Dário Alexandre
Cancelas Matos

Sistema Open-Source de Alta-Disponibilidade de Segurança de Redes

Open-Source High-Availability Network Security System

Dário Alexandre
Cancelas Matos

Sistema Open-Source de Alta-Disponibilidade de Segurança de Redes

Open-Source High-Availability Network Security System

"*Play the game with a smile on your face*"

— Ryan Serhant

Universidade de Aveiro
2022

Dário Alexandre Cancelas Matos

Sistema Open-Source de Alta-Disponibilidade de Segurança de Redes

Open-Source High-Availability Network Security System

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Paulo Jorge Salvador Serra Ferreira, Professor associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Doutor João Paulo Silva Barraca
professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Paulo Jorge Salvador Serra Ferreira
professor associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro (orientador)

Prof. Doutor Ricardo Morla
professor auxiliar da Faculdade de Engenharia da Universidade do Porto

agradecimentos / acknowledgements

Agradeço primariamente aos meus pais e irmão por todo o apoio incondicional, confiança e orgulho que me fizeram transmitir, desde o início do meu percurso escolar até à conclusão desta etapa, sentimentos que estou certo que irão perdurar. Agradeço também à minha namorada, que me motivou ao longo dos últimos meses e me ajudou a ter consciência do valor do meu trabalho e dedicação, bem como a todos os meus colegas com quem partilhei ao longo dos últimos anos as mesmas aventuras, alegrias, tristezas, e com quem experienciei momentos únicos. Finalmente, agradeço também o apoio do professor Paulo Salvador durante o desenvolvimento desta dissertação, o tempo dispendido e as interessantes conversas que se foram proporcionando ao longo do meu percurso. A todos, um grande obrigado, e até já.

Palavras Chave

Alta-Disponibilidade; Firewall; Load-Balance; Redundância; Escalabilidade; DDoS; IPTables; NFTables;

Resumo

Atualmente, devido ao crescente número de ataques informáticos e conhecimento geral do mundo digital, existe uma necessidade urgente de aprimorar medidas de segurança informática. Algumas destas medidas passam por implementar mecanismos de monitorização da rede. Esta dissertação aborda um sistema de segurança que implementa um mecanismo semelhante, com capacidade de prevenção e reação a possíveis falhas. Foca-se na implementação de uma solução Open-Source com vista a prevenir ataques DDoS e adaptar uma rede a dificuldades vividas em *realtime*, através de configurações inteligentes de equipamento de segurança como Load-Balancers e Firewalls. O sistema é capaz de monitorizar periodicamente o estado destas máquinas, bem como reconfigurar poltíticas de encaminhamento e regras de filtro de tráfego em cenários de falha de funcionamento de equipamento devido a ciber ataques. Mais ainda, providencia uma interface de interação com o administrador de rede para fornecer dados acerca de cada máquina e das ligações que constituem a infrastrutura de segurança, potenciando uma análise dos fluxos de tráfego e aplicação de regras de filtragem em Firewalls. No que toca a tecnologias, foram preparadas configurações tanto em IPTables como NFTables, com vista a ser compatível com o maio número de distribuições de Linux possível. O processo de monitorização e a implementação novas regras é automatizado através da linguagem Python e ligações SSH. O cenário de testes foi simulado em máquinas virtuais através dos softwares VirtualBox e GNS3, interagindo com o sistema implementado num computador real, e todas as funcionalidades definidas ao longo do documento poderam ser testadas com resultados positivos.

Keywords High-Availability; Firewall; Load-Balance; Redundance; Scalability; DDoS; IPTables; NFTables;

Abstract

Due to the growing number of cyber-attacks and the overall digital world knowledge, there is an urgent need to improve cyber security systems. Some of the measures implemented in these systems use network monitoring systems. This document regards a security system with a similar approach focused on prevention and reaction to a shortage of service. It is an Open-Source solution aiming to prevent DDoS attacks and adapt a network to *realtime* failures through smart configurations of security devices like Firewalls and Load-Balancers. The system is capable of periodically monitoring the state of the devices, as well as reconfiguring routing policies and packet filtering rules in scenarios of cyber attacks. Moreover, it provides an interface of interaction with the network admin to deliver data regarding the state of the security equipment and the connection between them, enhancing traffic flow analysis and preventive implementation of traffic filtering rules in Firewalls. As for technologies, these changes in the machines were designed to be implemented in IPTables and NFTables to be compatible with most Linux distributions. The monitoring and reconfiguration process was automated with Python scripts and SSH connections. The whole testing scenario was developed while being simulated with GNS3 and Virtualbox, interacting with a physical computer hosting the system. All functionalities defined along the document were tested and showed positive results.

