaigns. Given this scenario, it is essential to deepen the study of *middlebox* exploitation in amplified DDoS attacks and investigate effective ways to mitigate this threat.

1.2 Objective

This work aims to demonstrate and analyze, through practical laboratory experiments, how amplified reflection attacks on the TCP protocol can be exploited in *middleboxes*. Initially, an attack algorithm was developed and applied in a controlled environment to reproduce and understand in detail the real operation of these attacks and the vulnerabilities of the devices involved.

From this experimental scenario, the study was expanded to a large-scale scan of the Brazilian internet space, with the objective of identifying the presence and degree of vulnerability of *middleboxes* in real networks in the country. The choice of the Brazilian space is justified by the importance of mapping concrete risks in the local context, since cyber attacks directly affect national infrastructures and users, and also makes it possible to create mitigation strategies aligned with the particularities of the internet in Brazil.

With this approach, which goes from the laboratory to real-world field analysis, the work aims to evaluate the potential impact of attacks in different environments, analyze the results obtained, and investigate the technological evolution in mitigating these threats, contributing to the development of specific security recommendations to protect vulnerable devices in Brazilian networks.

In this way, the goal is to strengthen the resilience of national communication networks against emerging threats, improving the quality and security of systems and promoting a more secure and reliable digital environment for local users and organizations.

1.3 Work Structure

This work is structured into chapters. Chapter 2 presents the theoretical foundations necessary for understanding amplified reflection attacks on the TCP protocol, addressing the role of *middleboxes*, DoS and DDoS attacks, as well as mitigation strategies specifically for this type of attack that exploits *middleboxes*. Chapter 3 reviews related studies and media repercussions of the attack, including documented vulnerabilities in CVEs and their practical implications. Chapter 4 describes the methodology used in the experiments, also detailing the configured test environment and the source code used. In Chapter 5, the methodology used in a scan carried out on the Brazilian internet is presented with the objective of identifying vulnerable *middleboxes* by sending custom TCP packets. Chapter 6 analyzes the results obtained in the laboratory and discusses the data collected during the scan, interpreting the data found. Finally, Chapter 7 brings together the main conclusions of the study and proposes future directions, such as conducting tests with different *firewalls* and the continuous improvement of the source code, in collaboration with the French developer of the tool used.

Capítulo 2

Conceptual Review

The following sections present the necessary concepts for the development of this work. The concepts of *Middleboxes*, the TCP protocol, and denial of service attacks are discussed.

2.1 Middlebox

Middleboxes are intermediate network devices that perform functions beyond traditional IP routing, such as packet inspection, filtering, and modification [3] [6]. They operate between end hosts, allowing for the monitoring or alteration of data flows in transit through techniques like Deep Packet Inspection (DPI) [4]. According to RFC 3234, these devices can terminate, redirect, or modify packets [7]. A recent study indicates that around 40% of observed network paths are already impacted by *middleboxes*, highlighting their growing presence and influence in current infrastructure [8].

These devices play essential roles in various contexts, whether as censorship tools or as security elements in corporate networks, acting as *firewalls*, Network Address Translators (NAT), load balancers, and intrusion detection systems (IDS) [5]. Their deployment can occur in ways such as *out-of-path*, where they only receive unidirectional traffic due to routing through different paths between end hosts, or through mirroring techniques, which enable unidirectional or bidirectional traffic delivery using packet mirroring or optical splitters, as illustrated in Figura 2.1. In censorship scenarios, these devices can block connections based on domains or keywords by injecting RST packets, providing fake DNS responses, or displaying block pages that show messages like "Access Denied"or "Content Blocked"[9].

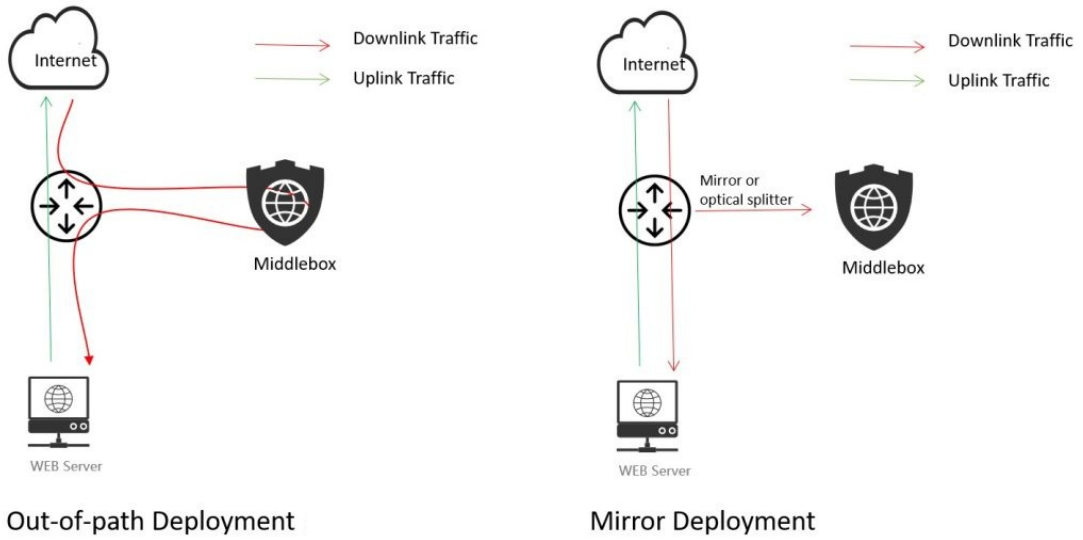

Figura 2.1: Difference between an *out-of-path* and a *mirroring middlebox* implementation.

These devices are fundamental for network performance, security, and control, being widely used in corporate environments, by operators, and in large-scale surveillance infrastructures, such as the "Great Firewall of China,"which blocks or filters access to foreign content and regulates online communication in the country [4] [10].

2.1.1 DPI (Deep Packet Inspection)

Deep Packet Inspection (DPI) is a technology typically used by *middleboxes* that allows for detailed analysis of network traffic, going beyond the traditional inspection limited to packet headers [11]. With DPI, it's possible to examine both headers and payloads, enabling the identification of threats, traffic origin, and the application involved [12] [4]. This analysis uses techniques like signatures, heuristics, and anomaly detection, in addition to filters and custom rules to block or redirect packets [12].

In addition to increasing security, DPI contributes to network optimization by classifying and prioritizing traffic, being especially useful in sensitive applications such as VoIP and critical corporate services [11]. In business environments, it is also used to enforce network usage policies and prevent threats from personal devices [12]. However, its application to encrypted data raises concerns about privacy, censorship, and the potential commodification of information [8]. Despite these controversies, DPI remains a strategic tool for the control and protection of digital networks [11].

2.1.2 Firewall and Next-Generation Firewall (NGFW)

Firewalls are *middleboxes*, implemented in *hardware* or *software*, that monitor, filter, and control network traffic according to predefined policies [13] [14]. They create a barrier between trusted and untrusted networks, allowing only authorized traffic and blocking malicious access, which is essential for ensuring the integrity, confidentiality, and availability of information against both external and internal threats [14].

Traditionally, *firewalls* use techniques like packet filtering, circuit proxy, and application proxy, operating at the transport and application layers. They are often deployed in demilitarized zones (DMZ) as part of defense-in-depth architectures, which can include bastion hosts, router filters, gateways, multiple *firewalls*, VPNs, and extranets, depending on the network topology [15].

Since the 1980s, *firewalls* have evolved from simple packet filters to advanced solutions with stateful inspection, intrusion detection, and Deep Packet Inspection (DPI) [13]. The introduction of what the market now calls *Next-Generation Firewalls* (NGFW) represented a significant advance, incorporating functionalities such as application and identity-based control, protection against advanced threats, cloud integration, and artificial intelligence (AI) analysis [14] [13].

These devices combine various security capabilities into a single platform, such as Intrusion Prevention System (IPS), data loss prevention (DLP), network segmentation, Zero Trust Network Access (ZTNA), protection for IoT devices, and sandboxing, protecting everything from on-premises *datacenters* to cloud-based infrastructures [14].

In addition to controlling traffic, *firewalls* also perform complementary functions, such as Network Address Translation (NAT), which hides internal addresses to make targeted attacks more difficult, and the creation of secure tunnels with VPNs for remote access. The AI-based features contribute to the analysis and response to threats in real-time [14] and are a current focus. Filtering rules are defined based on criteria such as IP addresses, ports, and protocols, organized into access control lists (ACLs) [13]. Adopting best practices, such as the principle of least privilege, proper documentation, firewall self-protection, and network segmentation, is essential to ensure its effectiveness [13].

2.1.3 Proxy

A *proxy* is a type of *middlebox* that can be integrated into *firewalls*. A *proxy* server acts as an intermediary in network communication, relaying packets between clients and servers [8]. It directs internet traffic, receiving requests from users and forwarding them to the desired destinations, and returning the responses to the clients. Additionally, modern

proxies offer functionalities such as *firewall*, content filtering, and caching, improving security and privacy [16].

These servers can mask the user's IP address, making tracking difficult and increasing security against cyberattacks. By hiding the real IP and inspecting incoming and outgoing data, *proxies* protect privacy and ensure an additional level of security. They can also be used to control internet access, save bandwidth, balance traffic, and prevent access to unwanted or potentially harmful sites [17]. Furthermore, they are often used to bypass geographical blocks, allowing access to location-restricted content [16].

There are different types of *proxies*, such as transparent, anonymous, distorting, and highly anonymous, each offering varying levels of privacy and specific functionalities [16]. The transparent *proxy* indicates to the site that it is a *proxy* and still transmits your IP address, being used for content control and filtering in corporate environments. The anonymous *proxy* also identifies itself as a *proxy* but does not pass on its IP, helping to protect your identity and prevent tracking, although it does not guarantee total anonymity. The distorting *proxy* identifies itself as a *proxy* and passes a fake IP, allowing you to appear to be in another location to bypass geographical restrictions. The highly anonymous *proxy* periodically changes the IP presented to the site, making tracking difficult and offering the highest level of privacy, as happens on the TOR network [16].

The choice of *proxy* type depends on the user's needs, such as organizational control, anonymity, performance, or large-scale data collection, and is strategically applied to protect organizational networks against unauthorized access and improve overall performance [17].

2.2 Transmission Control Protocol (TCP)

The *Transmission Control Protocol* (TCP) is a transport layer protocol of the TCP/IP model responsible for providing a reliable communication between processes in end systems, being widely used by applications like HTTP, FTP, SMTP, and SSH due to its robustness and data delivery guarantee [18] [19].

TCP operates in a connection-oriented manner, meaning a logical connection must be established before data transmission, using the process known as the *three-way handshake* [18] [9]. This *handshake* occurs in three steps: the client sends a segment with the SYN control bit activated, requesting the connection; the server responds with a segment containing SYN and ACK, accepting the request and providing its initial sequence number. Finally, the client sends an ACK segment, confirming the receipt and finalizing the connection establishment [9]. This exchange synchronizes sequence numbers and confirms that both sides are ready to start communication [18].

Once the connection is established, TCP divides the data into segments and transmits them with flow and congestion control, ensuring the receiver is not overwhelmed and the network is used efficiently. Furthermore, it ensures ordered and duplicate-free delivery through the sequence and acknowledgment number fields in the TCP header [18]. The TCP header, fundamental to the protocol's operation, contains fields such as source and destination ports, sequence number, acknowledgment number, window size (for flow control), and data offset (indicating the effective start of the data in the segment) [18].

In addition, TCP uses crucial control flags in connection management:

- **SYN** (Synchronize): initiates the connection, establishing sequence numbers;
- **ACK** (Acknowledgment): confirms the receipt of data or requests, always present after connection establishment;
- **FIN** (Finish): indicates that the sender has finished sending data, requesting connection termination;
- **RST** (Reset): immediately terminates the connection, usually due to an error or communication failure;
- **PSH** (Push): requests that the data be processed and passed immediately to the application [18].

When communication ends, connection termination usually occurs through a four-segment exchange, ensuring that both parties have safely finished transmission [18]. The three phases of a TCP communication can be seen in Figura 2.2.

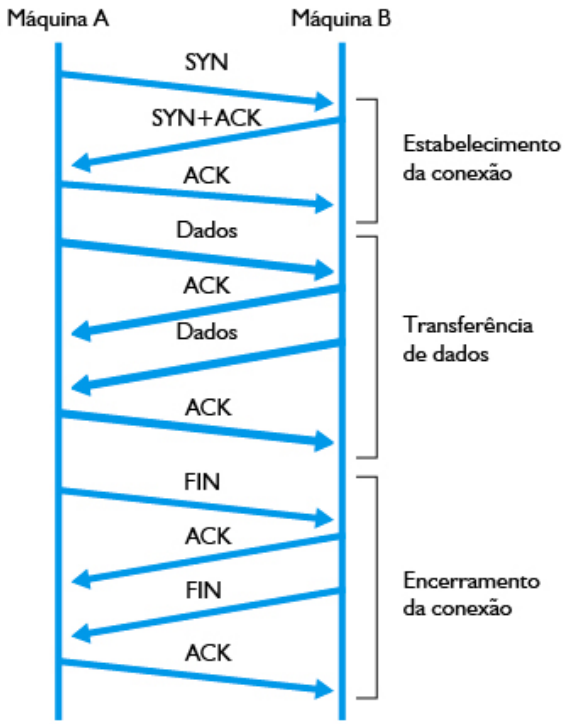

Figura 2.2: Three phases of a communication using TCP.

TCP, in conjunction with the IP protocol, ensures the reliability of data transmission at the transport layer of the TCP/IP model, being responsible for guaranteeing the correct delivery of packets, reordering them, detecting losses, and performing retransmissions when necessary, while IP addresses and forwards the packets between devices. This collaboration between the two protocols is fundamental for efficient communication in modern networks, including the internet, which is structured by the data link, internet, transport, and application layers [20] [19]. Furthermore, this foundation is essential for implementing multi-layer security solutions, such as *firewalls* and VPNs, ensuring secure and robust communication [19].

In contrast, the UDP (*User Datagram Protocol*) protocol, also belonging to the transport layer, offers a connectionless communication, prioritizing speed and low latency at the expense of reliability. Therefore, it is used in applications like VoIP, DNS, and streaming [19]. In Figura 2.3, you can see a bit of the difference between UDP and TCP in a communication.

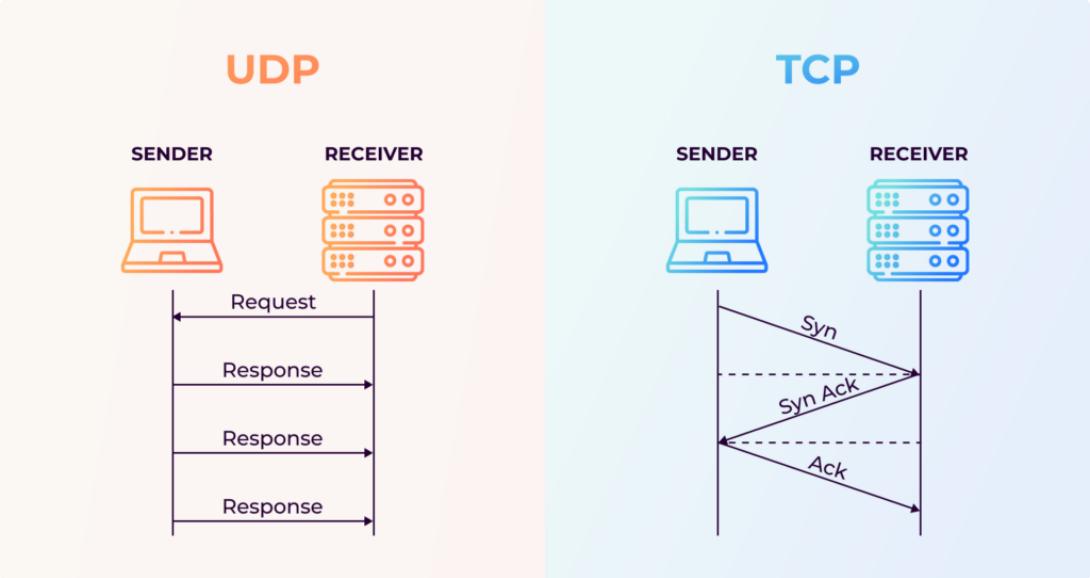

Figura 2.3: Difference between UDP and TCP communication.

2.3 Denial of Service (DoS)

A denial of service (DoS) attack occurs when a malicious actor uses a single machine to overload a computer, online service, or network resource, making it unavailable [21]. Among the common types of DoS attacks, flooding techniques stand out, such as *Smurf attack*, *Ping flood*, *Ping of Death*, and the SYN attack, which exploits the TCP *three-way handshake* to keep connections open and consume server resources [22] [21].

DoS attacks can typically be classified as volumetric or low-and-slow. Volumetric attacks, also known as floods, aim to saturate network bandwidth or exhaust system resources by sending large amounts of traffic, usually measured in gigabits per second (Gbps) [23]. These attacks target the network and transport layers (layers 3 and 4, respectively) of the OSI model [24] and use protocols like UDP, ICMP, or DNS, often with amplification or IP address spoofing, making mitigation difficult [25].

On the other hand, low-and-slow attacks operate at the application layer (layer 7 of the OSI model [24]) and are distinguished by simulating legitimate requests at a low frequency, with the goal of consuming server resources without generating abnormal traffic spikes [23]. By keeping connections open for long periods or performing slow and continuous interactions, these attacks can overload web servers, *firewalls*, and load balancers, often going unnoticed by traditional volume-based detection systems [26].

Defenses against DoS attacks are divided into four main categories: prevention, detection, source identification, and reaction. Prevention aims to stop attacks from reaching

the target, while detection seeks to recognize their occurrence. Source identification aims to trace the origin of the malicious traffic, and reaction seeks to mitigate or eliminate its effects [27].

2.3.1 Distributed Denial of Service (DDoS)

The distributed denial of service (DDoS) attack is a more advanced form of DoS, characterized by a coordinated action that uses multiple devices to overload systems or networks with excessive traffic, compromising their availability [25]. These attacks generally exploit vulnerabilities present in technologies used by clients, servers, or intermediate devices [28].

Source identification is made difficult by the use of spoofed IP addresses, which complicates tracing those responsible [25]. Vulnerable protocols, such as DNS and NTP, are commonly used for traffic amplification, producing data volumes significantly larger than those of the original requests [29]. Furthermore, the use of reflector servers intensifies the complexity of tracking [29] [1]. As a result, DDoS attacks can reach scales of gigabits per second, severely affecting the availability of essential services [25].

The motivations vary between hacktivism, extortion, cyber warfare, and unfair competition practices [25] [1]. The diversity of techniques used makes mitigation complex. Of note are UDP amplification attacks, which use protocols like DNS and NTP [29], and attacks spanning different vectors, such as those targeting the application layer, resource exhaustion attacks, and volumetric attacks, which generate massive traffic to overload the infrastructure [1].

To counter these threats, organizations can adopt strategies such as cloud mitigation, application *firewalls*, and hybrid approaches that integrate on-premises and remote solutions [4]. Advanced detection and prevention techniques are also fundamental to ensuring the continuity of services and the protection of critical data [1].

2.4 Amplified Reflection Attack

An amplified reflection attack is a denial of service technique in which the attacker exploits devices to send small requests with a spoofed IP address that points to the victim. These devices, upon processing the requests, generate significantly larger responses and redirect the amplified traffic to the victim, which results in a substantial increase in the volume of data received and makes attack mitigation difficult due to the complexity of distinguishing legitimate from malicious traffic [30] [31].

This type of attack combines two techniques: reflection, in which the attacker sends a request to a device (reflector) with a spoofed source IP address, causing the response to be sent to the victim, and amplification, in which the response generated by the reflector

is significantly larger than the original request, increasing the volume of traffic received by the victim and the impact of the attack [32] [33]. The exploitation can occur with both the UDP and TCP protocols, based on source address spoofing and response amplification. The difficulty in tracing the real origin of the traffic, due to the use of reflectors, makes this vector even more complex and challenging to mitigate [34].

The success of a reflection amplification attack depends on the amplification factor, as this represents the ratio between the volume of the generated response and the volume of the request, as shown in Equação 2.1, and can be measured in bytes or in the number of packets and varies according to the protocol used [32]. These attacks, which do not require prior device infection, become simpler by only requiring the identification of accessible reflectors and the sending of forged packets [30]. Mitigation requires a comprehensive approach, which includes blocking packets with spoofed IPs, source rate limiting, port control, and the use of signature filters [35].

2.4.1 Amplified Reflection Attacks over UDP

Amplified reflection attacks are particularly effective when they exploit UDP-based services, such as DNS, NTP, SNMP, SSDP, CLDAP, and others. This is due to the fact that these protocols do not verify the source address of the packets, which allows IP spoofing [33]. In IoT environments, protocols such as CoAP have also been used as vectors in amplified reflection attacks [32]. In addition, services like Memcached have also been used as attack vectors, allowing the reflected traffic to reach extremely high volumes, often reaching tens of millions of packets per second [36] [35].

2.4.2 Amplified Reflection Attacks over TCP

Although the UDP protocol is more commonly associated with amplified reflection attacks, studies show that devices can also be vulnerable to the TCP protocol, despite the complexity of the connection process due to the *three-way handshake*. This happens because many devices react unexpectedly when trying to establish the connection [35] [37]. Furthermore, incompatible or incomplete TCP stacks can be manipulated to generate responses without the need for a complete connection, making traditional detection and mitigation techniques ineffective when combined with IP spoofing and crafted packets [34].