Contents

List of Figures

List of Tables

Glossário

API	Application Programming Interface
ARP	Address Resolution Protocol
AWS	Amazon Web Services
CDN	Content Delivery Network
Cisco IOS	Cisco Internetwork Operating System
CPU	Central Process Unit
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
DMZ	Demilitarized Zone
DNS	Domain Name System
DoS	Denial of Service
GNS3	Graphical Network Simulator-3
HA	High Availability
HPE	Hewlett Packard Enterprise
HRSP	Hot Standby Router Protocol
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IoT	Internet-of-Things
IP	Internet Protocol
IPsec	Internet Protocol Security
ISP	Internet Service Provider
IT	Information Technologies
IS-IS	Immediate System-to-Immediate System
LB	Load-Balancing
LBaaS	Load Balancing as a Service
MAC	Media Access Control
NAT	Network Address Translation
OS	Operative System
OSPF	Open Shortest Path First
OSS	Open Source Software
PAN-OS	Palo Alto Networks Operative System
REST	Representational State Transfer
RIP	Routing Information Protocol
SME	Small and Medium Enterprises
SSH	Secure Shell
TCP	Transmission Control Protocol
TTL	Time-To-Live
UDP	User Datagram Protocol
VLAN	Virtual Local Access Network
VPC	Virtual Personal Computer
VPN	Virtual Private Network
VM	Virtual Machine

CHAPTER 1

Introduction

This chapter describes the motivation for developing this document, addressing the relevance of modern-day network attacks and the work regarding defense mechanisms.

1.1 Motivation

Over the last few years, technological advancements resulted in a peak of innovation in many distinct areas. Some of those areas had to adapt to the technological shift, such as education [1], [2] and media, which not only allowed for a more straightforward process of content consumption but also made accessible a greater variety of knowledge. Thus, the knowledge of networking and how these areas evolved was more accessible, making the general population more aware of how it was developed, and therefore, its vulnerabilities. Since a more significant portion of the population had more access to this information, the number of cyberattacks also peaked and continues growing, which naturally causes the need for more cybersecurity solutions [3], [4]. Moreover, since many business sectors considered essential to people's lives are also increasingly reliant on the digital world, these solutions require an increase in complexity, making the area more and more valuable. Network security systems and good practices are increasingly precious, as they allow for the rest of the technological solutions not to be compromised and thus keep the innovation flow [5].

1.2 Objectives and Contributions

The main goal of this work is to provide a High Availability (HA) security solution capable of preventing and adapting networks to DDoS attacks, which can also be used as a starting point for developing a more complex solution due to its Open Source character. Furthermore, it aims to provide HA for different network implementations, allowing its integration in devices from multiple manufacturers and its execution in any machine running a Linux distribution. Besides the compatibility features, it is built targeting scalability, allowing the administrator to customize the redundancy level of the system in order to achieve the best scenario regarding the performance-complexity trade-off.

This way, it is simpler to implement than some commercial solutions, widens the range of potentially interested clients, and enhances the customization for said clients.

Besides the features mentioned above, this work provides the functional requirements present in similar systems, such as the dynamic configuration of equipment, monitoring of their health, the proper flow of the system itself, and more. It is, therefore, a solution that aims to help prevent cyberattacks - more specifically Denial of Service (DoS) attacks - and general equipment failure, focused on being a beneficial tool for personal use or Small and Medium Enterprises (SME).

All the work developed is publicly available at *https://github.com/Dariod9/HA_NetworkSecuritySystem*.

1.3 Document Structure

The structure of this dissertation is the following:

- **Technical Background** - describes modern-day concepts regarding Networks, Network Security, and Open Source Software, as well as their role in the system further described,
- **System Architecture and Implementation** - depicts the generic architecture of the system. This chapter contains the use cases, functional and non-functional requirements of the solution and flowcharts of the system operation. It also presents and explains the network and the tools implemented with it. The chapter describes the code workflow, data structures and developed algorithms - for dynamically configuring remote equipment and detecting equipment failures. This chapter also includes the documentation of a REST API developed to provide data regarding the High-Availability system, its functioning, and the equipment status.
- **Tests and Results** - explains the tests made to the system, their results, and comparison with the use cases defined in the previous chapters.
- **Conclusion and Future Work** - presents a conclusion on the dissertation and solution developed, together with a section of future work possible to implement to improve the system's features and performance.

CHAPTER 2

Technical Background

This chapter details the state-of-the-art of current-day solutions regarding HA scenarios, starting with an explanation of the terminology and the mechanisms regarding load-balancing and firewall implementation. It is, thus, listed and explained the frameworks and technologies considered. The second section focuses on the commercial solutions for HA scenarios developed by some of the most popular Network Security companies.

2.1 Networking Background

Currently, the internet as we know it is based on a networking system that allows communication between different machines, no matter the physical distance between them. Many day-to-day productivity and entertainment tools that most people use rely on services hosted remotely (figure 2.1). Thus, the user equipment must know how to reach the servers hosting its desired service. In a traditional IP network, there are many ways of getting to know these paths, such as static routes or routing protocols (like Open Shortest Path First (OSPF),Routing Information Protocol (RIP) or Immediate System-to-Immediate System (IS-IS)). The second group consists of protocols implemented in the network routing equipment that allow them to exchange information about what route to follow to communicate with a specific peer. The first way consists of explicitly declaring routes in each piece of equipment, detailing the **next-hop** of packets for a specific destination. This method requires more network knowledge from the administrator and effort in order to be implemented. Contrarily, the second method relies more on the equipment software and therefore requires more computational power and resources.

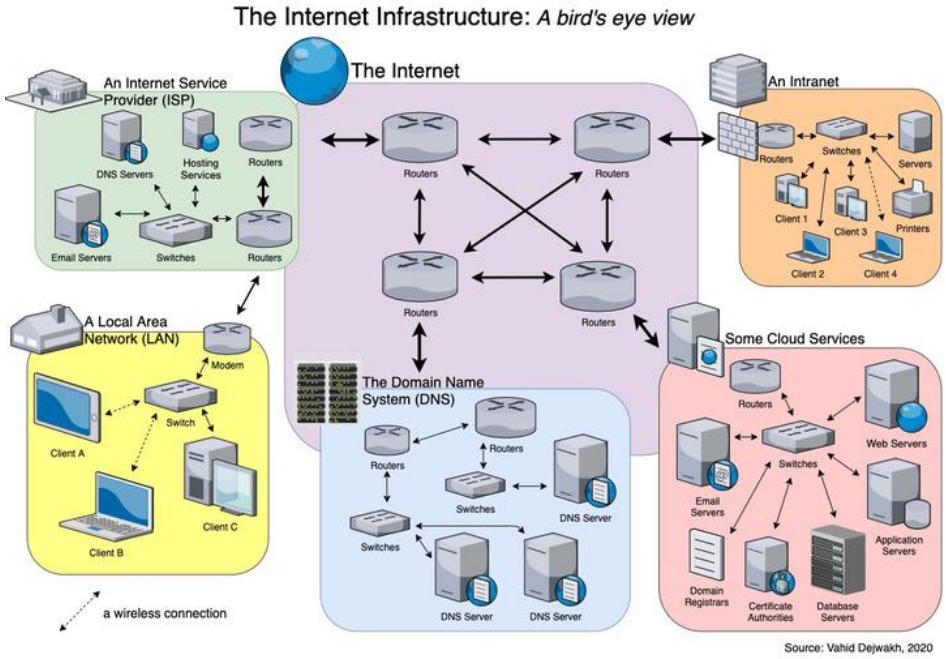


Figure 2.1: Example of communication between various networks[6].

Regarding the service itself, most providers divide their internal network in **VLANs** in order to separate traffic from different areas/services that the same company can run (figure 2.2). In addition, it is also very usual for there to be an exclusive admin VLAN used to set up and monitor the equipment. This separation is very efficient for more intelligent management of resources, preventing continuous access, possible network overload and blocking access to protected resources from non-authorized users.

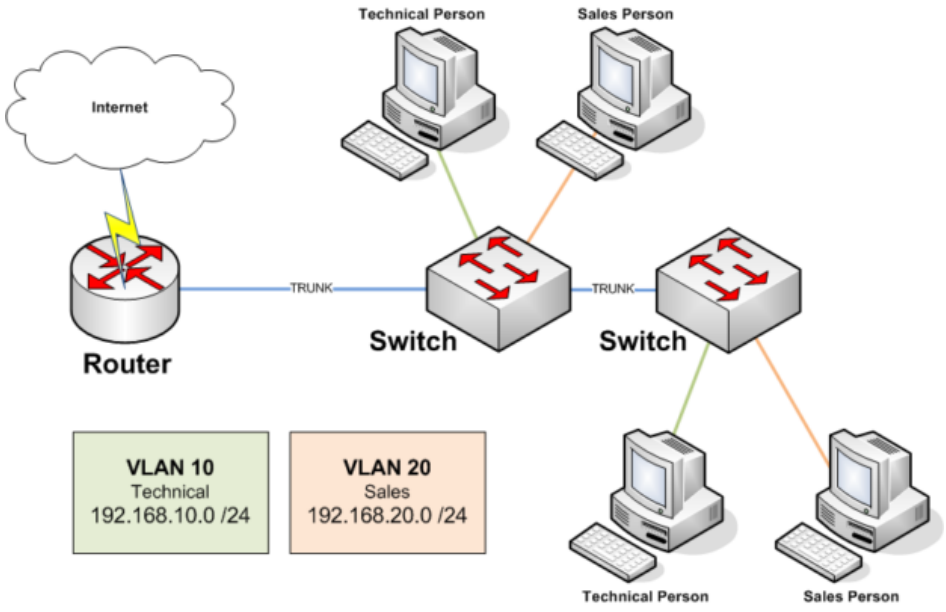


Figure 2.2: Example of VLANs implementation in a corporate network[7]

The subject of VLANs and restricted access introduces the topic of security, which is highly relevant in today's digital world. In order to guarantee a secure system, many measures can be implemented regarding equipment or the connection between them.

With the increase in popularity of some services, now used daily by a significant percentage of the world's population, their reliability and functioning are vital factors. This factor regards both end-users as well as service providers since many services involve payment for the usage of the service itself or payments to the provider by companies seeking advertisement. Moreover, since the number of participants is constantly rising, the network connections can get quite complex and resource-demanding.

In order to ensure the reliability of service, many network managers and providers need to implement redundancy. This solution requires more equipment and configuration between them but is a very efficient way of guaranteeing failover and HA. Entreprises like *Cisco* (Hot Standby Router Protocols (HRSPs)) and *Palo Alto* (in Palo Alto Networks Operative Systems (PAN-OSs)) have this feature built into their products. However, horizontal scaling aiming to ensure redundancy can be an inefficient use of resources. To make the most of the hardware and at the same time reduce the workload, also reducing the chance of equipment failure, traffic distribution mechanisms pose as a great tool.