An example of an amplified reflection attack using TCP is the exploitation of intermediate devices that respond in an amplified way to manipulated packets [3]. Techniques originally used in UDP, such as *carpet bombing*, can also be adapted for TCP, allowing attacks on multiple destinations simultaneously in a network or CIDR block, which ma-

kes detection and defense more difficult [35]. Another example is the TCP SYN+ACK reflection attack, in which forged SYN packets induce devices to respond with multiple SYN+ACK packets, significantly amplifying traffic even before the connection is fully established [35] [37].

2.5 Amplified Reflection Attack over TCP Exploiting Middleboxes

An amplified reflection attack over TCP exploiting *middleboxes* occurs when intermediate network devices, such as *firewalls*, *proxies*, or censorship systems, are manipulated to generate large volumes of TCP traffic against a victim, abusively exploiting the TCP protocol. This happens despite its theoretical resistance based on the three-way handshake mechanism (SYN, *SYN*+ACK, ACK), which traditionally protected it against this type of attack [9]. However, it has been shown that many *middleboxes* implement the TCP protocol incorrectly or incompletely, responding to out-of-state packets without requiring the full handshake, which paves the way for amplification attacks [38]. This type of attack has a significant amplification rate, defined as the ratio between the volume of data received by the victim and the volume of data sent by the attacker, as shown in Equação 2.1.

$$\text{Amplification Rate} = \frac{\text{Data received by the victim}}{\text{Data sent by the attacker}} \tag{2.1}$$

The attack consists of sending forged TCP packets containing malicious HTTP requests, usually aimed at blocked domains (such as pornography, social media, and others), to misconfigured *middleboxes*. As these devices frequently use Deep Packet Inspection (DPI) and are programmed to intercept access to restricted content, they end up generating responses such as TCP packets with extra data (*payloads* larger), HTTP headers (small responses that indicate the blocking or redirection status), or blocking pages (with HTML content, images, or scripts) directly to the forged IP address, i.e., the victim [4].

In addition to the direct response from the *middleboxes*, some devices also react to RST packets sent by the victim, resending previous data or keeping sessions open, which increases the load on the target [4]. This characteristic makes defense difficult, as the traffic passes through common ports like TCP 80 and carries valid HTTP packets, making filtering by signature or IP difficult [38]. Complete mitigation would require changes in the default behavior of *middleboxes*, firmware updates from manufacturers, and changes in the censorship policies of affected countries [38].

Although still less voluminous than UDP-based attacks, TCP attacks have shown growth in both frequency and impact, especially by enabling volumetric attacks with a fraction of the traffic previously required [4]. Initially considered only theoretical, amplified reflection attacks over TCP, with the use of *middleboxes*, have become a concrete and constantly evolving threat, currently actively monitored by security companies and incident response centers [38].

2.6 Mitigation Strategies for Amplified Reflection Attacks over TCP Exploiting Middleboxes

Amplified reflection attacks over TCP that exploit *middleboxes* represent a structural and complex problem that affects various network implementations and devices, with no single solution applicable to all contexts [3]. One of the main mitigation strategies consists of requiring the *middlebox* to observe both directions of communication and validate the complete establishment of the TCP handshake before injecting any response, making it difficult for the attacker to spoof the connection. This measure prevents the *middlebox* from automatically responding to forged packets, such as those of the SYN type containing improper payloads [3].

The use of SYN packets with a *payload* is considered a strong indicator of malicious traffic, as in legitimate contexts, these packets rarely contain data [4]. Thus, it is recommended that networks adopt blocking policies for these packets, in addition to dropping connections with *payloads* coming from ports 80 or 443. Although such measures may negatively impact some legitimate HTTP/HTTPS services, they are effective in preventing abuses [39]. ACL rules configured in *firewalls*, such as the following, are practical examples:

```
deny tcp any eq 80 host x.x.x.x match-all +SYN -ack packet-length
    gt 100
```

This rule discards anomalous SYN packets with more than 100 bytes coming from port 80, mitigating common attack vectors [4] [6].

Another relevant approach is to limit the size of the responses issued by *middleboxes*, using simple RST packets or minimal HTTP redirects. This significantly reduces the amplification factor [3] [6]. The simplification of blocking pages and the use of lean messages also contribute to this. Furthermore, it is recommended that devices only filter traffic originating from within the protected network and only respond to specific regions, restricting the scope of victims and limiting the use of the *middlebox* as a reflective attack vector [6].

Packet inspection tools like Snort can identify known signatures in blocking responses, allowing the blocking of malicious traffic based on unencrypted textual content [4]. Cloud mitigation solutions and Anti-DDoS systems with ACK verification are recommended for environments that need large-scale protection [39] [40]. Such systems can authenticate legitimate sessions, reducing the effectiveness of forged packets. The use of bidirectional *middleboxes* is especially relevant, as these devices analyze traffic from both the client and the server, and can detect spoofing attempts and validate established connections [6].

Another practical recommendation is to disable support for HTTP responses, as this protocol, in addition to being obsolete in many contexts, negatively interferes with the secure use of HTTPS [6]. When the source IP does not respond to legitimate communication attempts, RST packets should be sent to terminate the session and prevent the connection from remaining in an undefined state [39].

Finally, it is important to highlight that the TCP-*middlebox* reflection attack exploits characteristics that make traditional detection difficult: SYN packets with a *payload* and requests on multiple ports make scanning techniques like *nmap*, *telnet*, or *netcat* ineffective [41]. This reinforces the need for proactive and integrated mitigation strategies, considering both the nature of the packets and the expected behavior of network devices.

2.7 Chapter Summary

This chapter presented the essential theoretical foundations for understanding amplified reflection attacks over the TCP protocol exploiting *middleboxes*, addressing the concepts of DoS, DDoS, the functioning of the TCP protocol, in addition to the role of intermediate devices such as *firewalls*, NGFWs, *proxies*, and techniques such as DPI, often used in traffic inspection. The section provided the necessary conceptual basis to understand how these elements can be exploited by attackers to amplify malicious traffic, also highlighting mitigation strategies for this attack vector. In the next chapter, related studies and the repercussion of the attack in the specialized media will be presented, including documented vulnerabilities in CVEs and their practical implications in the context of network security.

Capítulo 3

Amplified Reflection Over TCP Exploiting Middleboxes

The amplified reflection attack technique over TCP, which exploits *middleboxes*, was initially described in 2021 by researchers from the University of Maryland and the University of Colorado Boulder [41]. This approach was presented at the USENIX Security 2021 conference, where the work received the Distinguished Paper Award [38]. Led by Kevin Bock, Abdulrahman Alaraj, Yair Fax, Kyle Hurley, and Dave Levin from the University of Maryland, and by Abdulrahman Alaraj and Eric Wustrow from the University of Colorado Boulder, the study was published in the article "*Weaponizing Middleboxes for TCP Reflected Amplification*"[3] and is described in Section 3.1.

3.1 The amplified reflection attack over TCP exploiting middleboxes

The referred article described amplified reflection attacks over TCP that are not limited to sending SYN packets, also being the first based on HTTP [38]. The main contribution was the discovery that *middleboxes*, especially those used in censorship regimes, can be induced to respond in an amplified manner to forged packets, even without the completion of the TCP handshake [38]. These responses, which are generally injected HTTP block pages as part of the censorship mechanism, can be exploited to generate traffic against the victim.

To identify packet sequences capable of provoking these responses, the authors used the Geneva tool, a genetic algorithm trained against censoring *middleboxes* [3]. The training of Geneva aimed to maximize the size of the responses obtained, resulting in five main sequences with high amplification potential [38]. Censored domains, such as

`www.youporn.com` and `plus.google.com`, were used to trigger the censorship mechanisms of 184 *middleboxes* identified through the CensoredPlanet project [38].

Infinite amplifications were identified and attributed to two main mechanisms: routing loops, where packets with an inadequate TTL circulate indefinitely, reactivating *middleboxes*; and sustained reflections by the victim itself, whose responses (such as RST packets) trigger new responses from the *middlebox*, perpetuating the cycle [38]. The traditional metric of amplification factor proved insufficient for these extreme cases, with the use of generated bandwidth being suggested as a more accurate metric, although difficult to measure ethically [38].

The attacks presented in the article explore different mechanisms for reflection and amplification of traffic directed at the victim [3]. The types of attacks described are illustrated in Figura 3.1, where the thick arrows represent the amplified traffic flows, while the red arrows indicate the packets responsible for triggering the amplification. *Destination reflection* (a) occurs when an attacker (A) spoofs the victim's IP and sends a packet to a destination (D), which responds to the victim (V), with the amplification (thick arrow) being the response larger than the stimulus (red arrow). In *Middlebox reflection* (b), an intermediate device *middlebox* (M) is the one that reflects the amplified traffic to the victim. *Destination and middlebox reflection* (c) combines both, where both the destination and the *middlebox* respond to a single packet from the attacker, increasing the volume of the attack. *Routing loop reflection* (d) exploits a loop between routers (R) and a *middlebox*, causing a single packet from the attacker to generate multiple amplified responses. Finally, *Victim-sustained reflection* (e) represents the most dangerous scenario, where the victim's own response to the initial attack (such as an RST packet) triggers new amplified responses from the destination, creating a self-sustaining and potentially infinite attack cycle [3].

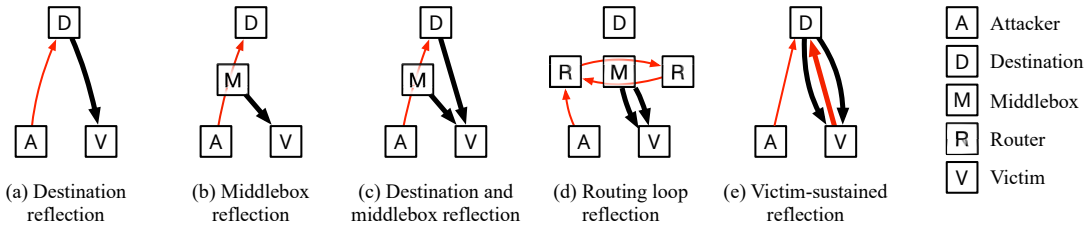

Figura 3.1: Types of amplified reflection attacks over TCP exploring middleboxes found by Bock et. al (Fonte: [3]).

The threat model adopted assumes an attacker completely off-path, who only spoofs the victim's IP address without interfering with the traffic, making the attack more realistic and difficult to mitigate [3]. In many cases, the generated traffic appears legitimate,

using TCP on port 80 with valid HTTP packets, which makes its detection by conventional mitigation systems difficult [38]. Furthermore, the attack can make it seem that the traffic came from any IP located behind the *middlebox*, making IP-based blocking ineffective.

The investigation also revealed that countries with stricter censorship regimes, such as China [10] and Saudi Arabia, have *middleboxes* that offer extremely high amplification factors [38]. However, even *middleboxes* located in countries without censorship can be exploited in the same way. It was found that infrastructures originally created to block unwanted content can be easily weaponized to generate DoS attacks against international targets, representing an unexpected threat to global internet security [3].

3.2 Repercussions in the media and online community

Several media outlets, including Brazilian sources like Canaltech [42] and international ones like Akamai [4] and Portswigger [43], have reported on amplified reflection attacks over TCP, which exploit *middleboxes*.

In March 2022, Akamai experts identified significant distributed denial of service (DDoS) campaigns targeting sectors such as finance, tourism, gaming, media, and hosting services. These attacks primarily used the TCP *middlebox* reflection technique [3], with *SYN* packet floods, reaching peaks of up to 11 Gbps and 1.5 million packets per second (Mpps) [4].

Although the attacks initially showed modest peaks, such as 50 *Mbps*, they evolved rapidly in sophistication and volume. Later, peaks of up to 2.7 and 11 Gbps were recorded, with the 1.5 Mpps rate constant in the most intense episodes [4]. Although these volumes are still lower than those of more traditional DDoS attacks, the growth of these incidents highlights their increasing adoption and the need for improved defense strategies.

A critical aspect is the high amplification factor, where a single 33-byte SYN packet can generate a response of up to 2,156 bytes, representing an amplification of approximately 65 times [4]. This behavior allows attackers to cause a major impact with little bandwidth consumption. Unlike conventional volumetric TCP attacks, which require large *botnets*, the exploitation of *middleboxes* is more accessible and efficient. Furthermore, certain intermediate devices, by ignoring RST packets sent by the victim, can generate practically infinite amplification, especially when the target uses active TCP ports [4].

In April 2022, NSFOCUS identified a TCP reflection attack involving *middleboxes* against a client of the cloud DDoS protection service in the Asia-Pacific region, reaching a peak of 7 Gbps [39]. The analysis revealed that the amplification was caused by HTTP

responses, with different amplification factors depending on the content returned [40]. The exploited vulnerability was in the incomplete detection of TCP connections by the *middleboxes*, allowing packets with fixed or zeroed sequence numbers to be accepted, making the devices susceptible to exploitation [40].

In December 2021, another attack occurred that affected a cloud services sector client, generating an attack bandwidth between 1 Gbps and 2 Gbps [44]. This type of attack, using *middleboxes* as amplifiers, represents a growing concern, as these devices make mitigation more difficult.

A study by *Shadowserver* revealed that approximately 18.8 million IPv4 addresses were vulnerable to TCP DDoS attacks through *middleboxes* [45]. Most of these responses were observed in China (more than 6.3 million), followed by Iran (about 5.2 million) and Indonesia (more than 2.7 million). This distribution can be seen in Figura 3.2 and Figura 3.3, which present, respectively, a global map and a hierarchical visualization of the scenario on April 23, 2022. The research identified amplifiers with extremely high amplification rates, reaching 6,583,549×, demonstrating the effectiveness of *middleboxes* in this type of attack [45].

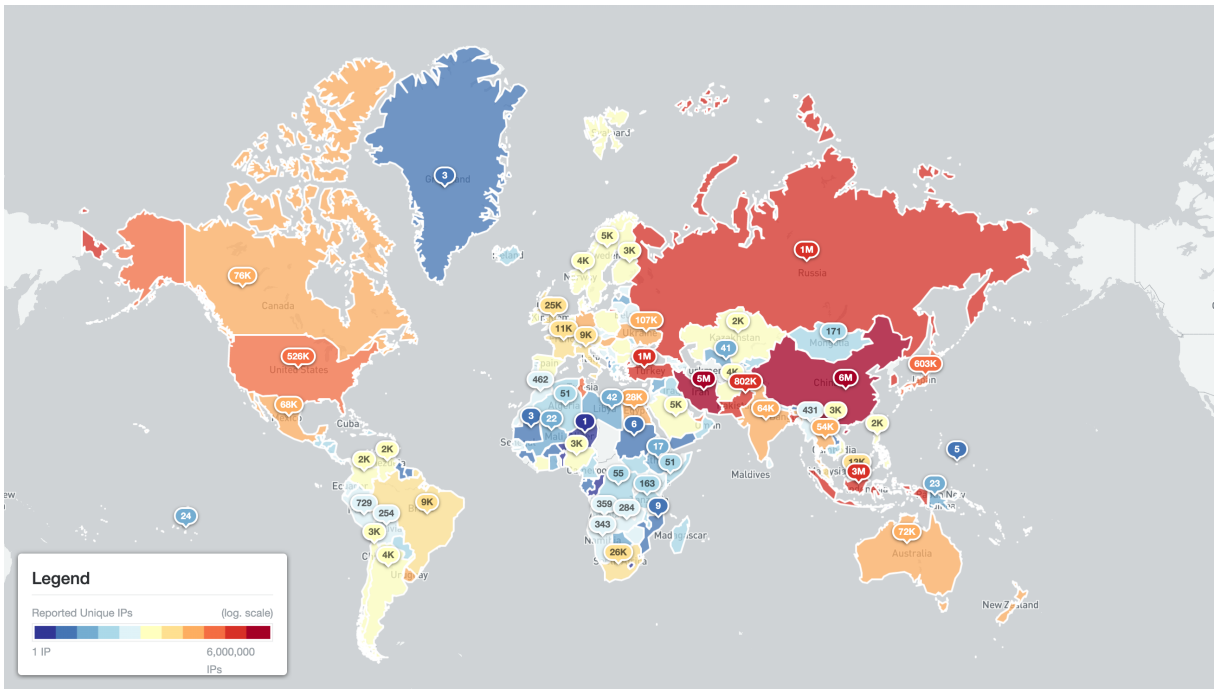

Figura 3.2: Exploitable *middleboxes* worldwide by unique IPv4 addresses (Fonte: [45]).

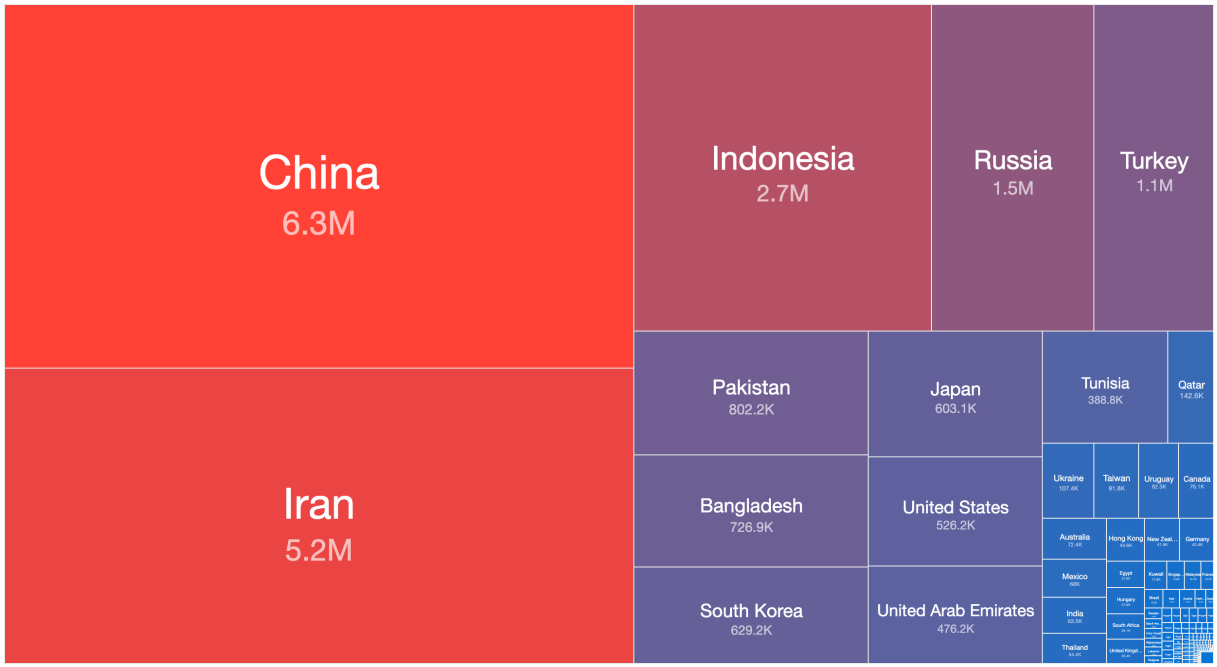

Figura 3.3: Countries with the most exploitable *middleboxes* in the world by unique IPv4 address count (Fonte: [45]).

3.3 CVEs and vendor solutions

Several security vulnerabilities (CVEs, *Common Vulnerabilities and Exposures*) associated with the amplified reflection attack over TCP, which exploit *middleboxes*, have been identified in security solutions from vendors such as Forcepoint, Fortinet, and Palo Alto. These companies have already provided fixes and updates to mitigate such flaws and strengthen the security of their products. However, given the complexity of these attacks, it is crucial to maintain continuous vigilance and adopt effective mitigation measures to ensure protection against future threats.

3.3.1 Forcepoint's CVE-2021-41530

The CVE-2021-41530 vulnerability affects the Forcepoint NGFW Engine in versions 6.5.11 and earlier, 6.8.6 and earlier, and 6.10.0, when the user HTTP response functionality is enabled [46]. This is a TCP-based amplified reflection flaw, where a SYN packet with a *payload* can trigger the generation of an HTML response by the *middlebox*, even without completing the TCP handshake [47]. The attacker can spoof the source IP address, causing the firewall's response to be sent to a random victim, which can result in an overload of unsolicited traffic [47].

This type of attack only occurs if the NGFW is configured with an HTTP response for internet requests, which is not a common practice but can happen inadvertently, where the amplification can be significant, reaching a factor of 100 times the size of the original request [47]. Although the *Forcepoint anti-spoofing* feature can partially mitigate the attack, its effectiveness depends on a correct firewall configuration [47].

The vulnerability was classified as high severity, with a CVSS score of 7.8, and was initially disclosed on September 27, 2021, with an update on October 4 of the same year [48]. No viable alternative solutions were identified to mitigate the problem without a version update [47]. The corrected versions are 6.5.12, 6.8.7, and 6.10.1 of the NGFW Engine [48].

3.3.2 Fortinet's CVE-2022-27491

The CVE-2022-27491 vulnerability affects the Fortinet FortiOS in various versions of the IPS *engine*, ranging from 7.201 to 7.214, 7.001 to 7.113, 6.001 to 6.121, 5.001 to 5.258 and prior to 4.086, in addition to FortiOS versions 6.0.0 to 6.0.14, 6.2.0 to 6.2.10, 6.4.0 to 6.4.8, 7.0.0 to 7.0.5 and 7.2.0 [49] [50] [51]. The flaw results from an inadequate verification of the origin of communication channels, which allows a remote and unauthenticated attacker to send "blocked page"HTML data to a victim through manipulated TCP requests, generating unexpected traffic and potentially overloading the victim's system [52] [51].

This behavior only occurs when a firewall policy is operating in flow-based inspection mode (default mode) with at least one security profile enabled [51]. The flaw is related to the FG-IR-22-073 bulletin and the VIGILANCE-VUL-39198 alert, being exploitable by attackers with advanced skills, especially in the context of an amplified reflection attack using a *middlebox* [53]. Affected products include FortiGate, FortiGate Virtual Appliance, and FortiOS [53].

The severity of the vulnerability was rated differently among sources. The NVD assigned a CVSS score of 7.5, classified as high, with a CVSS vector: 3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H [49]. The score provided by Fortinet's CNA was 6.8, considered medium, with a CVSS vector: 3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:N/I:N/A:H [49].

The definitive fix is available in FortiOS versions 6.2.11, 6.4.9, 7.0.6, 7.2.1 and above, as well as IPS engines 4.086, 5.259, 6.122, 7.114, and 7.215 [52]. As a temporary solution, it is recommended to disable or reconfigure security profiles that generate "blocked page"responses or use proxy-based inspection mode [51].

3.3.3 Palo Alto's CVE-2022-0028

The *vulnerability* CVE-2022-0028 affects the Palo Alto Networks PAN-OS system and allows for reflected and amplified denial of service attacks over TCP, with severity classified as high and a CVSS score of 8.6 [54]. It results from an incorrect configuration of the URL filtering policy, applied to security rules in zones with internet-facing interfaces, which is generally unintentional [55]. In these circumstances, an attacker can induce the firewall to respond to malicious packets, generating traffic against third parties and making the attack appear to originate from Palo Alto's PA (physical), VM (virtual), or CN (container) series devices [56] [57].

According to Palo Alto, although the flaw does not directly compromise the confidentiality, integrity, or availability of the affected devices, it can be exploited to mask the origin of malicious traffic [58]. Additionally, researchers warn that this behavior can be leveraged by *botnets* to conduct large-scale attacks and even leak information about the filtering policy, which could facilitate *phishing* campaigns [59].

The exploitation of this flaw requires the activation of URL filtering with blocked categories in external-facing zones and the absence of adequate protections, such as mitigation against packet-based attacks or SYN floods with cookies [56]. The vulnerability in Palo Alto's NGFW was initially discovered by the Shadowserver Foundation and its exploitation was confirmed by a service provider, who observed reflected attacks originating from multiple manufacturers, including Palo Alto Networks [59] [56].

The flaw was mapped to the CWE-406 category, which describes insufficient control of network message volume (traffic amplification) [60] [55], and was included in the CISA known vulnerabilities catalog on August 22, 2022 [61]. According to the CTIR Gov, cybercriminals tend to exploit vulnerabilities within minutes of their disclosure, as happened in this case [57]. Palo Alto published fixes for all affected PAN-OS versions, including versions 8.1.23-h1, 9.0.16-h3, 9.1.14-h4, 10.0.11-h1, 10.1.6-h6, and 10.2.2-h2, with updates released up to the week of August 15, 2022 [59].

The recommendations for mitigation include removing the vulnerable configurations or activating protections, such as "zone protection profiles,"packet filtering, or the use of SYN cookies, with the simultaneous application of all measures being unnecessary [57]. The CISA advised administrators to apply the patches as soon as possible, requiring the fix for federal systems by September 12, 2022 [61]. The discovery of the vulnerability was attributed to the company Excellium-Services S.A., and the updates also cover clients of the Cloud NGFW and Prisma Access solutions, which are already protected without the need for additional action [61, 55].

3.4 Forum discussions

There have been several discussions in forums about the Amplified Reflection Attack over TCP, which exploits *middleboxes*. Two examples of these discussions can be seen in forums from vendors like Check Point [62] and Cisco [63].

In the Check Point forum, a participant asked how to react to these attacks, raising concerns about DDoS (*Distributed Denial of Service*) protection on Check Point devices [62]. In response, the company provided useful resources, including links to specific products and a step-by-step guide on how to configure defenses against this type of attack [62]. In the developments, questions arose about the relationship between CVE-2022-0778 and TCP *middlebox* reflection attacks, seeking to understand how these attacks can be orchestrated through the exploitation of vulnerabilities in *middleboxes* [62]. Check Point explained how these devices can be used to amplify TCP traffic in DDoS attacks, providing guidance on best configuration practices, such as reviewing *anti-spoofing* options and implementing strategies to mitigate DoS [62].

In the Cisco forum, the conversation started with a request for help to protect against DoS attacks, specifically using the TCP *Middlebox Reflection* technique, with Bock et. al [3] being mentioned who detailed the vulnerability and asked if it would be possible to configure protection against this type of attack in Cisco firewalls or other Cisco solutions, such as email or web gateways [63]. The community's response clarified that this type of attack exploits vulnerable firewalls and content filtering systems to reflect and amplify TCP traffic, resulting in powerful DDoS attacks against the target [63]. Subsequently, more specific guidance was requested on how to create firewall rules to protect against this attack, questioning whether the configuration proposed in the article [3] would be sufficient in Cisco firewalls and requesting practical instructions on how to implement the protection [63].

3.5 Chapter Summary

In this chapter, related studies on the amplified reflection attack over TCP exploiting *middleboxes* were presented, as well as its repercussion in the specialized media and the technical community. Documented vulnerabilities in databases such as CVE were discussed, with emphasis on specific flaws identified in products from Forcepoint (CVE-2021-41530), Fortinet (CVE-2022-27491), and Palo Alto Networks (CVE-2022-0028), in addition to the solutions proposed by their respective manufacturers. Discussions in forums and online platforms that demonstrate the community's growing concern with this type of attack were also addressed. In the next chapter, the methodology used in the experiments will

be described, detailing the configured test environment and the source code used. Then, in the following chapter, the methodology used in the scan carried out on the Brazilian internet will be presented, with the aim of identifying vulnerable *middleboxes* by sending custom TCP packets.

Capítulo 4

Laboratory Tests

The first set of results obtained concerns bench tests carried out under controlled conditions to observe the attack in different configurations. Aiming to ensure not only repeatability but also the consistency of the results with previous works, a code available on the internet was used for this purpose. Below, we have a detailed description of this code, as well as a description of the testing procedures used, along with the results obtained and their discussion.

4.1 Description of the attack code

The code used to carry out the attacks was obtained from a public repository available on GitHub, authored by the user identified as `moloch54`. The project in question is titled *"Ddos-TCP-Middlebox-Reflection-Attack"* [64] and has the file `mra.py` intended for the execution of the attack. After a more in-depth investigation, it was possible to identify that the author of the code is named Sébastien Meniere, residing in Nancy, in the Grand Est region, in France, according to information found on his professional profile on LinkedIn.

The code proved to be extremely useful for conducting the laboratory experiments, allowing the practical implementation of the concepts addressed in the scientific article *"Weaponizing Middleboxes for TCP Reflected Amplification"* by Bock et al. [3]. Through this algorithm, it was possible to experimentally replicate the *Middlebox reflection* (b) attack technique Figura 3.1 described by the authors, facilitating the understanding and validation of the amplified reflection mechanisms using *middleboxes* over the TCP protocol, where the source IP address is forged with that of the victim, so that the responses generated by the *middleboxes* reach the target directly.

To understand how the attack works, it is necessary to review the basic process of TCP connection establishment, known as the TCP Handshake, which occurs in three steps, as described in detail in Section 2.2:

1. The client (SRC) sends a SYN packet to initiate the connection;
2. The server (DEST) responds with a SYN, ACK packet, acknowledging the request;
3. The client sends an ACK packet, finalizing the connection establishment.

The attack exploits the fact that, in networks with intermediate devices such as firewalls or middleboxes, packets can follow different paths. Suppose the SYN packet (spoofed with `SRC=Victim`, `DST=Server possibly forbidden by the middlebox`) is intercepted by a *middlebox*, but the real server's SYN, ACK response follows another route, not being observed by this *middlebox*. In this case, the *middlebox* sees the start of the connection (the SYN), but not the subsequent response (SYN, ACK), creating a state inconsistency.

The trick used by the attacker consists of sending, after the forged `SYN`, a second `ACK` packet (also with the source IP spoofed as being from the victim). The *middlebox*, not having registered the SYN, ACK, interprets this ACK as unexpected or invalid, and can generate automatic responses such as RST packets or, in specific cases, multiple blocking or warning messages, thus amplifying the traffic against the victim.

Additionally, the attack can be refined with the use of packets containing data. For example, after sending the SYN packet, the attacker can send an ACK+PSH packet with a payload, such as an HTTP request:

- Sending a SYN packet with a forged source IP address (of the victim) and destination to domains with high potential for being blocked by *middleboxes*: `SRC=Victim`, `DST=Pornhub|Youporn|Bittorrent...`;
- Then, sending an ACK+PSH packet with a payload: HTTP GET request, also with `SRC=Victim`.

These packets, when processed by content filtering *middleboxes*, can trigger multiple automatic responses, reinforcing the amplification effect.

The choice to use this publicly available code was motivated by its clear structure, aligned with the model and methodology described in the study by Bock et al. [3]. This standardization ensured that the implementation was in accordance with the attack specifications, minimizing variables and enabling a more precise analysis of the results. Opting for an already consolidated and documented tool also allowed focusing on the experimental and analytical aspects of the study, avoiding the need to develop code from

scratch and reducing the risk of introducing inconsistencies that could compromise the validity of the experiments.

4.1.1 Code architecture and functioning

The `mra.py` script, available in the `Ddos-TCP-Middlebox-Reflection-Attack` repository [64], directly uses IPv4 addresses of sites potentially subject to blocking by *middleboxes*, instead of working with domain names. These IP addresses can be easily obtained through tools such as the `nslookup` command, available in most operating systems. For example, executing the command `nslookup facebook.com`, as illustrated in Figura 4.1, returns the IPv4 address currently associated with the `facebook.com` domain, which, at the time of this analysis, corresponds to `157.240.13.35`. It is noted that this address can vary over time, as updates in the domain name system (DNS) occur.

Figura 4.1: Screenshot of the execution of the command `nslookup facebook.com`.

This choice is due to the way the packets are built in the attack, as the fields of the TCP/IP protocol are directly manipulated, the domain name is not used in the construction of the packets. By using the IP address directly, the need for name resolution during the sending of packets is avoided, which simplifies the process and allows for greater control over the content of the generated traffic, ensuring that it follows the necessary format to trigger the specific behavior of the *middleboxes*.

Based on this approach, a list of *websites* considered forbidden by *middleboxes* was defined, as shown in the following code snippet. The IPs listed correspond, respectively, to the domains `youporn.com`, `facebook.com`, `pornhub.com`, and `bittorrent.com`:

Snippet 4.1: List of IP addresses associated with domains frequently blocked by middleboxes for triggering the attack.

```
forbidden_websites = ["66.254.114.79", "157.240.13.35", "
    66.254.114.41", "98.143.146.7"]
```

The *generate* function forges TCP packets with a forged source IP, representing the beginning of a TCP connection:

Snippet 4.2: Creation of a TCP SYN packet with a forged IP.

```
ip = IP(src=target_ip, dst=dst_ip, ttl=255)
tcp = TCP(sport=sport, dport=80, seq=seq, flags="S")
packet = Ether() / ip / tcp
```

Next, ACK + PSH packets are constructed to simulate data transmission, with the PA flag activated and an HTTP payload. The acronym PA refers to the combination of the PSH and ACK flags in the TCP protocol. The PSH flag requests that the data be passed immediately to the application, without waiting for other packets, while the ACK flag indicates that the packet is acknowledging the reception of data. This combination is used to simulate the continuous sending of data and confirm its reception at the same time. The following code builds the TCP packet with these flags activated:

Snippet 4.3: Creation of a TCP packet with ACK and PSH flags and an HTTP GET payload.

```
payload = 'GET / HTTP/1.1\r\nHost: ' + dst_ip + '\r\n\r\n'
tcp = TCP(sport=sport, dport=80, ack=RandShort(), seq=seq +
   1, flags="PA")
packet = Ether() / ip / tcp / payload
```

Each thread responsible for generating packets saves the forged packets in distinct `.pcap` files. This approach is adopted to ensure that the packets generated by each thread are stored separately, facilitating the later manipulation and analysis of the data. The name of each file is dynamically generated based on the thread index (i), ensuring that there is no file overwriting and that each set of packets is recorded individually. The command used to save the packets is the following:

Snippet 4.4: Saving the generated packets to a thread-specific .pcap file.

```
wrpcap(f"{i}.pcap", uni_list)
```

In this case, the `wrpcap` function of the `scapy` package is responsible for writing the packets present in the `uni_list` to the `i.pcap` file. This file contains the packets generated by the corresponding thread, which will later be merged for the execution of the attack. The use of distinct files facilitates the creation of multiple packet instances in parallel, in addition to allowing control over the number of packets generated by each thread.

These files are subsequently joined with `mergecap`, resulting in a single file named `11.pcap`:

Snippet 4.5: Command to merge .pcap files into a single file.

```
cmd = f"mergecap -a {fi} -w 11.pcap"
os.system(cmd)
```

Finally, the continuous sending of the generated packets is performed using the `tcpreplay` command. This utility allows the reproduction of the packets saved in the `.pcap` file on the network interface defined in Scapy. The `-i` option is used to specify the network interface to be used, which is obtained from the Scapy configuration (`conf.iface`). The command also includes other important options for controlling the reproduction of packets, such as the transmission rate (`-mbps=4`), the number of repetitions (`-loop=9999999999`), and the duration of the attack (`-duration={duration}`), in addition to the display of statistics every 10 seconds (`-stats=10`).

The complete command used for the execution of `tcpreplay` is the following:

Snippet 4.6: Command for continuous packet reproduction with tcpreplay.

```
cmd = f"sudo tcpreplay -i {conf.iface} --preload-pcap --loop
   =9999999999 --mbps=4 --duration={duration} --stats=10 11.
   pcap"
os.system(cmd)
```

This command is executed with superuser privileges (`sudo`) to ensure the necessary access to the network interface. The use of `-preload-pcap` ensures that the entire `11.pcap` file is loaded into memory before starting the reproduction, optimizing the process. The configuration of `-loop=9999999999` makes `tcpreplay` send the packets continuously, simulating an overload on the target during the period specified by the `-duration` option. This approach allows for the execution of a long-duration attack, while the periodic statistics provide information on the attack's performance in real time.

4.2 Code execution

To execute the code related to the amplified reflection attack over TCP, based on the instructions provided in the `README.md` file of the repository, it is necessary to follow the steps described below. Before starting the execution, one must ensure that all dependencies are properly installed and configured. The necessary tools are:

- **tcpreplay**: Used to reproduce the generated packets on a network interface.
- **mergecap**: Used to merge the packets generated by multiple threads into a single `.pcap` file.
- **scapy**: Python library used for the construction and manipulation of TCP packets.

To install the dependencies on the system, run the following commands (on Debian/Ubuntu-based systems):

Snippet 4.7: Installation of the necessary dependencies for attack execution.

```
sudo apt-get install tcpreplay mergecap python3-scapy
```

After ensuring that the dependencies are installed, the code can be executed directly from the terminal. The command to start the attack is the following:

Snippet 4.8: Command to start the attack.

```
sudo python3 mra.py <time_in_seconds> <target_IP>
```

Here, the `<time_in_seconds>` parameter defines how long the attack will be executed, in seconds, and `<target_IP>` is the IP address of the target that one wishes to overload with the amplified traffic.

The script begins by generating forged TCP packets. To do this, it sends a SYN packet (start of the TCP handshake) to the destination site (which is filtered by a *middlebox*). Then, an ACK + PSH packet, with an HTTP payload, is sent to the same destination. This packet causes the *middlebox* to respond with an RST, or even with an error page. The generated traffic is then stored in `.pcap` files and repeated indefinitely using *tcpreplay*, sending the packets to the network interface specified in the code.

To perform the attack, the command would be similar to the following example, where the attack will be executed for 300 seconds against the fictitious IP 123.4.5.6:

Snippet 4.9: Example of attack execution.

```
sudo python3 mra.py 300 123.4.5.6
```

An example of the command execution can be seen in Figure 4.2. It is important to remember that the use of this code to carry out attacks without authorization is illegal, unethical, and should not be used for malicious purposes. The intention is only educational and for performing tests in controlled environments with explicit permission for security analysis.

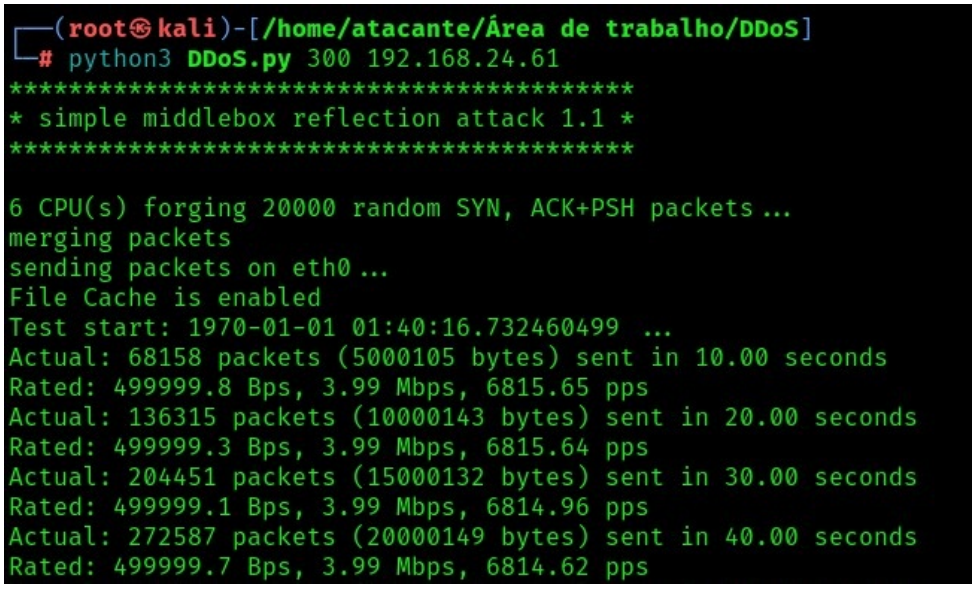

Figura 4.2: Screenshot of the execution of the `mra.py` attack script.

4.2.1 Communication with the code author

After identifying that the author of the repository and the code available on GitHub was the user `moloch45`, corresponding to Sébastien Meniere, an attempt was made to contact him via LinkedIn. The communication took place in French, on which occasion he was asked if he was, in fact, the owner of the mentioned account, which was promptly confirmed by him. Subsequently, he was informed that the referred code was being used as part of a final paper for a Computer Science course, developed in Brazil, and that there was an intention to properly credit the authorship of the work.

After that, the author was thanked for the code provided, highlighting its usefulness for the paper, which deals with amplified reflection attacks over TCP exploiting *middleboxes*. He was also informed that a laboratory environment had been set up, consisting of a target machine, an attacking machine, and a pfSense firewall to conduct experiments, as demonstrated in Figura 4.3.

In response, Sébastien showed surprise and interest upon learning that the code was being tested in a laboratory. He asked if it had worked correctly, as he had never performed this type of test, which reinforces the relevance of the experiment conducted in the laboratory described in this paper.

After that, the author was asked how he would like to be cited, and there was also a mention of future plans to test the attack on other firewalls, both physical and virtual, such as those from the Palo Alto and Forcepoint brands. Sébastien responded positively, authorizing the use of his name and suggesting the inclusion of the link to his GitHub

profile as a reference, demonstrating enthusiasm upon learning of the project's progress. Subsequently, he was presented with the results of the tests initially performed on pfSense, followed by the tests on FortiGate, from Fortinet, which were also successful. He asked about the bandwidth obtained in the DDoS tests and showed interest in reading the paper once it was finalized.

4.3 VMware Workstation Pro Virtualizer

For the virtualization of the target, attacker, and pfSense and FortiGate firewall virtual machines, VMware Workstation Pro 17 for Personal Use (version 17.6.3-24583834) was used [65], available for free for personal use. The software can be downloaded from the official VMware website (`https://www.vmware.com/products/desktop-hypervisor/workstation-and-fusion`), and it is necessary to register on the Broadcom portal (`https://support.broadcom.com`).

VMware Workstation is a widely recognized desktop hypervisor for Windows and Linux systems, allowing for the creation, running, and use of virtual machines with various operating systems, such as Windows 11 and Linux distributions, without the need to restart the computer. It is a robust and versatile platform, ideal for software development, testing, and simulations in isolated and secure virtual environments [65].

4.4 Laboratory configuration

The laboratory environment was composed of three distinct environments, all using the same topology illustrated in Figura 4.3, with the same configured IP addresses. Each environment had three main elements: a target, an attacker, and a *middlebox*, represented by a firewall. In the first scenario, the firewall was implemented using pfSense with the pfBlockerNG add-on. In the second scenario, the firewall consisted of the combination of pfSense with the Squid and SquidGuard software. Finally, in the third scenario, the firewall used was FortiGate.

It is worth noting that the firewalls were intentionally configured to simulate both the typical behavior of corporate environments and incorrect configurations, precisely to highlight how certain configuration choices can make such devices vulnerable to amplified reflection attacks over TCP, even though they are solutions widely used in business environments.