2.2.1 Cyber Attacks

Some of the most common and unpredictable types of attacks are **DoS** or **DDoS** attacks. These are defined by the causing the failure of network machines, planned and executed by malicious sources. Both attacks aim to disrupt the traffic flow of a network, overloading one or more pieces of equipment or infrastructure with abnormally large amounts of packets. They only differ in the way the attack is executed: the first originates from a unique source of traffic; the second, most common and more challenging to prevent, consists of causing device failure by inducing the processing of numerous packets originating from distinct compromised sources. Also known as *brute-force* DDoS attacks, these may include computers, servers, network equipment, or Internet-of-Things (IoT) devices. The first step of the malicious process is to infect multiple machines with malware, allowing the attacker to control them remotely, forming a **botnet**, as depicted in figure 2.3. Next, the attack targets a victim's network, producing traffic to overwhelm the network's structure.

This kind of attack is particularly hard to detect and protect against because there are many sources of traffic to analyze and compare to what is considered "normal" traffic. Moreover, the infected machines used by the attacker are often seen as legitimate devices that should be able to frequently communicate with the victim's network, which hardens the protection of the network. In addition, because a large number of users commonly accesses the most targeted services, the considered "normal" traffic can already include spikes, which might also seem as an attack, increasing the difficulty of protection.

Another group of DDoS attacks is *semantic* attacks, which target specific protocols or applications that might contain exploitable weaknesses. Also known as *low-rate* DDoS, it

allows the attacker to materialize the attack with a low volume of traffic, which makes it even more similar to the usual traffic of the network, and, therefore, even harder to detect [8], [9].

In order to detect DDoS, the network can be mainly analyzed by looking for suspicious patterns or behaviors. These include significant traffic from a specific address or IP range and unexpected high demand for specific endpoints or resources. Abnormal conduct might also be detected by creating patterns for usual users and comparing them to the network traffic. This comparison might raise alerts based on the excessive similarity of behavior from different users like identical location, equipment, software versions, or based on deviations from the typical pattern, such as spikes at unusual hours of the day or locations.

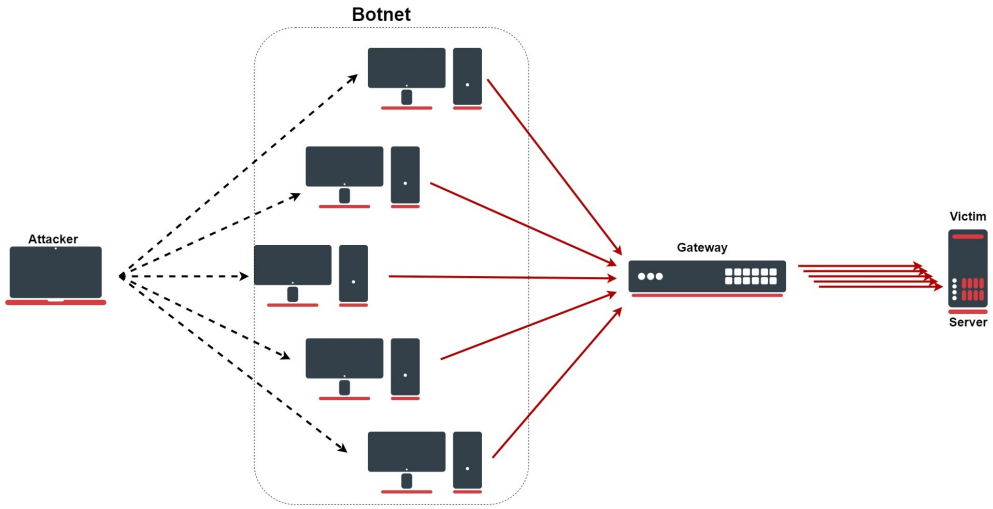


Figure 2.3: Visual representation of a DDoS attack using multiple infected machines targetting the same gateway [10].

2.2.2 Load Balancing

In order to maintain said reliability, one helpful approach to avoid network equipment or server failure is Load-Balancing (LB), which essentially consists of distributing traffic to different machines. This distribution can be but is not forcefully based only on source and destination since different packets that share these attributes in equal values can be load-balanced. Many other parameters can be helpful to load balance packets, such as IP version, protocol, or ports. In addition, other criteria regarding the devices themselves can be used, such as load-balancing to servers with the least bandwidth consumed, the least response time, or even parameters like the maximum number of clients, processing power, or resource utilization. Implementing this feature can happen with software in routing equipment, cloud, or specialized hardware equipment. It is an intelligent way to divide the traffic between machines so that none of them is required to do an overly heavy workload, causing a possible failure due to a lack of cal resources (Internal Storage, Memory, among others).

On-Site Solutions

One way to implement LB on-site only is through specific hardware, with cal appliances dedicated to distributing traffic received to other machines. These solutions are particularly

relevant in scenarios where customization is valued since they do not require communicating with a provider. However, hardware-based Load-Balancers usually require more maintenance, a bigger budget, and can rely more on the network connection and factors like Time-To-Live (TTL).