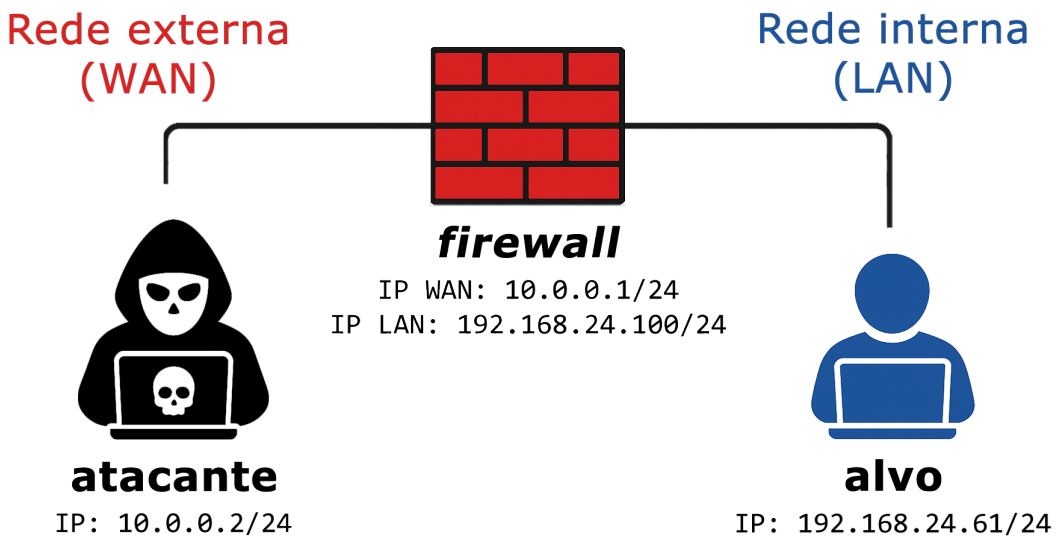

Figura 4.3: Laboratory network topology used in the three experimental attack environments.

4.4.1 Target machine

The machine used as a target in the attacks of this experiment was an instance of the Ubuntu Linux 22.04.5 LTS (Jammy) operating system. It is a popular open-source desktop operating system around the world and has good long-term support [66]. The following are its virtual hardware specifications:

- **RAM Memory:** 4096 MB
- **Processor:** 2 vCPUs
- **Storage:** 25 GB
- **Network adapters:** 1000 Mb/s (Intel Gigabit Internet 82545EM)

The VM is equipped with two network interfaces. The first is connected to the LAN segment, identified as `LAN` in VMware, simulating the machine's presence in a protected local network, located behind the firewall. The second interface is configured in NAT mode and is used exclusively for internet access through the connection with the host computer. This configuration allows for the download of updates, packages, and the performance of necessary external communications.

The network configuration of the target machine was defined as follows:

- **Hostname:** `Ubuntu`
- **LAN IP address:** `192.168.24.61/24` (ens34)

- **INTERNET IP address:** IP assigned via DHCP (ens37)
- **Default Gateway:** `192.168.24.100`
- **DNS Server:** `192.168.24.100`

4.4.2 Attacker machine

The machine used as an attacker in the experiments was based on the Kali Linux operating system, in version 2025.1 (kali-rolling). Its virtual hardware specifications are as follows:

- **RAM Memory:** 8096 MB
- **Processor:** 6 vCPUs
- **Storage:** 32 GB
- **Network adapters:** 1000 Mb/s (Intel Gigabit Internet 82545EM)

The attacker VM has two network interfaces. The first is connected to the LAN segment, identified as `WAN` in VMware, simulating that the machine is located on the internet, that is, outside the protected environment. This interface allows the VM to play the role of an external agent trying to attack the target machine within the internal network. The second interface is configured in NAT mode and is used exclusively for internet access through the host computer's connection, allowing for the download of updates, packages, and other external communications necessary to prepare the attack environment.

The network configuration of the attacker machine was defined as follows:

- **Hostname:** `kali`
- **WAN IP address:** `10.0.0.2/24` (eth0)
- **INTERNET IP address:** IP assigned via DHCP (eth1)
- **Default Gateway:** `10.0.0.1`

4.4.3 pfSense Firewall

The first firewall used in the experiments was the *pfSense Community Edition* (CE), in version `2.7.2-RELEASE` [67]. It is an open-source firewall software distribution, based on FreeBSD, that can be installed on a physical computer or in a virtual machine to make up a dedicated firewall in a network [68].

The virtual hardware specifications assigned to the machine are presented below:

- **RAM Memory:** 2048 MB
- **Processor:** 2 vCPUs
- **Storage:** 20 GB
- **Network adapters:** 1000 Mb/s (Intel Gigabit Internet 82545EM)

The pfSense VM was configured with three network interfaces:

- **em0:** Connected to the LAN segment, identified in VMware as `WAN`, simulating the exit of internal machines towards the internet;
- **em1:** Connected to another LAN segment, identified as `LAN`, representing the internal access interface to the firewall;
- **em3:** Configured in NAT mode and is used exclusively to allow internet access via the host computer's connection, enabling the download of updates, packages, and other external communications necessary for preparing the attack environment.

The network configuration of the firewall virtual machine was defined as follows:

- **Hostname:** `pfSense.home.arpa`
- **WAN IP address (em0):** `10.0.0.1/24`;
- **LAN IP address (em1):** `192.168.24.100/24`;
- **INTERNET IP address (em2):** assigned via DHCP.

Outbound NAT was configured to redirect traffic from the LAN network to the WAN interface, simulating the exit of the internal network to the external one through the firewall, as illustrated in Figure 4.4.

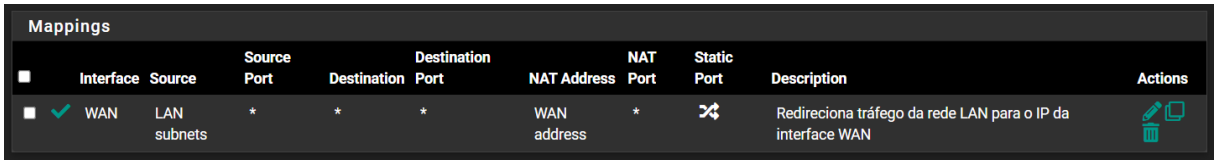

Figura 4.4: NAT rules present in pfSense (Firewall → NAT).

The firewall rules configured for the WAN interface are presented in Figure 4.5, and include:

- A rule that allows the reception of ICMP (ping) packets originating from external machines to internal machines on the LAN;

- A rule that allows `ping` from external machines directly to the IP address of the pfSense WAN interface;
- A rule that allows external access to the target's `Web` server on port 80/TCP, simulating its public exposure;
- A rule that allows machines on the internal network to freely access all sites, being subsequently restricted by the `pfBlockerNG` and `squidGuard` filtering solutions, responsible for blocking domains such as `youporn.com`, `facebook.com`, `pornhub.com`, and `bittorrent.com` and processed before this rule.

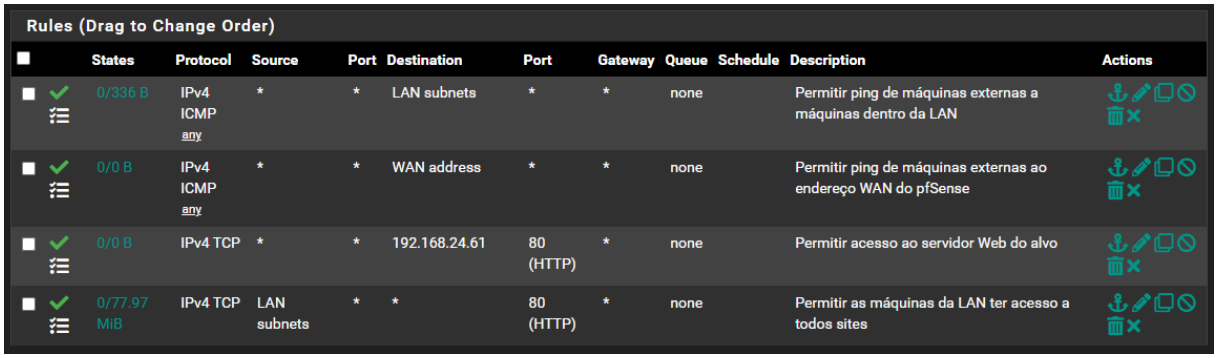

Figura 4.5: WAN interface rules present in pfSense (Firewall → Rules).

For the LAN interface, the rules are configured as shown in Figure 4.6, containing by default:

- The `anti-lockout rule`, which prevents access to the pfSense web interface from being blocked;
- The `default allow LAN to any` rule, allowing unrestricted outbound traffic from the LAN;
- The `default allow LAN IPv6 to any` rule, with equivalent behavior for IPv6 traffic.

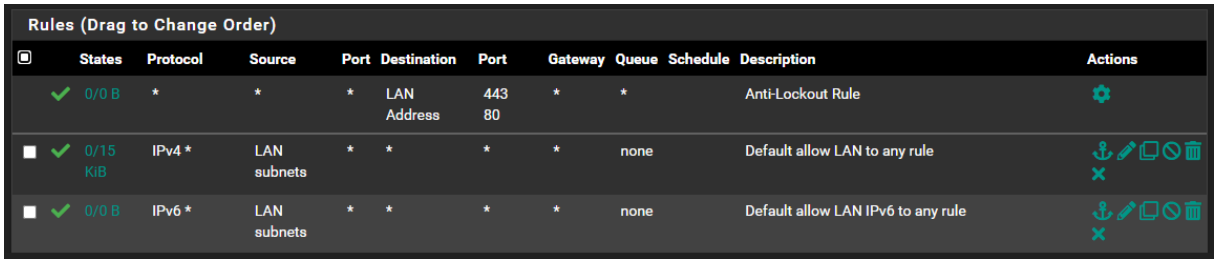

Figura 4.6: LAN interface rules present in pfSense (Firewall → Rules).

pfSense + pfBlockerNG Firewall

`pfBlockerNG` [69] is an additional package available in *pfSense* that acts as a blocker of malicious domains and IPs, offering advanced network access control functionalities through the use of external lists. It allows administrators to block entire categories of sites, such as pornography, social media, gambling, among others. One of its main functions is the use of DNSBL (*DNS Blackhole List*), which intercepts DNS queries and responds with local addresses, preventing access to certain domains. In addition, `pfBlockerNG` also works with lists of malicious IPs, enabling the creation of firewall rules to block traffic associated with these addresses [69].

To allow the blocking of forbidden sites, an explicit `DROP` rule for packets originating from the internal network destined for IP addresses associated with forbidden sites, such as `youporn.com`, `facebook.com`, `pornhub.com`, and `bittorrent.com`, was added to the top of the LAN interface rules, as presented in Figure 4.6. These addresses were obtained and maintained through IP lists configured directly in `pfBlockerNG`, as illustrated in Figure 4.7.

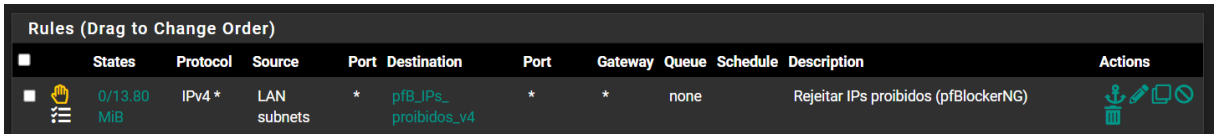

Figura 4.7: Rule for blocking packets destined for forbidden IPs added to the LAN interface.

The package also allows the display of a block page to the user whenever they try to access a forbidden domain or IP, as demonstrated in Figure 4.8. This page is hosted locally on the *pfSense* itself and serves to inform that the content has been blocked by network security policies. However, it is important to note that `pfBlockerNG` does not perform a real redirection of the request: it only responds with a local IP without changing the URL displayed in the browser. This means that, despite displaying the block page, the original request is not actually redirected.

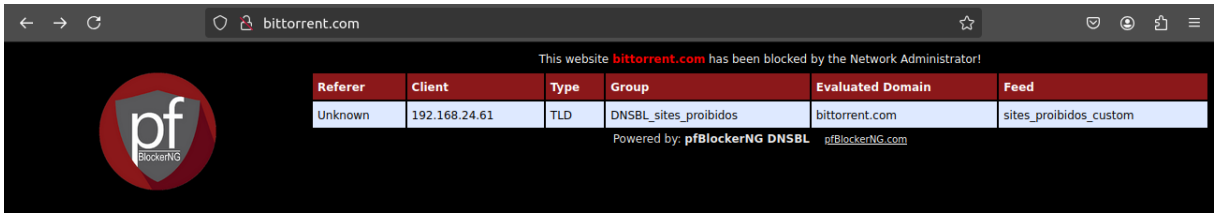

Figura 4.8: Block page of the pfSense pfBlockerNG package.

This behavior has important implications in the context of TCP reflection attacks exploiting *middleboxes*. As the *middlebox* (in this case, *pfSense* with `pfBlockerNG`) does not perform redirection or active re-sending of content, it is not capable of reflecting the block page to an external target. Thus, `pfBlockerNG` does not prove useful for the execution of amplification attacks based on this type of response, since the entire blocking process occurs locally and does not interact directly with the final destination of the traffic.

pfSense + Squid + SquidGuard Firewall

Squid is a web caching *proxy* widely used in corporate, educational, and experimental environments. It acts as an intermediary between clients (like browsers) and internet servers, caching frequently accessed web pages. This allows for significant bandwidth savings and an improvement in response time for end users. In addition to its caching functionalities, *Squid* also offers advanced features for access control, content filtering, user authentication, and detailed request logging. It is compatible with various protocols, such as HTTP, HTTPS, and FTP, and can be run on various platforms, including Unix/Linux and Windows operating systems. *Squid* is distributed under the GNU GPL license, being a free, robust, and highly configurable solution.

Integrated with *Squid*, *SquidGuard* is a URL redirector that complements the *proxy*'s functionalities by offering content filtering mechanisms based on domain lists, categories, or regular expressions. With this feature, it is possible to block access to unwanted or potentially dangerous sites and redirect the user to a custom block page, as illustrated in Figure 4.9. To enable the blocking of forbidden sites, a list of *target categories* was created in the *SquidGuard Proxy Filter* service, present in pfSense, as demonstrated in Figure 4.10.

SquidGuard was the main component used in this experiment precisely because it allows for the real redirection of the request, changing the URL displayed in the browser and sending the block page directly to the user who tries to access restricted content.

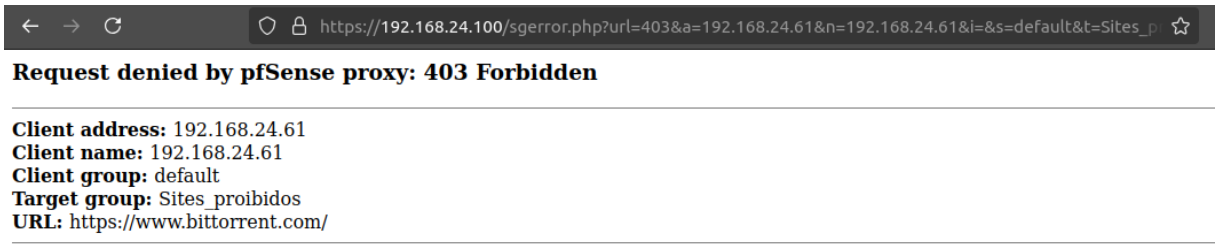

Figura 4.9: Block page of the pfSense SquidGuard package.

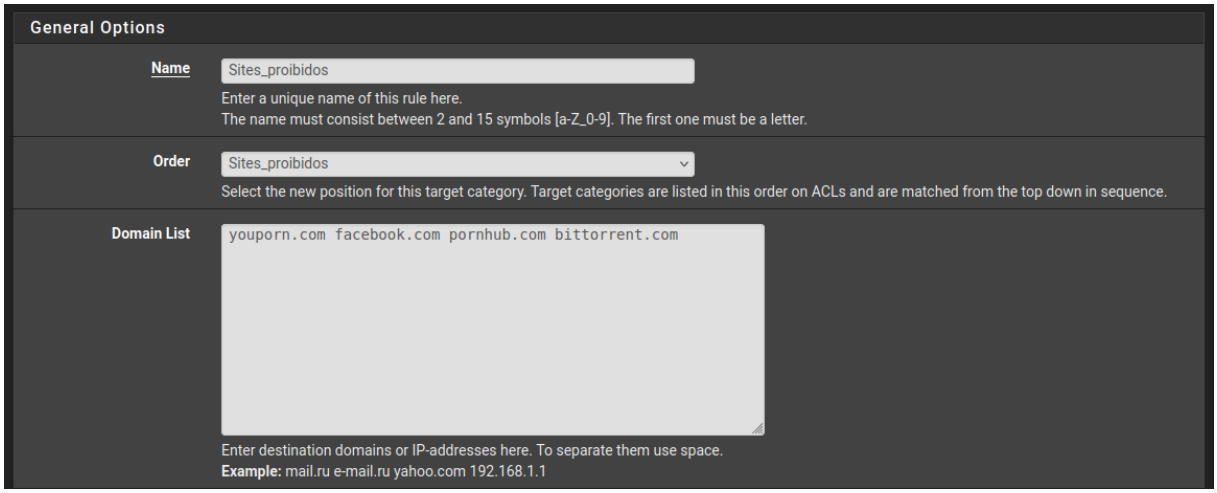

Figura 4.10: Target categories configured in SquidGuard for blocking forbidden sites (Services → SquidGuard Proxy Filter → Target categories).

4.4.4 FortiGate Firewall

The second firewall used in the experiments was the NGFW FortiGate-VM64, version 7.2.0 (build 1157, 220331 - GA.F) [70]. The choice of this version was due to the fact that it is on the list of vulnerable versions, as identified by the vulnerability CVE-2022-27491 [49] [50] [51]. FortiGate NGFWs offer advanced protection for users and data, combining security functionalities with high performance through Fortinet's dedicated processors and is a consolidated and widely adopted commercial solution in the market [71].

Large institutions, such as the University of Brasília (UnB), actively adopt this solution to ensure the security of their networks. A practical example of this use can be observed in Figura 4.11, which shows the block page displayed by FortiGate when a user tries to access a forbidden site. This test was carried out within the UnB itself, confirming the effective use of the system by the institution. In addition, the same page can be viewed on the target machine when doing the same in the laboratory.

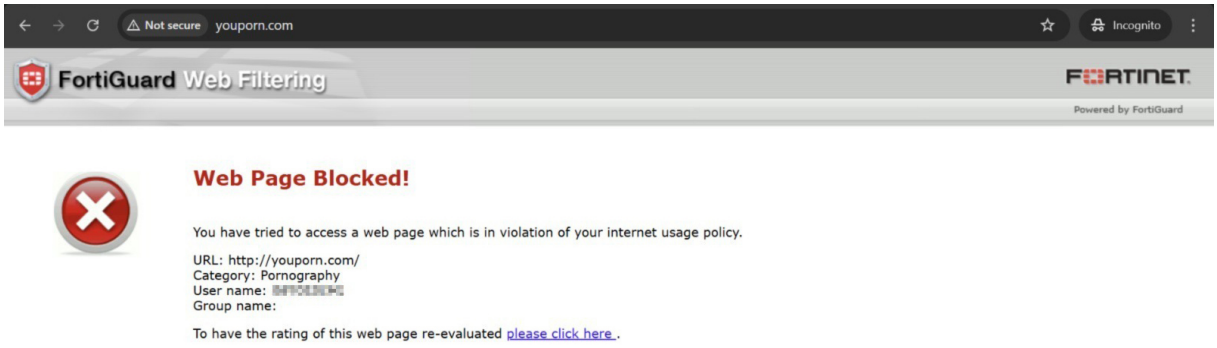

Figura 4.11: FortiGate block page.

The specifications of the virtual environment used in the tests are presented below:

- **RAM Memory:** 2048 MB
- **Processor:** 1 vCPU
- **Storage:** Not specified
- **Network adapter:** 10000 Mb/s (Intel Gigabit Internet 82545EM)

Adopted as a substitute for pfSense, the FortiGate firewall was configured with three network interfaces, replicating the topology and function assigned in the previous configuration. The network configuration of the firewall was defined as follows:

- **Hostname:** `FortiGate-VM64-KVM`
- **WAN IP address:** `10.0.0.1/24` (port1)
- **LAN IP address:** `192.168.24.100/24` (port2)
- **INTERNET IP address:** IP assigned via DHCP (port3)

Given the finding that neither pfBlockerNG nor Squid with SquidGuard were able to reflect block pages to the victim during the tests, it was necessary to look for a more suitable alternative for the objectives of the experiment. In this context, it was decided to use FortiGate, a next-generation firewall solution developed by Fortinet, which proved to be more effective for the proposed scenario.

FortiGate was chosen for offering a content filtering mechanism more integrated into the network flow, with the real ability to intercept connections and respond directly with custom block pages, which are effectively sent to the client as complete HTTP responses. Unlike the previous solutions, FortiGate does not depend on DNS manipulation

or explicit proxies to display block messages; it acts directly on the traffic, with deep packet inspection (DPI) and immediate response, which increases the chance of the page being reflected to the forged destination in amplification attacks.

Furthermore, FortiGate allows for more granular control of security policies and responses, with specific tools for customizing block messages, session analysis, and handling behavior-based connections. These resources make the solution more suitable for advanced security tests and for analyzing how *middleboxes* interact with forged traffic.