Cloud-Based Solutions

Often known as Load Balancing as a Service (LBaaS), **Cloud-Based Load Balancers** rely on the internet to offer LB capability to a system. These are services provided by external companies (Google Cloud[1], Cloudflare[2], Oracle[3]) and usually require a subscription fee. With cloud-based solutions, the traffic is redirected to external machines running software dedicated to distributing traffic to other equipment while distributing the workload between themselves and other resource sources, as seen in figure (2.4). This kind of solution raises many advantages, the biggest of them being the lower cost of implementation for the system developer, budget and effort-wise, and the flexibility of configuration in rules. It is a more cost-effective solution due to various reasons. These include guaranteed LB capability, scalability of the scenario, management of resources by the LBaaS provider, or lower cost in the long-term, even if higher at first. Among the software-based tools for LB, one can highlight two groups: Cloud-native or Platform-agnostic. The second includes any solution running in network equipment of a service, providing its own HA functionalities. An example of the first kind is *Traefik*[4], a popular tool that provides Load-Balancing, reverse-proxy and other network functionalities through a cloud infrastructure. While it is still a modern solution, it has proven to be a viable tool in performance tests with other popular solutions, with similar results to *HAProxy*[5] and even outperforming *NGINX*[6] [11], [12].

[1] `https://cloud.google.com/`
[2] `https://www.cloudflare.com/`
[3] `https://www.oracle.com/pt/`
[4] `https://traefik.io/`
[5] `https://www.haproxy.org/`
[6] `https://www.nginx.com/`

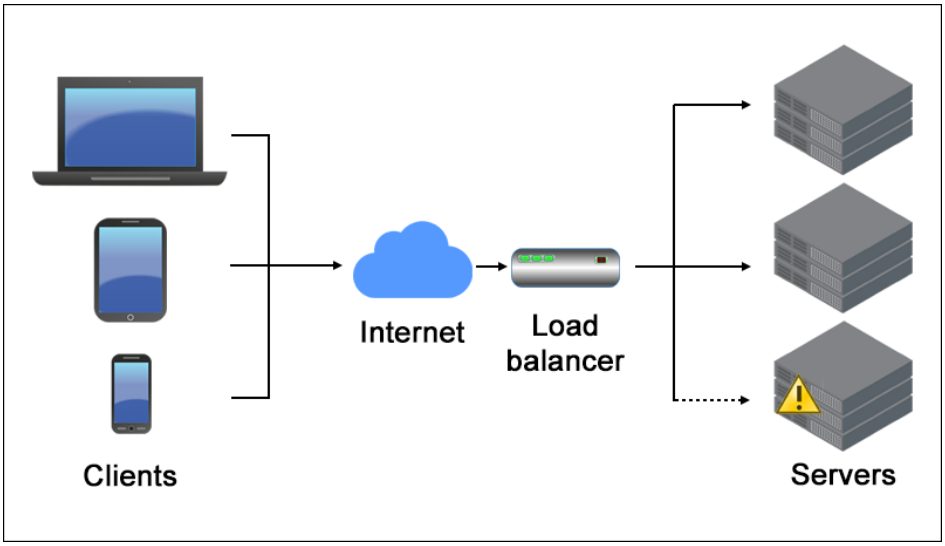


Figure 2.4: Example of the implementation of Load-Balancing to distribute traffic accessing servers [13]

2.2.3 Firewalls

One advantage of Load-Balancing is providing resilience to the network against possible attacks, while deducing the possible source of the attack or separating the most suspicious activity, based on the effect produced on the servers and network equipment. After obtaining the source of the harmful traffic, **Firewalls** might be handy to mitigate possible attacks. This kind of equipment is often responsible for controlling the traffic in a network, allowing the implementation of complex rules to filter, accept, reject, or drop said traffic, acting in inbound, pre-routing, post-routing, or outbound packets. They are often divided into *Stateful* or *Stateless*. The first classification inspects a variety of data in packets and the communication channels, linking that information to pattern behaviors that can help the decision of filtering. The latter type focuses mainly on simple packet parameters like source and destination addresses, ports, protocols and lengths. If these characteristics differ from the administrator's acceptance rules, the firewall filters the packet.

Similarly to Load-Balancers, Firewalls too can be hardware or software-based. Again, both create a boundary and protect one or more target networks.

Hardware-Based Firewalls

Hardware-Based Firewalls are appliances specifically designed to manage traffic destined for specific networks and from various sources. This kind of equipment requires more in-depth knowledge to implement and manage throughout its usage and is usually the chosen solution by entities that value security above other features. Therefore, the cost of these machines can also be higher than the subscription of a similar software service. Although, since managing the workload is usually done before traffic reaches firewalls, the need for physical resources is questionable. This way, the cost of a Hardware-Based Firewall can be worth it in the long run since software-based solutions might require an increase in cost over time, adapting to the system's needs.

Software-Based Firewalls

Regarding Software-Based Firewalls, one can note the more straightforward implementation and monitoring of the firewall since it is software that can be present in many different machines, originally designed to perform different roles. Furthermore, software Firewalls can be implemented in routers and personal computers, aiming to protect the network or specific devices. These characteristics make it a more flexible, user-friendly solution and one that can offer a more customized experience. Due to its lightweight processing requirements, it is commonly used to offer protection against specific platforms or websites [14]. Despite this feature, they can require more overall setup time, depending on the level of protection desired, since its implementation may involve the configuration of multiple devices. Software-Based Firewalls can also work as a service provided by external entities (*Sophos*[7], *CheckPoint*[8], *Huawei*[9]), which can lead to higher costs, directly depending on the complexity of the solution. Commercial and Open-source solutions will be explored in the next section.

2.2.4 Commercial Solutions

Palo Alto Networks

Palo Alto Networks, a cyber security company based in California, provides a HA solution based on the connection between two machines running *PAN-OS (Palo Alto Networks Operative System)*. This connection turns the two peers into *Stateful Firewalls* since they keep track of flow information, share synchronized network data and policy configurations and use such information to change their behavior based on the hypothetical failure of the other peer. However, administration data and log data are not synchronized. This solution establishes HA policies between up to sixteen peers, limiting the scenario of having multiple firewalls configured similarly and making it more complex to implement a more secure and redundant system. Moreover, the *Palo Alto Networks* HA solution shows little flexibility regarding the nature of the devices in use since it requires PAN-OS for its implementation. That being said, the operative system is based on Linux, which may indicate similar limitations compared to other Linux-based Operative System (OS)s. It is also not feasible for small clients since the lowest price to pay for a version of said system is 41 580 US dollars. This price represents the lower subscription (VM-1000-HV-ENT), constituing the 3-year renewal of a single device in an HA pair. Another Palo Alto product worth mentioning is the **VM-Series**, a virtualized firewall solution focused mainly on cloud computing security. As such, it is possible to integrate with multiple cloud environments (*Amazon Web Services (AWS)*[10], *VMware*[11], *Google Cloud Platform*, among others). However, it must comply with the environmental limitations and policies, resulting in a lack of HA compatibility in some integration cases.

[7] `https://www.sophos.com/en-us`
[8] `https://www.checkpoint.com/pt/`
[9] `https://www.huawei.com/en`
[10] `https://aws.amazon.com/`
[11] `https://www.vmware.com/`

Hewlett Packard

Hewlett Packard Enterprise (HPE)[12] is a company based in San Francisco which also provides a HA clustering solution based on software and cloud integration. *HPE Serviceguard* is designed for Linux, which makes it possible to implement in local devices, and not only in the cloud, which provides the client more privacy over the network data.

CloudFlare

Cloudflare is a company based in San Francisco, USA, specializing in providing Content Delivery Network (CDN), Domain Name System (DNS), and internet security services. Although it presents a more economical solution, it only provides LB features at a cloud level, not fulfilling all the functional requirements of the system addressed in this dissertation. Cloudflare's solution ensures service availability by performing load-balancing of the traffic to and from a client's domain. Therefore, even though the network connection is guaranteed, the solution does not act to block the source of malicious traffic, relying on more services/machine power to maintain the system. This way, one cannot ensure the attack will not happen again or at a level where the available resources will not suffice. Besides that, the solution is DNS-based, limiting the clients' possibility to implement it in their service network. However, Cloudflare's product is SME-friendly since its cost depends on the DNS queries received and the complexity of the availability measures. The most simple subscription costs five United States dollars, plus fifty cents of a dollar for every five hundred thousand queries, providing two origins, five origins per pool, sixty-second health checks, and checks from only one region.

2.2.5 Secure Structures

The equipment described previously usually constitutes the security infrastructure of networks. The general network scenario implemented by a service provider consists of various networks for different purposes, such as different services, data centers, or even buildings. One of the networks is often representative of the connection to the Internet Service Provider (ISP). Each of the other networks contains equipment dedicated to communicating with one another and the general Internet. However, while each zone might cover security devices for specific needs, other machines are used as a *man-in-the-middle* to analyze the traffic and thus help with the routing and maintaining the network health. A similar structure is present in figure 2.5.

[12] `https://www.hpe.com/`

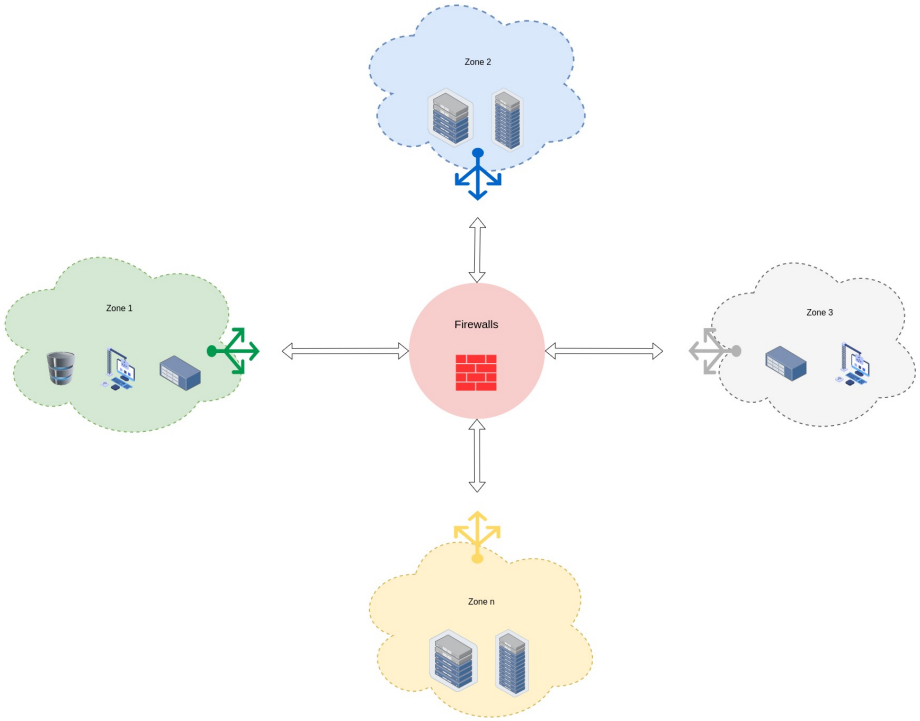


Figure 2.5: Generic network topology with different zones of access

2.2.6 OSS Solutions

pfSense (Integrated Solution)

Combining features from both types of devices described in Section 2.2.2 and 2.2.3, pfSense[13] is an Open-Source solution for packet filtering and routing developed by Netgate[14]. Its main advantages are the lower cost it requires to implement and maintain, due to its open-source character, and the flexibility of deployment since it targets both on-premises and cloud infrastructures. Furthermore, it has proven to excel in performance, having been implemented widely in businesses and governments, it also provides always-available tech support for its users. Due to the high demand for pfSense products, the Netgate team offers training to the customers and different plans for enterprises or individual deployments, containing different features and pay-per-use pricing in the more complex scenarios.