Therefore, the use of FortiGate represented an evolution in the experimental methodology, offering greater control and visibility over traffic, in addition to a higher potential to generate reflected responses useful for the study of TCP amplification attacks based on *middleboxes*.

The *FortiGate* was configured with firewall rules similar to those created in *pfSense*, as illustrated in Figures 4.12 and 4.13, which correspond, respectively, to the following policies:

- A rule that allows the reception of ICMP (ping) packets originating from external machines (WAN network) to internal machines on the LAN network;
- A rule that allows outbound traffic from internal machines (LAN network) to the external network (WAN network) through the *FortiGate*.

For the blocking of forbidden sites, a *Web Filter* profile was created, which was later applied to the LAN access rules, as shown in Figure 4.14.

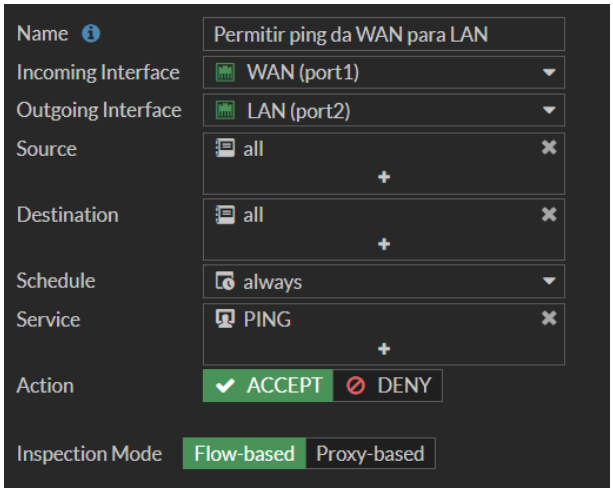

Figura 4.12: Firewall policy in FortiGate allowing `ICMP` from the WAN network to the LAN.

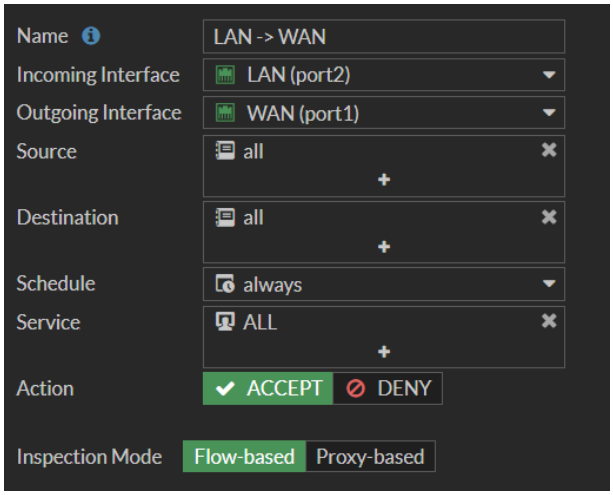

Figura 4.13: Firewall policy in FortiGate allowing access from the LAN network to the WAN.

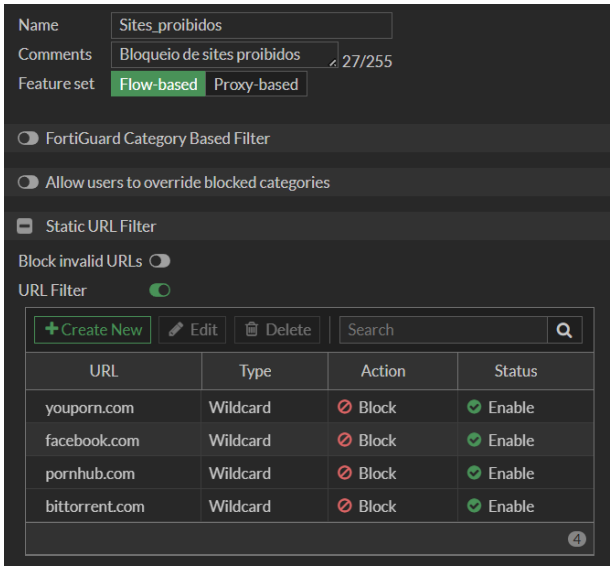

Figura 4.14: Configuration of the *Web Filter* in FortiGate for blocking forbidden sites (Security Profiles → Webfilter).

4.5 Chapter Summary

In this chapter, the configuration of the experimental environment used for the analysis of amplified reflection attacks over TCP exploiting *middleboxes* was detailed, covering both the source code structure and the virtual and network elements of the laboratory. Initially, the architecture and functioning of the attack code used were presented, including

execution guidelines and information obtained through direct communication with its author. Next, the testing environment built on VMware Workstation Pro was described, composed of virtual machines representing the attacker, the target, and the *middlebox* devices, specifically the pfSense and FortiGate firewalls, with their respective configurations. This structure allowed for the controlled simulation of the attack and the observation of its behavior when faced with different intermediate devices. In the next chapter, the methodology of a scan conducted on the Brazilian internet will be described with the objective of identifying vulnerable *middleboxes*, by sending customized TCP packets, aiming to evaluate the presence and distribution of this type of vulnerability in a real scenario.

Capítulo 5

Scanning for Vulnerable Middleboxes in Brazil

Following the laboratory tests, a scan was carried out on the IPv4 BR space with the objective of identifying vulnerable middleboxes, by sending customized TCP packets. The objective of this scan was to evaluate the presence and distribution of this type of vulnerability in a real scenario, as well as its susceptibility to exploitation.

5.1 ZMap

ZMap is a fast, stateless, single-packet network *scanner*, designed for internet-wide research. On a computer with a gigabit Ethernet connection, it can scan the entire public IPv4 address space on a single port in less than 45 minutes. It supports operating systems like GNU/Linux, Mac OS, and BSD, offering modules for TCP SYN scans, ICMP, DNS, massive UDP probe sending, and other functions [72].

It is modular and open-source, designed to perform fast scans of the entire Internet; the probe modules are extensible, generate packets, and interpret responses. The output handlers allow the scan results to be redirected to other processes, databases, or custom code [72].

5.1.1 forbidden_scan module

The ‘`forbidden_scan`‘ module [73] `https://github.com/Kkevsterrr/zmap/blob/master/src/probe_modules/module_forbidden_scan.c` of ZMap is not part of the tool's official repository; it was created by Kevin Bock and his team as part of the research presented in the article "*Weaponizing Middleboxes for TCP Reflected Amplification*"[3].

This module was specifically developed to detect *middleboxes* that inspect the content of TCP packets still during the connection establishment process.

This module performs TCP scans with a payload in SYN packets, sending, for example, an HTTP request (‘`GET / HTTP/1.1`‘) with the ‘`Host: exemplo.com`‘ header, embedded directly in the SYN packet, which is unusual in legitimate TCP connections. The objective is to verify if intermediate devices (such as firewalls or proxies) respond to this anomalous packet.

The module performs global and per-thread initialization to build the packets properly, using Ethernet, IP, and TCP headers, in addition to the custom payload. During packet generation, it correctly calculates the TCP and IP checksums, and sets fields such as source IP, destination IP, TTL, sequence, and acknowledgment (ACK) numbers.

When receiving responses, the module validates if they correspond to what is expected, analyzing ports, sequence/ACK and response type (such as RST, SYN-ACK or packets with data). The processing function extracts and records relevant fields from the response, such as ports, payload size, flags, among others. The main focus of this module is to identify improper behavior of *middleboxes* that process anomalous packets and respond with data, which can be exploited in the attack studied in this work.

5.2 Scanning methodology

This study employed a methodology that aimed to replicate tests performed by organizations such as ShadowServer [41] (`https://www.shadowserver.org/news/over-18-8-million-ips-vulnerable-to-middlebox-tcp-reflection-ddos-attacks`) and Akamai [4] (`https://www.akamai.com/blog/security/tcp-middlebox-reflection`). Additionally, it sought to validate and deepen the studies presented in the article by Kevin Bock et al. [3].

The work of ShadowServer was crucial in conducting extensive Internet scans, quantifying vulnerable *middlebox* devices to TCP reflection distributed denial of service (DDoS) attacks and identifying millions of susceptible IPs, whose tests served as the basis for the replication. Akamai contributed with the detailed observation and analysis of these attacks in real environments, explaining how firewalls and content filtering systems can be exploited to reflect and amplify TCP traffic, providing a practical context for the replicated tests. The seminal studies by Kevin Bock et al. [3], with their article "*Weaponizing Middleboxes for TCP Reflected Amplification*,"were fundamental in being the first to detail and demonstrate the feasibility of this new DDoS attack vector, showing how *middleboxes* could be exploited to amplify TCP attacks.

In this sense, the approach of the present study consisted of conducting a comprehensive scan of the Brazilian IP address space. During this scan, the main focus was the identification of anomalous response patterns emitted by *middleboxes*. The analysis of these patterns aimed to determine the possibility of exploitation for TCP traffic amplification, characterizing them as potential vectors for the triggering of reflection DDoS attacks.

5.2.1 Collection of Brazilian IP blocks

To obtain the list of IP address blocks assigned to Brazil, the *Country IP Blocks* website (`https://www.countryipblocks.net/acl.php`) was used. This platform allows the generation of lists of IP address blocks by country, being a useful tool for implementing access controls based on geographical location.

The procedure consisted of accessing the site, selecting the country "Brazil", choosing the CIDR format, and finally, generating the list of IP blocks according to the records available in the service's database.

The result obtained presents the blocks in CIDR format, as in the following example:

```
138.97.188.0/22
138.97.192.0/22
138.97.196.0/22
138.97.204.0/22
138.97.208.0/22
138.97.212.0/22
```

Based on this, the list of IP blocks generated in CSV format was used to perform the scan proposed in this work.

5.2.2 Scan execution

The identification of potential TCP reflectors in the Brazilian IP address space was conducted through an active scan, using a customized version of the ZMap tool [72]. Specifically, the ZMap *fork* developed by the user Kkevsterrr (Kevin Bock) (`https://github.com/Kkevsterrr/zmap/tree/4b507fa8df203d0ae4df1f06cce2e1a8814fe546`) was employed, which is associated with the studies of Bock *et al.* [3] on the exploitation of *middleboxes* for the purpose of traffic amplification.

The implementation of the tool began with cloning the repository containing the modified version of ZMap, using the commands:

```
git clone --recursive https://github.com/breakerspace/weaponizing
   -censors
```

```
cd weaponizing-censors/zmap
```

After that, the scope file with the Brazilian IPv4 blocks (`blocos_ipv4_brasil.txt`) was added to this directory and the `scan` directory was created.

The `git clone -recursive` command ensures that the repository's submodules are also correctly cloned. Next, the `zmap` directory is accessed, where the source files to be compiled are located. The `blocos_ipv4_brasil.txt` file, containing the Brazilian IPv4 address blocks, must be inserted into this directory. The `scan` directory is created to store the output files generated by the scan.

Subsequently, the tool was compiled from its source code with the following command:

```
cmake . && make -j4
```

In this command, `cmake .` configures the compilation environment in the current directory, generating the necessary `Makefile` files. The `&&` operator conditions the execution of the subsequent command on the success of the previous one. Then, `make -j4` performs the compilation of the source code, using four parallel processes to optimize the executable's build time.

After successful compilation, the scan was effectively carried out by the following command:

```
sudo src/zmap -i ens37 -M forbidden_scan -p 80 \
    -w "blocos_ipv4_brasil.txt" \
    -f "saddr,len,payloadlen,flags,validation_type" \
    -o "scan/scan_brasil.csv" -O csv
```

The parameters used in this command are detailed below:

- `sudo src/zmap`: Executes ZMap, located in the `src/` directory, with superuser privileges. Such privileges are essential for sending raw packets and for low-level network operations.
- `-i ens37`: Specifies the network interface (`ens37`) to be used for sending the probe packets.
- `-M forbidden_scan`: Selects the customized probe module, named `forbidden_scan`. This module sends a TCP SYN packet that contains, in its *payload*, an HTTP GET request (for the *host* `freedomhouse.org`, by default). The objective is to elicit immediate responses from *middleboxes* that inspect HTTP traffic, aiming to identify those that return blocked pages or other anomalous responses that indicate interception and potential for traffic amplification.

- `-p 80`: Defines the scan destination port as 80, corresponding to the HTTP protocol, which is frequently inspected and manipulated by *middleboxes*.
- `-w "blocos_ipv4_brasil.txt"`: Points to the whitelist file named `blocos_ipv4_brasil.txt`. This file contains the Brazilian IPv4 address blocks that were the scope of the scan.
- `-f "saddr,len,payloadlen,flags,validation_type"`: Specifies the fields to be recorded in the output file for each received response, such as the source IP address of the response (`saddr`), the total length of the response IP packet (`len`), the payload length of the response packet (`payloadlen`), the TCP flags of the response packet (`flags`), and a custom field by the module, `validation_type`. This last field categorizes the way the respondent's TCP layer (or *middlebox*) acknowledged the reception of the TCP SYN probe with payload.
- `-o "scan/scan_brasil.csv"`: Designates the name and path of the output file (`scan/scan_brasil.csv`) where the scan results will be stored.
- `-O csv`: Defines the output file format as CSV, facilitating subsequent data analysis.

The central purpose of this scanning stage was, therefore, to systematically map and catalog *middlebox* devices within the Brazilian IP address space that exhibit response behaviors indicative of potential for exploitation in TCP reflection and amplification attacks.

5.3 Methodology for statistical analysis and generation of scan charts

After performing the scan with ZMap, the collected data was processed with the objective of extracting descriptive statistics and quantitative indicators about the potential reflectors present in the Brazilian IPv4 space.

For this, the `stats.py` script, included in the repository `https://github.com/breakerspace/weaponizing-censors`, was used, executed with the following command:

```
sudo python3 stats.py zmap/scan/scan_brasil.csv 149
```

In this command, the first argument is the path to the CSV file generated by the scan, while the second argument represents the number of bytes transmitted per probe for each IP address, in this case, 149 bytes, a value that corresponds to the estimated size of the TCP SYN packet with an embedded HTTP GET payload used in the `forbidden_scan` module.

The `stats.py` script performs a detailed statistical analysis on the received response packets, including:

- The total count of analyzed packets.
- The total number of unique IP addresses that responded to the scan (IPs that emitted some response).
- The total number of IPs that exhibited amplification behavior (sent more data than received).
- The total bytes sent by amplifying IPs.
- The average amplification rate observed.
- The IP addresses that received the most data.
- The IP addresses that received the most packets.
- The distribution of TCP flags in the response packets.
- The generation of cumulative distribution function (CDF) charts for:
  - Number of packets received per IP.
  - Amount of bytes received per IP.
  - Amplification factors.

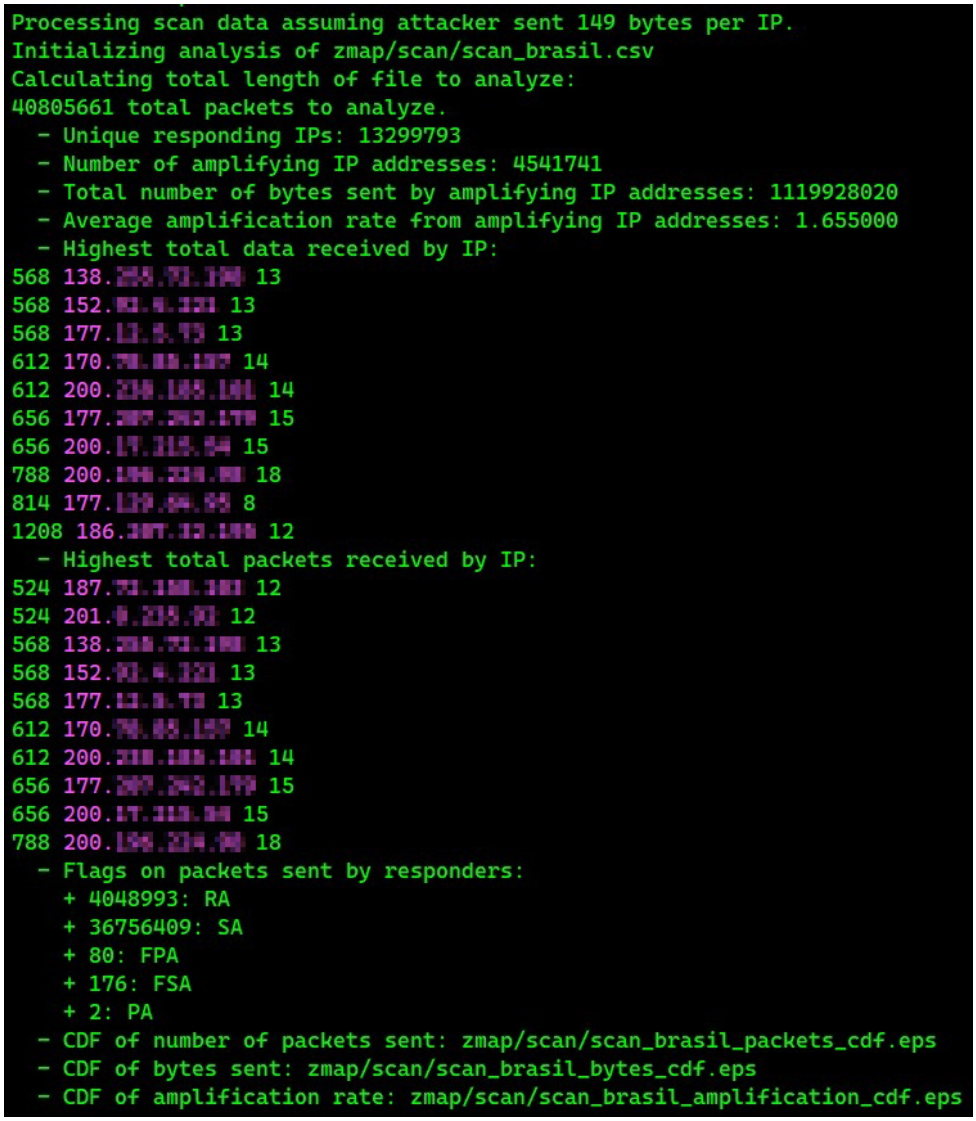

Figura 5.1: Output generated by the execution of the `stats.py` script with the scan result present in the `scan_brasil.csv` file.

The charts generated by the script (`scan_packets_cdf.eps`, `scan_bytes_cdf.eps` and `scan_amplification_cdf.eps`) were later used in Section 6.5, which visually illustrate the distribution of responses and the amplifying behaviors identified. Also included were charts generated through the `matplotlib` (for basic and customizable visualizations) and `seaborn` (for statistical charts with a more refined visual) libraries, which allowed for the clear representation of the data obtained.

5.4 Chapter Summary

In this chapter, the methodology adopted for performing a scan on the Brazilian internet was presented with the objective of identifying *middleboxes* vulnerable to amplified reflection attacks over TCP. Initially, the use of the ZMap tool was described, with emphasis on the `forbidden_scan` module, specially developed to detect this type of vulnerability. Next, the steps for collecting the IP address blocks belonging to Brazil were explained, as well as the procedures used for sending customized TCP packets, capable of revealing anomalous behavior of intermediate devices. This scan aimed to identify, on a large scale, the presence of susceptible *middleboxes*, providing an initial view of the national panorama. In the next chapter, the results obtained both in the laboratory environment and in the field scan will be analyzed, with the interpretation of the collected data and discussion about the distribution, intensity, and possible practical implications of these vulnerabilities.

Capítulo 6

Analysis of Experimental Results and the Scan in Brazil

6.1 Statistical analysis and generation of experiment charts

To evaluate the behavior of the TCP reflection attack and the response of the *middleboxes*, two traffic capture files (`.pcap`) were analyzed: one corresponding to the traffic generated by the attacker (`atacante.pcap`, Figura 6.1) and another referring to the traffic received by the target (`alvo.pcap`, Figura 6.2).

The analysis was performed using a Python script, developed with the `scapy`, `pandas`, `matplotlib`, `seaborn`, and `tqdm` libraries, which allowed the reading, processing, and visualization of the data extracted from the captures.

In Figura 6.1, it is possible to observe the sending of forged packets by the attacker, with spoofed source IP addresses (corresponding to the victim), containing `SYN` sequences, which aims to simulate a legitimate communication with services potentially blocked by *middleboxes*. These packets are responsible for triggering the *middlebox* to generate reflected responses.

In Figura 6.2, the direct result of this manipulation is visualized: packets sent by the firewall (*middlebox*) in response to the forged requests, arriving at the victim's *host*. These packets represent the reflected traffic and are the practical evidence of the exploitation of *middleboxes* as attack vectors.

Initially, the script extracted the TCP packets present in the captures, collecting information such as timestamp, flags, and packet size. Then, the data was filtered based on a user-configurable time interval (in seconds), allowing the analysis to focus on specific time windows.

In addition, the relative time of each packet was calculated in relation to the start of the capture, facilitating the temporal visualization of the traffic behavior and enabling the generation of charts that highlight patterns, traffic peaks, and variations in the *middlebox*'s response throughout the attack execution.

No.	Time	Source	Destination	Protocol	Length	Info
17	0.001815	192.168.24.61	66.254.114.41	TCP	54	30544 → 80 [SYN] Seq=0 Win=8192 Len=0
18	0.001905	192.168.24.61	98.143.146.7	TCP	54	45783 → 80 [SYN] Seq=0 Win=8192 Len=0
19	0.002032	192.168.24.61	66.254.114.79	TCP	54	8618 → 80 [SYN] Seq=0 Win=8192 Len=0
20	0.002120	192.168.24.61	157.240.13.35	TCP	54	42221 → 80 [SYN] Seq=0 Win=8192 Len=0
21	0.002247	192.168.24.61	157.240.13.35	TCP	54	24639 → 80 [SYN] Seq=0 Win=8192 Len=0
22	0.002336	192.168.24.61	98.143.146.7	TCP	54	38563 → 80 [SYN] Seq=0 Win=8192 Len=0
23	0.002474	192.168.24.61	157.240.13.35	TCP	54	5355 → 80 [SYN] Seq=0 Win=8192 Len=0
24	0.002562	192.168.24.61	98.143.146.7	TCP	54	42691 → 80 [SYN] Seq=0 Win=8192 Len=0
25	0.002673	192.168.24.61	98.143.146.7	TCP	54	12898 → 80 [SYN] Seq=0 Win=8192 Len=0
26	0.002763	192.168.24.61	66.254.114.41	TCP	54	45431 → 80 [SYN] Seq=0 Win=8192 Len=0
27	0.002892	192.168.24.61	66.254.114.79	TCP	54	17521 → 80 [SYN] Seq=0 Win=8192 Len=0
28	0.002980	192.168.24.61	66.254.114.41	TCP	54	31791 → 80 [SYN] Seq=0 Win=8192 Len=0

Figura 6.1: Excerpt of the `atacante.pcap` packet capture in Wireshark, showing the forged packets sent by the attacker with the aim of initiating the TCP reflection attack.

No.	Time	Source	Destination	Protocol	Length	Info
17	9.890023	157.240.13.35	192.168.24.61	TCP	60	80 → 5014 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
18	9.890189	157.240.13.35	192.168.24.61	TCP	60	80 → 50629 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
19	9.890749	66.254.114.79	192.168.24.61	TCP	60	80 → 8618 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
20	9.890983	98.143.146.7	192.168.24.61	TCP	60	80 → 49538 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
21	9.891132	66.254.114.41	192.168.24.61	TCP	60	80 → 46733 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
22	9.891284	98.143.146.7	192.168.24.61	TCP	60	80 → 3933 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
23	9.891425	98.143.146.7	192.168.24.61	TCP	60	80 → 25510 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
24	9.891568	66.254.114.41	192.168.24.61	TCP	60	80 → 30544 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
25	9.891722	98.143.146.7	192.168.24.61	TCP	60	80 → 45783 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
26	9.891897	157.240.13.35	192.168.24.61	TCP	60	80 → 42221 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
27	9.892956	157.240.13.35	192.168.24.61	TCP	60	80 → 24639 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
28	9.892957	157.240.13.35	192.168.24.61	TCP	60	80 → 5355 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Figura 6.2: Excerpt of the `alvo.pcap` packet capture in Wireshark, showing the responses sent by the firewall to the target *host* as a result of the TCP reflection attack during the attack.

6.2 pfSense + pfBlockerNG firewall results

During the tests carried out in a controlled environment, the effectiveness of pfSense allied with the additional pfBlockerNG package was evaluated, which acts as a DNS request filter by redirecting queries to forbidden domains to an internal IP address, where a local block page is served. This redirection keeps the original URL displayed in the browser, providing transparency to the user and a clear visual notification of the block, as illustrated in Figura 4.8 [69].

However, this DNS-based approach imposes relevant limitations against amplification attacks over TCP. Unlike *middleboxes* that intercept connections and perform deep inspection or packet rewriting, pfBlockerNG does not operate as a transparent *proxy* or generate response pages in malicious or forged TCP packets, limiting its amplification potential. The traffic analysis via Wireshark, illustrated in Figura 6.2 and corroborated by the pfSense logs in Figura 6.3, confirmed that, despite the correct sending of forged packets, there was no significant return of amplified traffic, while in legitimate accesses the block page was displayed normally.

According to Sébastien Meniere, the author of the code, part of the absence of a response may be related to the incomplete structure of the test packet payloads, reinforcing that the traffic format and content directly influence the generation of responses. Thus, although effective in blocking manual accesses by DNSBL, the pfBlockerNG architecture does not favor reflection followed by amplification in TCP-based attacks.

For a quantitative evaluation, a five-minute experiment was performed in which the attacker transmitted 2,044,369 packets (150,000,005 bytes) and the target received 2,044,368 packets (122,662,080 bytes), as shown in Figura 6.4 and Figura 6.5. The amplification rate, according to Equação 2.1, was 1.00×, indicating that the volume of the response was, at most, equivalent to that of the request, not constituting effective amplification. However, the reflection attack was proven to be successful, as the forged packets generated responses directed to the target, validating the functioning of the reflection mechanism. Figura 6.6 complements this analysis by showing the distribution of TCP flags in the received packets, evidencing the passive nature of the system's response.

These results indicate that, although pfBlockerNG ensures protection against malicious traffic amplification, it allows for packet reflection, which represents a partial vulnerability. Compared to *middleboxes* that inject HTTP responses, pfBlockerNG proves to be safe for mitigating TCP amplification risks, but does not completely eliminate the reflective potential of the attack.

Block- Last 25 Alert Entries

Date	IF	Rule	Proto	Source	Destination	GeoIP	Feed
May 19 16:15:00	WAN	pfB_IPs_proibidos_v4 (1744722876)	TCP-S	192.168.24.61:11812 Unknown	66.254.114.41:80 reflectededge.reflected.net	Unk	IPs_proibidos_cu... 66.254.114.41
May 19 16:15:00 [1]	WAN	pfB_IPs_proibidos_v4 (1744722876)	TCP-S	192.168.24.61:33045 Unknown	98.143.146.7:80 98-143-146-7-host.colocrossing.com	Unk	IPs_proibidos_cu... 98.143.146.7
May 19 16:15:00	WAN	pfB_IPs_proibidos_v4 (1744722876)	TCP-S	192.168.24.61:29485 Unknown	157.240.13.35:80 edge-star-mini-shv-02-sin6.facebook.com	Unk	IPs_proibidos_cu... 157.240.13.35
May 19 16:15:00	WAN	pfB_IPs_proibidos_v4 (1744722876)	TCP-S	192.168.24.61:43996 Unknown	66.254.114.79:80 reflectededge.reflected.net	Unk	IPs_proibidos_cu... 66.254.114.79
May 19 16:15:00	WAN	pfB_IPs_proibidos_v4 (1744722876)	TCP-S	192.168.24.61:48684 Unknown	98.143.146.7:80 98-143-146-7-host.colocrossing.com	Unk	IPs_proibidos_cu... 98.143.146.7
May 19 16:15:00	WAN	pfB_IPs_proibidos_v4 (1744722876)	TCP-S	192.168.24.61:59306 Unknown	66.254.114.41:80 reflectededge.reflected.net	Unk	IPs_proibidos_cu... 66.254.114.41
May 19 16:15:00	WAN	pfB_IPs_proibidos_v4 (1744722876)	TCP-S	192.168.24.61:13523 Unknown	66.254.114.79:80 reflectededge.reflected.net	Unk	IPs_proibidos_cu... 66.254.114.79

Figura 6.3: pfBlockerNG logs in pfSense, evidencing the successful blocking of domains classified as forbidden during the attack execution.

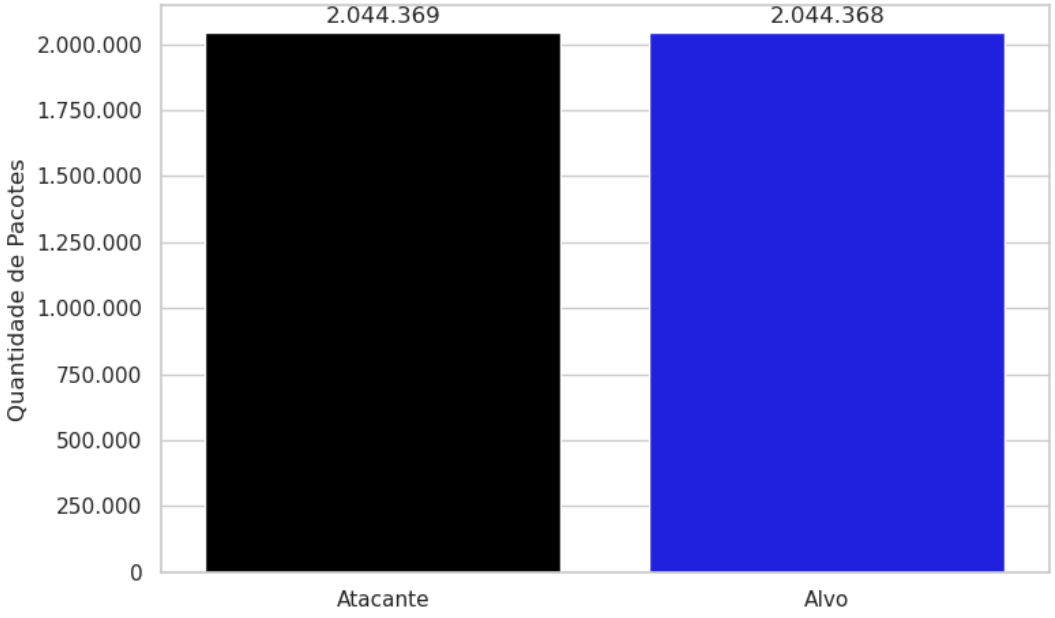

Figura 6.4: Comparison between the number of packets sent by the attacker and the packets received by the target during the 5-minute attack using pfSense with pfBlockerNG as a *middlebox*.

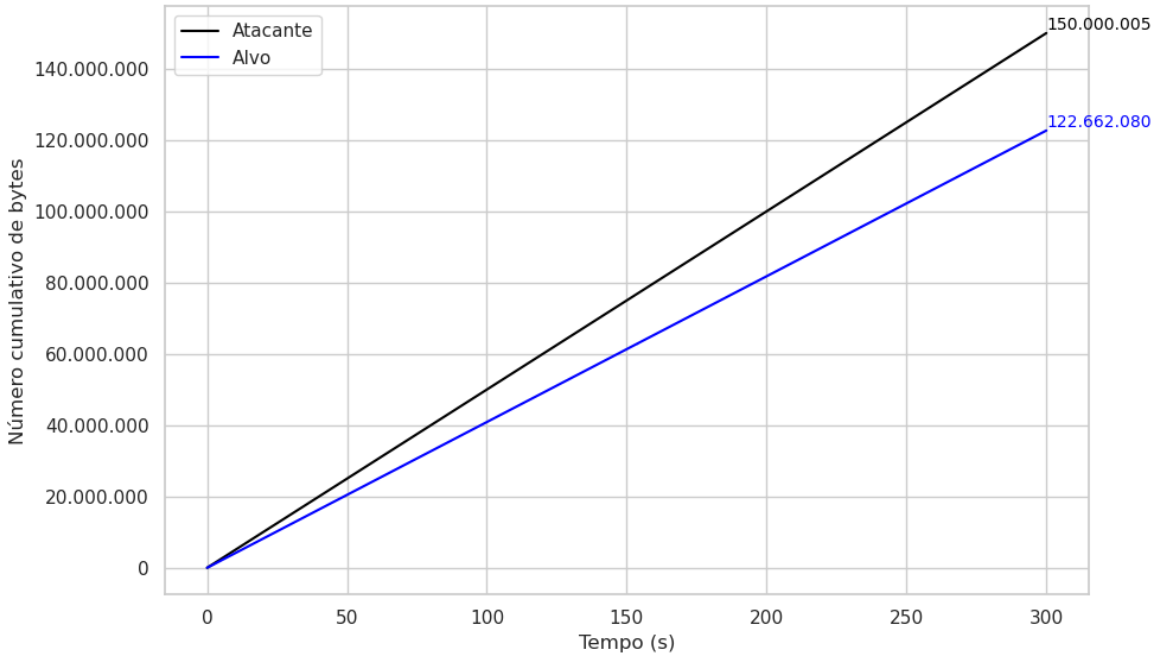

Figura 6.5: Cumulative bytes over the 5-minute period using pfSense with pfBlockerNG as a *middlebox*.

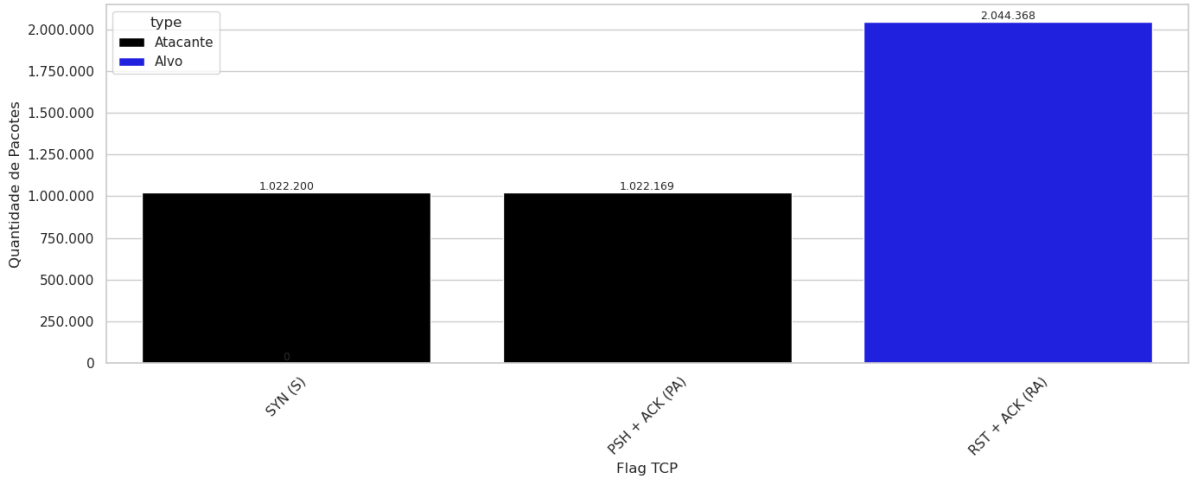

Figura 6.6: Distribution of TCP flags of the attacker and target using pfSense with pfBlockerNG as a *middlebox*.

6.3 FortiGate firewall results

The attack on FortiGate presented a behavior similar to that observed in pfSense, blocking the forbidden domains during the attack execution, as evidenced in Figura 6.7. FortiGate

reflected the packets to the target, but without causing amplification. However, a slight loss was observed in the packets reflected towards the victim, likely due to the additional technologies and advanced filtering mechanisms present in FortiGate, which are more robust compared to those in pfSense.

Despite the expectations regarding FortiGate, the tests showed that this solution also failed to reflect the block page to the attack target. Although it has advanced inspection and filtering resources, FortiGate depends on the complete establishment of the TCP connection (the *three-way handshake*) to apply policies such as content-based blocks or the display of warning HTML pages. Thus, it does not respond to incomplete TCP packets, such as those used in attacks with isolated SYN or ACK flags.

When it receives packets not belonging to valid sessions, for example, forged packets simulating connection attempts without a response from the destination, FortiGate silently discards or blocks the request, without generating visible HTTP responses. This behavior prevents the firewall from responding to out-of-context traffic that could be exploited for amplification.

Thus, even being a robust and professional solution, FortiGate presented the same practical result as pfBlockerNG in the experiment, where it was not possible to provoke the generation and sending of a block page to the external target. This limitation evidences the difficulty of exploiting modern *middleboxes* in TCP reflection attacks, due to the strict handling of incomplete sessions and forged connections.

To quantitatively evaluate the firewall's behavior, a five-minute experiment was carried out in which the attacker sent 2,044,328 packets, totaling 149,999,948 bytes, while the target received 1,384,473 packets and 83,068,380 bytes (see Figura 6.8 and Figura 6.9). The amplification rate, according to Equação 2.1, was 0.68×, indicating that the response volume was about 32.27% less than the original request, that is, there was no effective amplification that represented a real risk of attack in this scenario. Figura 6.10 presents the distribution of the TCP flags in the received packets, providing additional information about the type of response generated.

Date/Time		Source	Device	Destination	Application Name	Result	Policy ID
46 minutes ago		192.168.24.61		157.240.13.35		Deny: UTM Blocked	WAN -> INTERNET (3)
46 minutes ago		192.168.24.61		66.254.114.41		Deny: UTM Blocked	WAN -> INTERNET (3)
46 minutes ago		192.168.24.61		98.143.146.7		Deny: UTM Blocked	WAN -> INTERNET (3)
46 minutes ago		192.168.24.61		98.143.146.7		Deny: UTM Blocked	WAN -> INTERNET (3)
46 minutes ago		192.168.24.61		157.240.13.35		Deny: UTM Blocked	WAN -> INTERNET (3)
46 minutes ago		192.168.24.61		98.143.146.7		Deny: UTM Blocked	WAN -> INTERNET (3)
46 minutes ago		192.168.24.61		66.254.114.41		Deny: UTM Blocked	WAN -> INTERNET (3)
46 minutes ago		192.168.24.61		66.254.114.79		Deny: UTM Blocked	WAN -> INTERNET (3)
46 minutes ago		192.168.24.61		66.254.114.41		Deny: UTM Blocked	WAN -> INTERNET (3)

Figura 6.7: FortiGate logs, evidencing the successful blocking of domains classified as forbidden during the attack execution.

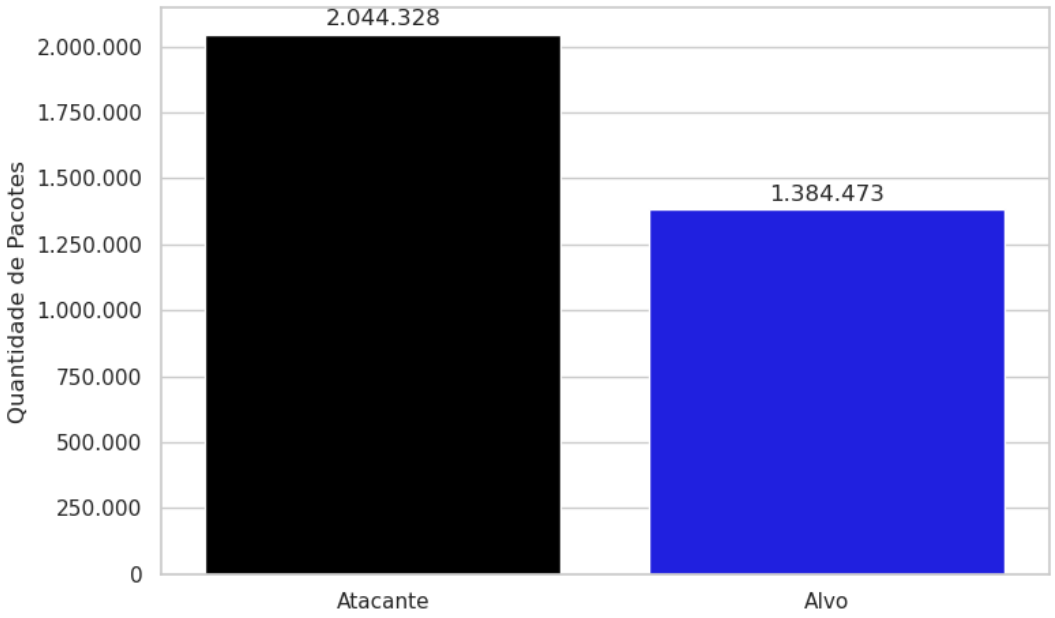

Figura 6.8: Comparison between the number of packets sent by the attacker and the packets received by the target during the 5-minute attack using FortiGate as a *middlebox*.

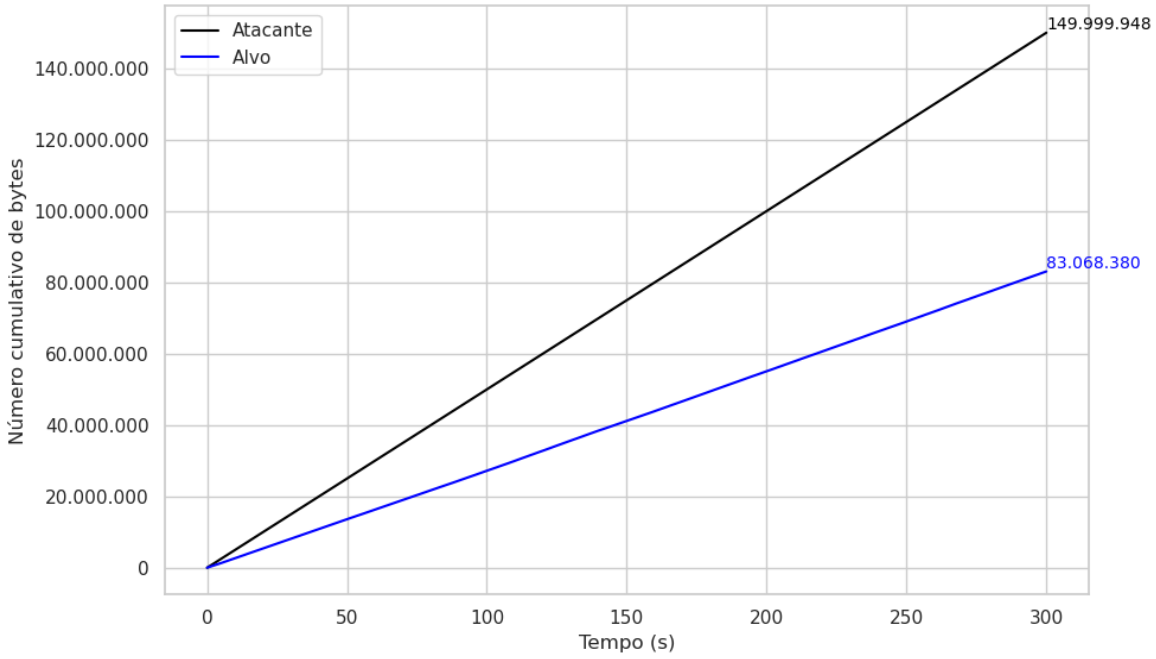

Figura 6.9: Cumulative bytes over the 5-minute period using FortiGate as a *middlebox*.