VyOS (Integrated Solution)

Similarly to the system approach in this dissertation, *VyOS*[15] is an Open-Source solution developed by *Vyatta* to ensure network security and HA through packet filtering and routing [15]. It is a free operating system based on Linux distributions that allows the implementation of Stateful and zone-based firewalls with its integration in a physical device, as well as Dynamic Host Configuration Protocol (DHCP), Network Address Translation (NAT) and policy-based routing features. Moreover, besides the Firewall functioning, it also allows

[13] `https://www.netgate.com/pfsense-plus-software`
[14] `https://www.netgate.com/`
[15] `https://vyos.io/`

for more security-related features like Internet Protocol Security (IPsec) tunnels and quality of service methods.

Netfilter (Package Solutions)

Providing tools similar to Software-Based Firewalls, Netfilter is a free Open Source Software (OSS) project focused on delivering network analysis, management, filtering, and overall packet mangling solutions. Netfilter's software comes embedded in Linux kernels and allows developers to use many of its programs [16].

IPTables (software)

Often referred to as Linux's firewall, IPTables is a command line program by Netfilter that allows the user to configure rule sets processed at the kernel level, thus implementing firewall functionalities and NAT to the OS. Said rules are applied to each packet sequentially, and the action occurs if the criteria defined matches packet data, usually found in the packet header. To a list of rules is given the name of **chain**. IPTables is, by default, composed of the PREROUTING, INPUT, OUTPUT, FORWARD, and POSTROUTING chains, according to the default steps of packet processing, allowing the user to manage different packets in different stages of the flow. Also built-in are five **tables**, all designed for different purposes:

- *raw* - for packet configuration separating connection data;
- *filter* - to implement typical firewall actions, filtering packets based on specified criteria;
- *NAT* - for network address translation;
- *mangle* - aiming to alter packets in specific ways, such as packet marking or connection marking;
- *security* - to implement access control networking rules;

Each table consists of different chains, and even though creating or deleting chains is possible, the tables are unalterable. The relation between chains and tables is depicted shortly in figure 2.6.