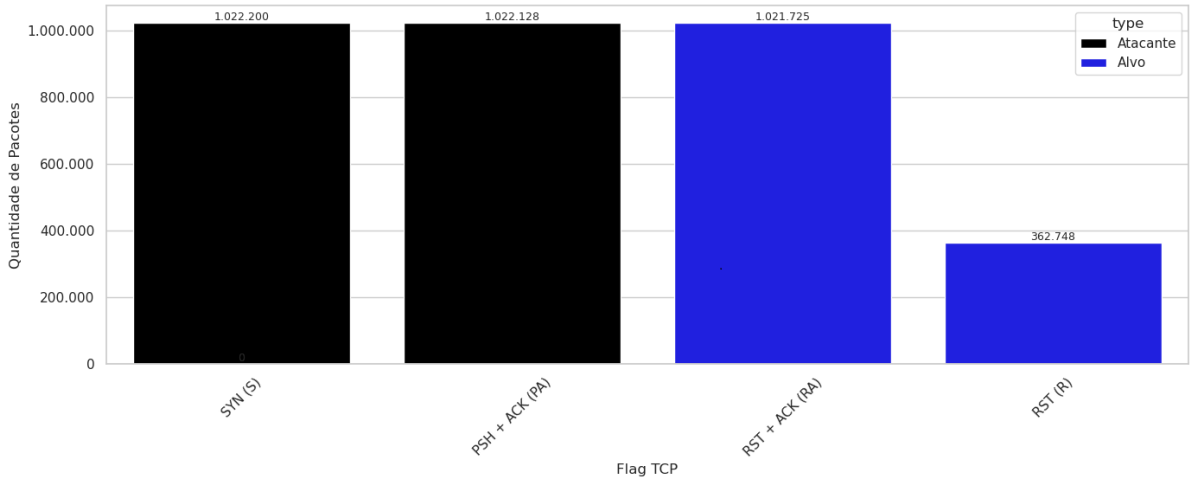

Figura 6.10: Distribution of TCP flags of the attacker and target using FortiGate as a *middlebox*.

6.4 Comparative analysis of the results of the three firewalls

The comparative analysis of the three environments, pfSense with pfBlockerNG, pfSense with Squid and SquidGuard, and FortiGate, showed that while pfSense with pfBlockerNG

and FortiGate allowed the reflection mechanism, the environment with Squid and SquidGuard did not respond to the forged packets, preventing reflection and amplification. However, none of the scenarios showed traffic amplification. In pfSense with pfBlockerNG, almost all the packets sent by the attacker were reflected, resulting in a unitary amplification (1.00×), without the injection of significant content for amplification. pfSense with Squid and SquidGuard effectively blocked legitimate accesses, but did not respond to forged packets, preventing reflection and amplification, due to its dependence on the complete establishment of the TCP connection. FortiGate partially reflected the packets, with an amplification rate of 0.68×, indicating that the target received about 32.27% fewer packets than were sent. This firewall rejects isolated or malformed TCP packets, reducing amplification risks and showing greater robustness in the control and filtering of suspicious sessions.

In general, the results indicate that, in the context of the code used in this work, all the analyzed systems demonstrated resistance to amplification via TCP when confronted with forged packets. Despite this, it was observed that some of them still allow a certain degree of reflection, although insufficient to enable an effective amplification attack. The observed behavior reinforces the idea that, for amplification to occur, the *middlebox* needs not only to respond to the traffic, but also to inject considerable content, such as HTML pages or detailed error messages, even in incomplete connections, which was not verified in the experiments. Table 6.1 presents a comparison of the results obtained, indicating the volume of packets sent by the attacker, the volume of reflected packets received by the target, the percentage received in relation to the total sent, and the respective amplification rate for each tested *middlebox*. The results with Squid and SquidGuard were excluded due to the lack of representative data, since, despite the browser-based block, the attack code's traffic was allowed, indicating an inconsistent behavior of the *proxy* when faced with different types of requests. In the other cases, pfSense with pfBlockerNG reflected practically all packets (almost 100%), with unitary amplification (1.00×), while FortiGate presented a reflection rate of about 67.7

Tabela 6.1: Comparison of the results obtained during five minutes of attack on the *middleboxes* pfSense + pfBlockerNG and FortiGate

Middlebox	**Packets sent**	**Packets received**	**% received**	**Amplification**
pfSense + pfBlockerNG	2,044,369	2,044,368	99.99%	1.00×
FortiGate	2,044,328	1,384,473	67.73%	0.68×

These observations show that, although TCP reflection attack techniques are still functional to some degree, most modern *middleboxes* implement mechanisms that make their exploitation as amplification vectors difficult. In particular, many require the complete establishment of the TCP connection before sending responses with substantial data. Thus, the effective execution of the attack would require modifications to the code used, in order to bypass these barriers imposed by more robust *middleboxes*.

6.5 Analysis of the scan results

Initially, the scan in Brazil was performed in a conventional way, without the use of any additional tunnel or service, but it presented a weak result, with a low number of responses. This is possibly due to internet provider blocks, common in domestic connections, which restrict or filter packets considered anomalous or of high frequency, especially since the scan was performed from a network of this type. To bypass this limitation, the scan was redone using a VPN connection with an exit point in the United States, which allowed for a different and more permissive exit route, resulting in a significantly higher response rate.

According to the result of the `stats.py` script, illustrated in Figura 5.1, and based on the data generated by `zmap` in conjunction with the `forbidden_scan` module, the scan performed in the Brazilian IP address space, covering 40,805,661 IPv4 blocks, identified a total of 13,299,793 unique IP addresses that responded to the sent requests. Among these, 4,541,741 IPs were classified as amplifiers, that is, possible *middleboxes* that returned responses larger than the sent request packets, indicating vulnerability to the amplified reflection attack over TCP (see Figura 6.11). This scan execution had an approximate duration of 2 hours and 26 minutes, as illustrated in Figura 6.12.

The high number of responding IPs identified as amplifiers, more than 4.5 million, does not directly represent the number of distinct *middleboxes* in the Brazilian IP space. Instead, this value reflects the total number of IP addresses that responded to the probes sent by the scan, which may include multiple IPs located behind the same intermediate device. It is common for *middleboxes*, such as firewalls, transparent proxies, and DPI (*Deep Packet Inspection*) systems, to interfere with the traffic of large ranges of IPs, responding on behalf of various devices. Thus, a single *middlebox* can generate responses from hundreds or thousands of IP addresses behind it, inflating the number of IPs classified as vulnerable.

Furthermore, the high number of amplified responses can be attributed to the way these devices handle TCP packets with sensitive content, where even without the complete handshake, many return extensive block messages (such as HTML pages), resulting

in traffic amplification. In the Brazilian context, the widespread adoption of *middleboxes* by providers, institutions, and corporate networks, often misconfigured or outdated, contributes significantly to this scenario. Therefore, the results obtained not only confirm the existence of vulnerable *middleboxes* on a large scale in the country, but also reinforce the hypothesis that these devices are concentrated in certain points of the network infrastructure, being shared by multiple IPs.

A clear example of this phenomenon was observed in the scans performed on two real networks, here called network A and network B to preserve their identity, which, despite having a relatively small infrastructure, showed a significant number of amplifying IPs. In network A, 12,481 unique responding IPs were identified, of which 8,819 were amplifiers, with an average amplification rate of 1.777× and a total of 2,334,588 amplified bytes; in network B, there were 6,319 responding IPs, with 4,444 being amplifiers, with an average rate of 1.772× and a total of 1,173,560 amplified bytes. These results indicate that the high number of amplifying IPs is not necessarily linked to the presence of many distinct devices, but rather to the action of a few strategically positioned *middleboxes*, capable of responding for large blocks of addresses and affecting traffic significantly, as also evidenced by the aggregated results of the scan across the entire Brazilian IP space presented in Table 6.2.

Tabela 6.2: Results of the scans on different IP address blocks.

Network	**Responding IPs**	**Amplifying IPs**	**Non-amplifying IPs**	**Amplified Bytes**	**Average amplification**
Brazil	13,299,793	4,541,741	8,758,052	1,119,928,020	1.655×
A	12,481	8,819	3,662	2,334,588	1.777×
B	6,319	4,444	1,875	1,173,560	1.772×

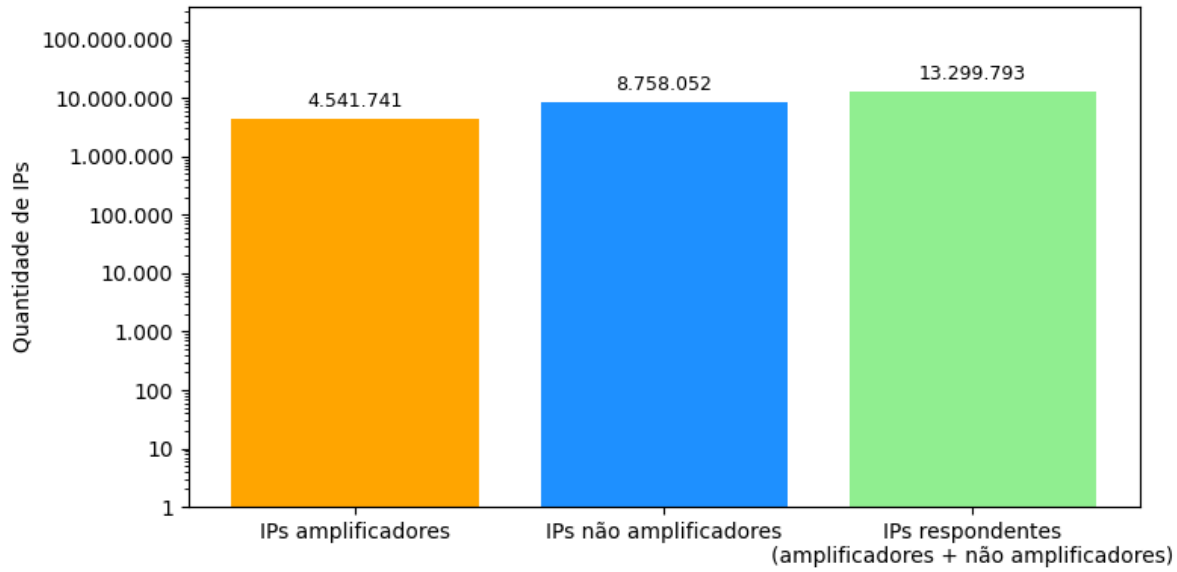

Figura 6.11: Quantitative of IP addresses that responded to the scan in amplifiers and non-amplifiers, including the total number of respondents.

```
0:00 0%; send: 1 0 p/s (16 p/s avg); recv: 0 0 p/s (0 p/s avg); drops: 0 p/s (0 p/s avg); hitrate: 0.00%
0:01 0%; send: 10294 10.3 Kp/s (9.69 Kp/s avg); recv: 0 0 p/s (0 p/s avg); drops: 0 p/s (0 p/s avg); hitrate: 0.00%
0:02 0%; send: 20337 9.98 Kp/s (9.83 Kp/s avg); recv: 0 0 p/s (0 p/s avg); drops: 0 p/s (0 p/s avg); hitrate: 0.00%
0:03 0%; send: 30386 10.0 Kp/s (9.89 Kp/s avg); recv: 0 0 p/s (0 p/s avg); drops: 0 p/s (0 p/s avg); hitrate: 0.00%
0:04 0%; send: 40293 9.90 Kp/s (9.89 Kp/s avg); recv: 5 4 p/s (1 p/s avg); drops: 0 p/s (0 p/s avg); hitrate: 0.01%
0:05 0% (2h26m left); send: 50391 10.0 Kp/s (9.92 Kp/s avg); recv: 5 0 p/s (0 p/s avg); drops: 0 p/s (0 p/s avg); hitrate: 0.01%
0:06 0% (2h26m left); send: 60394 9.98 Kp/s (9.93 Kp/s avg); recv: 20 14 p/s (3 p/s avg); drops: 0 p/s (0 p/s avg); hitrate: 0.03%
0:07 0% (2h26m left); send: 70445 10.0 Kp/s (9.94 Kp/s avg); recv: 22 1 p/s (3 p/s avg); drops: 0 p/s (0 p/s avg); hitrate: 0.03%
0:08 0% (2h26m left); send: 80407 9.94 Kp/s (9.94 Kp/s avg); recv: 23 0 p/s (2 p/s avg); drops: 0 p/s (0 p/s avg); hitrate: 0.03%
0:09 0% (2h26m left); send: 90480 10.0 Kp/s (9.95 Kp/s avg); recv: 23 0 p/s (2 p/s avg); drops: 0 p/s (0 p/s avg); hitrate: 0.03%
```

Figura 6.12: Screenshot of the scan execution using `zmap` along with the `forbidden_scan` module.

This number represents approximately 34% of the responding IPs, a significant proportion that demonstrates the widespread presence of *middleboxes* with undesirable or misconfigured behavior in Brazilian networks (Figura 6.13).

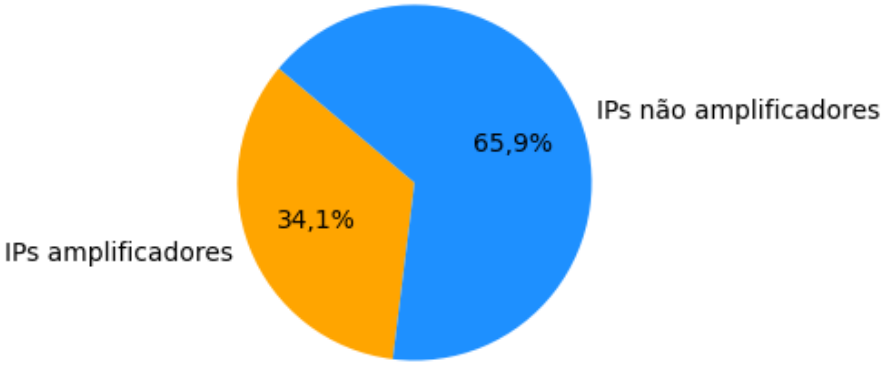

Figura 6.13: Percentage of responding IPs classified as amplifiers and non-amplifiers.

The IPs classified as amplifiers were responsible for transmitting a total of 1,119,928,020 bytes (approximately 1.04 GB) in responses. The average amplification rate observed was 1.655×, indicating that, on average, each byte sent by the attacker resulted in 1.655 bytes of response from the *middlebox*.

Additionally, the TCP flags present in the response packets emitted by the IPs were analyzed. The distribution of these flags helps to understand the behavior of the respondents' network system (see Figura 6.14):

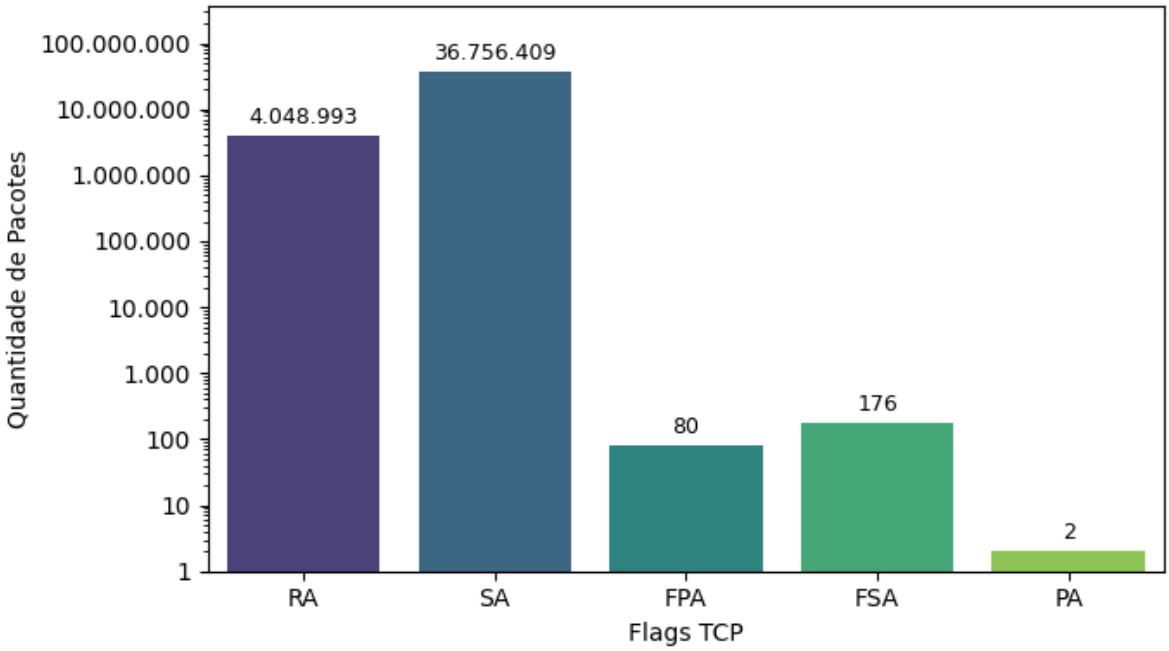

Figura 6.14: Distribution of TCP flags present in the response packets emitted by the IPs.

- **SA (SYN-ACK)**: 36,756,409 packets – represents the majority of responses, characterizing *middleboxes* that respond as if they were establishing connections.
- **RA (RST-ACK)**: 4,048,993 packets – indicates that some systems were rejecting connections immediately after the attempt.
- **FPA (FIN-PSH-ACK)**: 80 packets – rare presence, but indicates flows that were terminated with data still being sent.
- **FSA (FIN-SYN-ACK)**: 176 packets – an atypical combination of flags that may indicate unconventional behavior of some devices.
- **PA (PSH-ACK)**: 2 packets – also rare, but possible in specific *middlebox* implementations.

The predominance of the `SA` flag shows that most of the responses were compatible with systems that respond to SYN packets as if the TCP handshake were being initiated normally. This behavior is particularly useful for amplification attacks, as the attacker can spoof the source address and obtain responses directed to the victim, thus amplifying the traffic volume.

In addition to the aggregated statistics, cumulative distribution functions (CDFs) were also generated for better visualization of the dispersion of values among the amplifying IPs. Figura 6.15 shows the CDF of the number of packets sent by each amplifying IP, while Figura 6.16 represents the CDF of the total amount of bytes transmitted. Figura 6.17 displays the CDF of the amplification rate observed in each IP.

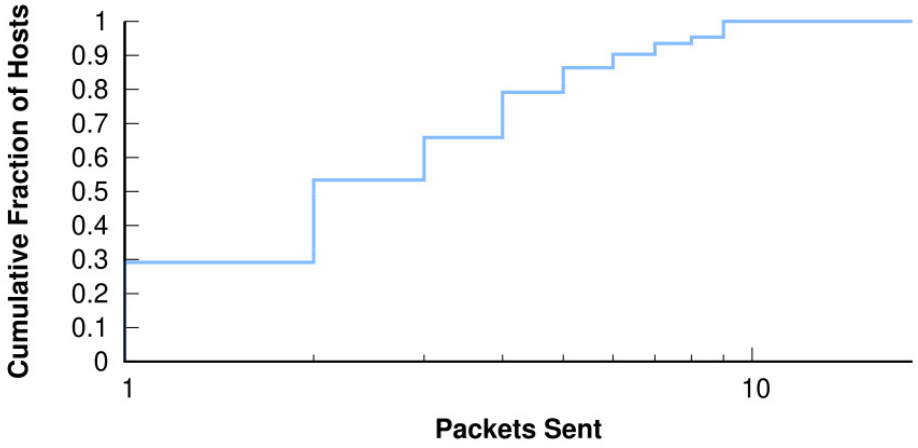

Figura 6.15: Cumulative distribution of the fraction of amplifying *hosts* as a function of the number of packets sent by each IP.

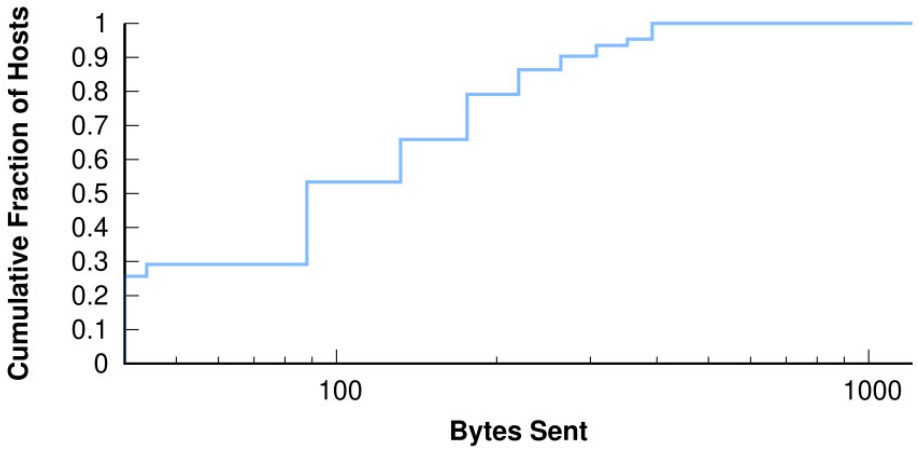

Figura 6.16: Cumulative distribution of the fraction of amplifying *hosts* as a function of the number of bytes sent by each IP.

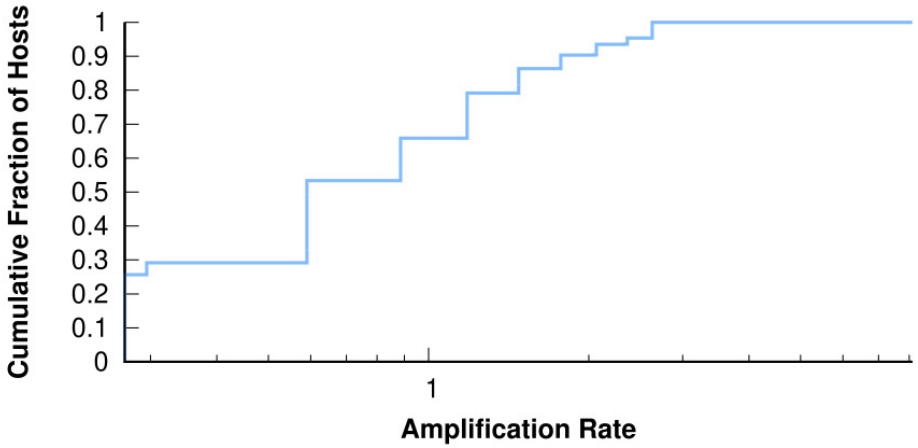

Figura 6.17: Cumulative distribution of the fraction of *hosts* as a function of the amplification rate observed in each amplifying IP.

These curves allow observing, for example, how many amplifying IPs responded with more than a certain number of packets or bytes, and how many presented amplification rates greater than 1.5 or even 10x, being essential to characterize the offensive power of these *middleboxes* in real distributed attack scenarios.

6.6 Chapter Summary

In this chapter, the results obtained in the laboratory experiments and in the Brazilian internet scan were presented and discussed, with a focus on identifying potentially vul-

nerable *middleboxes*. Different firewall configurations were analyzed, including pfSense with pfBlockerNG, pfSense with Squid and SquidGuard, and FortiGate, observing their behavior when faced with the TCP reflection attack. The data from the large-scale scan provided an overview of the distribution of these vulnerabilities in Brazil. In the next chapter, the main conclusions of the study are presented, along with suggestions for future work, such as performing tests with other *middleboxes* and improving the source code, in collaboration with the original developer of the analyzed tool.

Capítulo 7

Conclusion and Future Works

This work aimed to analyze the feasibility of amplified reflection attacks over the TCP protocol, by exploiting the behavior of *middleboxes*. The research was structured in two main stages: (i) the construction of a controlled laboratory environment for experimentation and (ii) the performance of an active scan on Brazilian IP address blocks, with the intention of identifying devices susceptible to this type of attack. Both phases were successfully conducted, allowing not only to achieve the proposed objectives, but also to contribute significantly to the technical and practical literature on the topic.

In the first stage, a testing environment composed of three distinct scenarios was developed, all based on the same network topology. Each scenario involved an attacking machine, a target machine, and different firewall solutions, representing possible *middleboxes*: (i) pfSense with pfBlockerNG, (ii) pfSense with Squid and SquidGuard, and (iii) FortiGate. The objective was to observe the behavior of these *middleboxes* when faced with forged packets with IP spoofing, simulating traffic from potentially censored domains.

Although not all scenarios produced amplified traffic in large volume, the criterion adopted to determine the susceptibility of the devices — based on the methodology proposed by Bock et al. [3] — was the presence of any reflected response to the target from packets forged by the attacker. In two of the three scenarios, the sending of response packets by the *middlebox* to the target was observed, validating the functioning of the reflection attack in a controlled real environment. This practical validation, based on public and reproducible code, represents an unprecedented and relevant contribution to the technical understanding of the attack, also considering the difficulty in adequately configuring the environments and precisely capturing the traffic for analysis.

The second stage involved the execution of an active scan on Brazilian IP address blocks. Probe packets were sent with the forged source address, simulating traffic coming from a censored destination. Several devices responded to these packets, indicating the

existence of *middleboxes* in operation, susceptible to exploitation for amplification and reflection purposes. Although not all IPs responded with intensity, the simple presence of a response according to the criteria of Bock et al. was sufficient to classify them as susceptible.

It is concluded, therefore, that the objectives of this work were fully achieved. It was demonstrated that the amplified reflection attack via *middleboxes* is technically viable both in a controlled environment and in a real context, evidencing a concrete threat to the security of networks and infrastructures. The results obtained reinforce the importance of awareness about this attack vector, especially in environments that use content-based filtering systems.

References

[1] Silva, Eduardo J. A., Gabriel R. L. Andrade e Rogério L. S. Oliveira: *Ataques Negação de Serviço Distribuído (DDoS): o Que é e Como Prevenir*. 2024. 1, 11

[2] Nexusguard: *DDoS Trend Report 2024*, 2024. `https://www.nexusguard.com/threat-report/ddos-trend-report-2024`, acesso em 2025-04-08. 1

[3] Bock, Kevin, Abdulrahman Alaraj, Yair Fax, Kyle Hurley, Eric Wustrow e Dave Levin: *Weaponizing Middleboxes for TCP Reflected Amplification: Censors pose a threat to the entire Internet.*, agosto 2021. `https://geneva.cs.umd.edu/posts/usenix21-weaponizing-censors/`, acesso em 2025-03-27. 1, 2, 4, 12, 14, 16, 17, 18, 23, 25, 26, 44, 45, 46, 68

[4] Akamai Security Intelligence Response Team: *TCP Middlebox Reflection: Coming to a DDoS Near You*, março 2022. `https://www.akamai.com/blog/security/tcp-middlebox-reflection`, acesso em 2025-03-27. 1, 2, 4, 5, 11, 13, 14, 15, 18, 45

[5] Huang, Shan, Félix Cuadrado e Steve Uhlig: *Middleboxes in the Internet: A HTTP perspective*. 2017 Network Traffic Measurement and Analysis Conference (TMA), páginas 1–9, 2017. 1, 4

[6] INCIBE: *TCP Middlebox Reflection: new DDoS attack vector*, maio 2022. `https://www.incibe.es/en/incibe-cert/blog/tcp-middlebox-reflection-new-ddos-attack-vector`, acesso em 2025-03-27. 4, 14, 15

[7] Carpenter, B. e S. Brim: *Middleboxes: Taxonomy and Issues*. Relatório Técnico RFC3234, RFC Editor, fevereiro 2002. `https://www.rfc-editor.org/info/rfc3234`, acesso em 2025-04-29. 4

[8] Benhabbour, Ilies e Marc Dacier: *ENDEMIC: End-to-End Network Disruptions – Examining Middleboxes, Issues, and Countermeasures – A Survey*. ACM Comput. Surv., 57(7), fevereiro 2025, ISSN 0360-0300. `https://doi.org/10.1145/3716372`, Place: New York, NY, USA Publisher: Association for Computing Machinery. 4, 5, 6

[9] Pal, Debashis: *A new DDoS attack vector: TCP Middlebox Reflection*, outubro 2022. `https://blog.apnic.net/2022/10/18/a-new-ddos-attack-vector-tcp-middlebox-reflection/`, acesso em 2025-03-27. 4, 7, 13

[10] Stanford University: *China's Great Firewall*. `https://cs.stanford.edu/people/eroberts/cs181/projects/2010-11/FreeExpressionVsSocialCohesion/china_policy.html`, acesso em 2025-05-02. 5, 18

[11] Daniels, Dan: *What is Deep Packet Inspection (DPI)?*, outubro 2023. https://blog.gigamon.com/2023/10/16/deep-packet-inspection/, acesso em 2025-04-29. 5

[12] Fortinet: *O que é inspeção profunda de pacotes (DPI)?* https://www.fortinet.com/br/resources/cyberglossary/dpi-deep-packet-inspection, acesso em 2025-04-29. 5

[13] Check Point Software: *What is TCP/IP?* https://www.checkpoint.com/cyber-hub/network-security/what-is-tcp-ip/, acesso em 2025-04-30. 6

[14] Fortinet: *What is TCP/IP in Networking?* https://www.fortinet.com/resources/cyberglossary/tcp-ip, acesso em 2025-04-30. 6

[15] Abie, Habtamu: *An Overview of Firewall Technologies*. dezembro 2000. 6

[16] Buckbee, Michael: *What is a Proxy Server and How Does it Work?*, junho 2022. https://www.varonis.com/blog/what-is-a-proxy-server, acesso em 2025-04-30. 7

[17] Fortinet: *What Is A Proxy Server? How does It Work?* https://www.fortinet.com/resources/cyberglossary/proxy-server, acesso em 2025-04-30. 7

[18] Stevens, William Richard: *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley Professional Computing Series. Addison-Wesley, 1994, ISBN 978-0-201-63346-7. 7, 8

[19] Fortinet: *What Is a Firewall? Definition and Types of Firewall*. https://www.fortinet.com/resources/cyberglossary/firewall, acesso em 2025-04-30. 7, 9

[20] Check Point Software: *What is a Firewall? The Different Types of Firewalls*. https://www.checkpoint.com/cyber-hub/network-security/what-is-firewall/, acesso em 2025-04-30. 9

[21] Check Point Software: *O que é negação de serviço (DoS)?* https://www.checkpoint.com/pt/cyber-hub/cyber-security/what-is-denial-of-service/, acesso em 2025-04-26. 10

[22] Cloudflare: *O que é um ataque de negação de serviço (DoS)?* https://www.cloudflare.com/pt-br/learning/ddos/glossary/denial-of-service/, acesso em 2025-04-26. 10

[23] Imperva: *DDoS Attack Types & Mitigation Methods*. https://www.imperva.com/learn/ddos/ddos-attacks/, acesso em 2025-06-16. 10

[24] Kumar, Sumit, Sumit Dalal e Vivek Dixit: *The OSI model: Overview on the seven layers of computer networks*. 2(3), ISSN 2348-120X. Publisher: Research Publish Journals. 10

[25] Akamai: *What Is a DDoS Attack?* https://www.akamai.com/glossary/what-is-ddos, acesso em 2025-03-30. 10, 11

[26] Akamai: *What Is a Low and Slow Attack?* `https://www.akamai.com/glossary/what-is-a-low-and-slow-attack`, acesso em 2025-06-16. 10

[27] Peng, Tao, Christopher Leckie e Kotagiri Ramamohanarao: *Survey of network-based defense mechanisms countering the DoS and DDoS problems.* ACM Comput. Surv., 39, abril 2007. 11

[28] Selamat, Ali, Rizaain Yusof e Nur Udzir: *Systematic literature review and taxonomy for DDoS attack detection and prediction.* International Journal of Digital Enterprise Technology, 1:292, janeiro 2019. 11

[29] Krupp, Johannes, Michael Backes e Christian Rossow: *Identifying the Scan and Attack Infrastructures Behind Amplification DDoS Attacks.* Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, páginas 1426–1437, 2016. `https://doi.org/10.1145/2976749.2978293`, event-place: Vienna, Austria. 11

[30] Gondim, João J. C., Robson de Oliveira Albuquerque e Ana Lucila Sandoval Orozco: *Mirror saturation in amplified reflection Distributed Denial of Service: A case of study using SNMP, SSDP, NTP and DNS protocols.* Future Generation Computer Systems, 108:68–81, 2020, ISSN 0167-739X. 11, 12

[31] Vasques, Alan Tamer e João J. C. Gondim: *Ataques DDoS por Reflexão Amplificada Sobre Refletor IoT Rodando CoAP.* 2020 15th Iberian Conference on Information Systems and Technologies (CISTI), páginas 1–6, 2020. 11

[32] Vasques, Alan Tamer e João J. C. Gondim: *Amplified Reflection DDoS Attacks over IoT Mirrors: A Saturation Analysis.* 2019 Workshop on Communication Networks and Power Systems (WCNPS), páginas 1–6, 2019. 12

[33] Rossow, Christian: *Amplification Hell: Revisiting Network Protocols for DDoS Abuse.* janeiro 2014. 12

[34] Ogaili, Riyadh Rahef Nuiaa al, Selvakumar Manickam e Ali Alsaeedi: *Distributed reflection denial of service attack: A critical review.* International Journal of Electrical and Computer Engineering, 11:5327–5341, dezembro 2021. 12

[35] Azure Network Security Team: *Anatomy of a DDoS amplification attack*, maio 2022. `https://www.microsoft.com/en-us/security/blog/2022/05/23/anatomy-of-ddos-amplification-attacks/`, acesso em 2025-05-01. 12, 13

[36] NETSCOUT: *What is a Reflection Amplification Attack?* `https://www.netscout.com/what-is-ddos/what-is-reflection-amplification-attack`, acesso em 2025-04-30. 12

[37] Kührer, Marc, Thomas Hupperich, Christian Rossow e Thorsten Holz: *Hell of a handshake: abusing TCP for reflective amplification DDoS attacks.* 8th USENIX Workshop on Offensive Technologies (WOOT 14), 2014. 12, 13

[38] Bock, Kevin, Abdulrahman Alaraj, Yair Fax, Kyle Hurley, Eric Wustrow e Dave Levin: *Weaponizing Middleboxes for TCP Reflected Amplification*. 30th USENIX Security Symposium (USENIX Security 21), páginas 3345–3361, agosto 2021. https://www.usenix.org/conference/usenixsecurity21/presentation/bock. 13, 14, 16, 17, 18

[39] Jie Ji: *7 Gbps TCP-Middlebox-Reflection Incident Mitigated by NSFOCUS - NSFOCUS, Inc., a global network and cyber security leader, protects enterprises and carriers from advanced cyber attacks.*, abril 2022. https://nsfocusglobal.com/pt-br/7-gbps-tcp-middlebox-reflection-incident-mitigated-by-nsfocus/, acesso em 2025-03-27. 14, 15, 18

[40] Jie Ji: *Research and Analysis of Middlebox-based TCP Reflective Amplification Attacks*, junho 2022. https://nsfocusglobal.com/pt-br/research-and-analysis-of-middlebox-based-tcp-reflective-amplification-attacks/, acesso em 2025-03-27. 15, 19

[41] Shadowserver Foundation: *Over 18.8 million IPs vulnerable to Middlebox TCP reflection DDoS attacks*, abril 2022. https://www.shadowserver.org/news/over-18-8-million-ips-vulnerable-to-middlebox-tcp-reflection-ddos-attacks/, acesso em 2025-03-27. 15, 16, 45

[42] Branco, Dácio Castelo: *Novo método de ataque DDoS é detalhado por pesquisadores*, março 2022. https://canaltech.com.br/seguranca/novo-metodo-de-ataque-ddos-e-detalhado-por-pesquisadores-212057/, acesso em 2025-03-27. 18

[43] Dickson, Ben: *Middleboxes now being used for DDoS attacks in the wild, Akamai finds*, março 2022. https://portswigger.net/daily-swig/middleboxes-now-being-used-for-ddos-attacks-in-the-wild-akamai-finds. 18

[44] Nguyen, Huy: *Analysis of TCP Amplification DDoS Attacks*, fevereiro 2022. https://www.corero.com/analysis-of-tcp-amplification-ddos-attacks/, acesso em 2025-03-27. 19

[45] The Shadowserver Foundation: *MEDIUM: Vulnerable DDoS Middlebox Report*, outubro 2024. https://www.shadowserver.org/what-we-do/network-reporting/vulnerable-ddos-middlebox-report/, acesso em 2025-04-08. 19, 20

[46] CVE: *CVE-2021-41530*, outubro 2021. https://www.cve.org/CVERecord?id=CVE-2021-41530, acesso em 2025-04-01. 20

[47] Forcepoint: *Security Advisory: TCP Reflected Amplification*, setembro 2021. https://support.forcepoint.com/s/article/000041168, acesso em 2025-04-01. 20, 21

[48] Forcepoint: *Security Advisory: CVE-2021-41530 TCP Reflected Amplification vulnerability*, setembro 2021. https://help.forcepoint.com/security/CVE/CVE-2021-41530.html, acesso em 2025-04-01. 21

[49] NIST: *CVE-2022-27491*, setembro 2022. https://nvd.nist.gov/vuln/detail/cve-2022-27491, acesso em 2025-03-27. 21, 39

[50] SecAlerts: *CVE-2022-2749: TCP Middlebox Reflection*, setembro 2022. https://secalerts.co/vulnerability/CVE-2022-27491, acesso em 2025-03-27. 21, 39

[51] FortiGuard Labs: *TCP Middlebox Reflection*, setembro 2022. https://www.fortiguard.com/psirt/FG-IR-22-073, acesso em 2025-03-27. 21, 39

[52] Tenable: *Fortinet Fortigate TCP Middlebox Reflection (FG-IR-22-073)*, outubro 2024. https://www.tenable.com/plugins/nessus/209716, acesso em 2025-03-27. 21

[53] Vigilance.fr: *Vulnerability FortiOS via TCP Middlebox Reflection*, setembro 2022. https://vigilance.fr/vulnerability/FortiOS-denial-of-service-via-TCP-Middlebox-Reflection-39198, acesso em 2025-03-27. 21

[54] Sangfor Technologies: *CVE-2022-0028: Palo Alto Networks PAN-OS Reflected Amplification Denial-of-Service Vulnerability*, novembro 2022. https://www.sangfor.com/farsight-labs-threat-intelligence/cybersecurity/cve-2022-0028-palo-alto-networks-pan-os-reflected-amplification-denial-of-service-vulnerability, acesso em 2025-03-27. 22

[55] NIST: *CVE-2022-0028 Detail*, outubro 2022. https://nvd.nist.gov/vuln/detail/CVE-2022-0028, acesso em 2025-05-06. 22

[56] Arghire, Ionut: *Palo Alto Networks Firewalls Targeted for Reflected, Amplified DDoS Attacks - SecurityWeek*, 2022. https://www.securityweek.com/palo-alto-networks-firewalls-targeted-reflected-amplified-ddos-attack/, acesso em 2025-03-27. 22

[57] CISO Advisor: *Firewalls da Palo Alto Networks podiam fazer ataques DDoS*, agosto 2022. https://www.cisoadvisor.com.br/firewalls-da-palo-alto-networks-podiam-fazer-ataques-ddos/. 22

[58] Palo Alto Networks: *CVE-2022-0028 PAN-OS: Reflected Amplification Denial-of-Service (DoS) Vulnerability in URL Filtering*, 2022. https://security.paloaltonetworks.com/CVE-2022-0028, acesso em 2025-03-27. 22

[59] Cyber Solutions By Thales: *CVE-2022-0028*, 2022. https://cds.thalesgroup.com/en/tcs-cert/CVE-2022-0028, acesso em 2025-04-02. 22

[60] CWE: *CWE-406: Insufficient Control of Network Message Volume (Network Amplification) (4.17)*. https://cwe.mitre.org/data/definitions/406.html, acesso em 2025-06-10. 22

[61] Greig, Jonathan: *Palo Alto warns of firewall vulnerability used in DDoS attack on service provider | The Record from Recorded Future News*, agosto 2022. https://therecord.media/palo-alto-warns-of-firewall-vulnerability-used-in-ddos-attack-on-service-provider, acesso em 2025-03-27. 22

[62] Check Point CheckMates: *TCP Reflected Amplification*, março 2022. https://community.checkpoint.com/t5/Threat-Prevention/TCP-Reflected-Amplification/td-p/144862, acesso em 2025-03-27. 23

[63] Cisco Community: *How to Protect TCP Middlebox Reflection*, setembro 2022. https://community.cisco.com/t5/network-security/how-to-protect-tcp-middlebox-reflection/td-p/4719042, acesso em 2025-03-27. 23

[64] Meniere, Sébastien: *Ddos-TCP-Middlebox-Reflection-Attack*, 2023. https://github.com/moloch54/Ddos-TCP-Middlebox-Reflection-Attack, acesso em 2025-03-27. 25, 27

[65] VMware: *Fusion and Workstation*. https://www.vmware.com/products/desktop-hypervisor/workstation-and-fusion, acesso em 2025-05-13. 32

[66] Ubuntu: *Download Ubuntu Desktop*. https://ubuntu.com/download/desktop, acesso em 2025-05-13. 33

[67] pfSense: *Download pfSense Community Edition*. https://www.pfsense.org/download/, acesso em 2025-05-13. 34

[68] pfSense: *Getting Started With pfSense Software*. https://www.pfsense.org/getting-started/, acesso em 2025-05-13. 34

[69] pfSense Documentation: *pfBlocker-NG Package*. https://docs.netgate.com/pfsense/en/latest/packages/pfblocker.html, acesso em 2025-05-14. 37, 53

[70] FortiCloud: *VM Images*. https://support.fortinet.com/support/, acesso em 2025-05-13. 39

[71] Fortinet: *Next Generation Firewall (NGFW)*. https://www.fortinet.com/br/products/next-generation-firewall, acesso em 2025-05-13. 39

[72] Durumeric, Zakir, Eric Wustrow e J Alex Halderman: *ZMap: Fast Internet-wide scanning and its security applications*. Em *22nd USENIX Security Symposium*, 2013. 44, 46

[73] Bock, Kevin: *zmap*. https://github.com/Kkevsterrr/zmap, acesso em 2025-03-27. 44