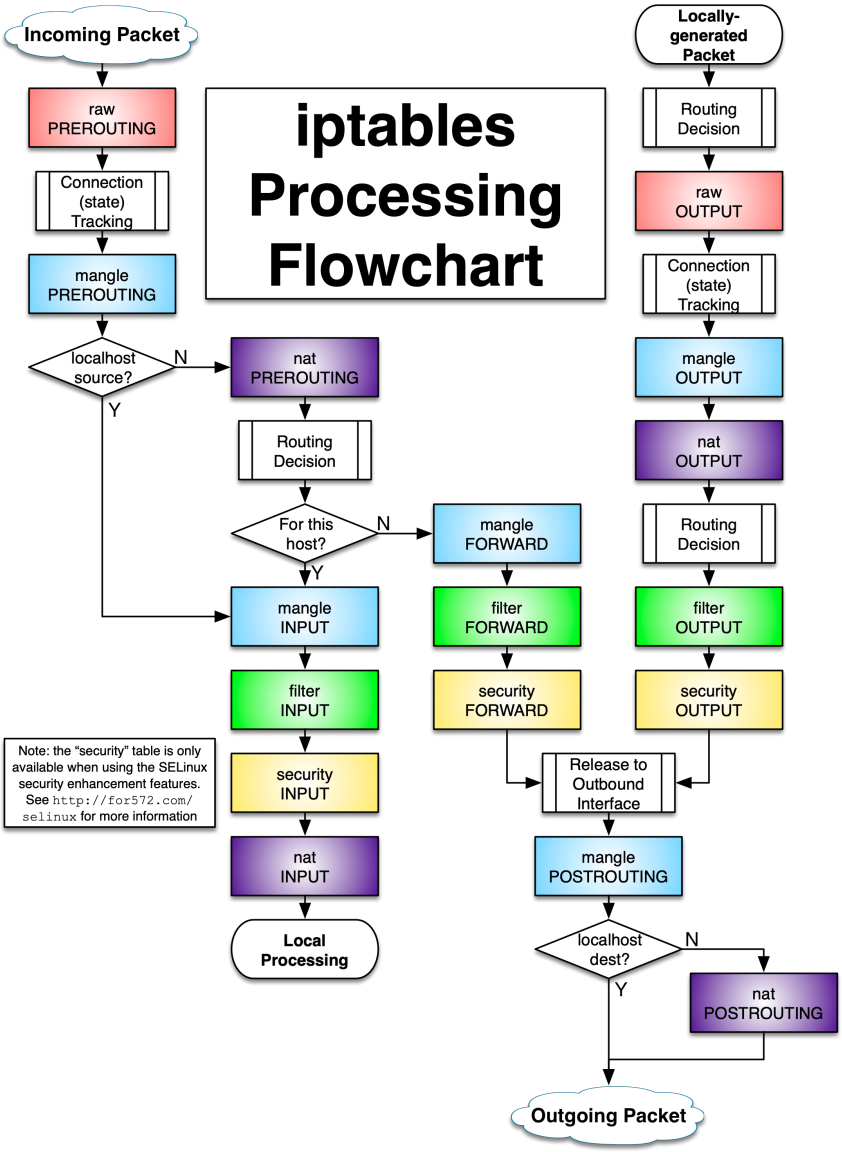


Figure 2.6: IPTables flowchart of processing for each packet (based on [17])

While each table is composed of different chains, the processing of packets does not follow said chains sequentially. Since these correspond to different steps of the flow, packets can "jump" to different chains of different tables according to the rules defined by the user. For example, despite the definition of chain-including tables, a possible flow includes the transition from "mangle `input`" to "filter `input`", swapping the table but keeping the chain.

Regarding the IPTables possible actions, the program can manipulate network flows in a variety of ways. It offers the possibility to implement simple filtering capabilities through the standard options (`accept`, `drop`, `queue`, `reject`) and keeps a record of that handling via the LOG functionality. Moreover, it also provides the ability to implement routing mechanisms with the `output` and `forwarding` options. Said mechanisms can get quite complex since the software allows for decision-making in various stages of the packet-handling process. Finally, IPTables is also capable of forcing a jump (`jump`) between different chains for different

processing goals, as well as marking packets, (`mark`) option, which allows for more composite filtering and forwarding algorithms, either by the program itself or with the help of other frameworks who analyze packet data and perform accordingly. This marking feature extends the matching criteria beyond the default flow parameters already supported, which include Transmission Control Protocol (TCP) protocol details (source, destination, flags and other options), User Datagram Protocol (UDP) destination and source, and Internet Control Message Protocol (ICMP) protocol types. Lastly, the software also supports matching via internal parameters - like the total number or rate of packets processed by a specific rule, called `limit` - and more peculiar details like Media Access Control (MAC) addresses and packet states.

Despite the broad range of operations IPTables provides and its complex structure, the most common use cases are relatively easy to implement since basic firewall rules or NAT requires little knowledge of the program and a reduced number of rules [18].

NFTables (software)

NFTables is a new framework by Netfilter designed to replace IPTables, based on a network-specific Virtual Machine (VM). It uses the same infrastructure and resources already present in the Linux kernel to support IPTables. However, it provides a wider range of functionalities, namely IPv6 packet handling, Address Resolution Protocol (ARP) packet filtering and Ethernet bridge packet manipulation [19]. NFTables performs while offering a more efficient and user-friendly experience - a comparison is done in listing 2.2.6 and 2.2.6. It allows data structures like sets and maps within rules for better memory access and field concatenation. Moreover, it offers a more straightforward syntax, multiple actions in rules, and a more scalable configuration. Contrary to IPTables, NFTables atomically updates rules in run time and does not require rewriting in the Linux kernel every time the user updates the rules [20], [21]. However, its performance is not yet proven to be better, as IPTables shows a very powerful character, hence being the default packet filtering tool for Linux distributions for the past years [22].

```
1 iptables -A INPUT -p tcp -m multiport --dports 80,443 -m
    conntrack -j ACCEPT
```

Listing 2.1: Example of a rule definition with IPTables

```
1 nft add rule ip filter INPUT ip protocol tcp tcp dport{ 80,443}
    counter accept
```

Listing 2.2: Example of a rule definition with NFTables

WireGuard (software)

WireGuard is a security tool that provides a different approach to tunneling. It combines the features of IPSec with Virtual Private Network (VPN) solutions, all implemented as a

Linux virtual network interface, allowing more secure communications, better performance levels and more intuitive implementation. Although the topic of secure communication is not the main focus of this dissertation, WireGuard implements an interesting feature to help avoid DoS attacks. Since encryption processes are known to be Central Process Unit (CPU) intensive, this software mitigates the threats of overloading hardware by not processing a handshake message between the peers if one is under heavy load. Instead, this mechanism is replaced by a cookie mechanism in the destination device, which is later used by the source peer in other messages to guarantee authenticity. This cookie is the result of a mix between a secret that changes every two minutes with the IP address of the entity that initiated the communication. On the destination end, depending on the load, the device chooses to whether process or reject the packet based on its cookie [23].

CHAPTER 3

System Architecture and Implementation

In this chapter, the system's general architecture will be depicted, explaining the use cases and describing the required infrastructure to achieve all its functional requirements. Lastly, the implementation of the solution is referred, along with the algorithms developed and frameworks in use.

3.1 System Overview

This dissertation aims to develop a system to implement over multiple zones to ensure connectivity and HA between all of them, with the help of firewall rules and load-balancing. These features will be implemented at the edge of each zone, filtering and guaranteeing the correct flow of traffic. The system is hosted in a device outside these zones, controlled by the system admin, and performs this implementation and all configurations remotely. In addition, this machine must access all security equipment to ensure their maintenance and retrieve valuable data. An example of a network integrating a similar management solution is seen in figure 3.1.

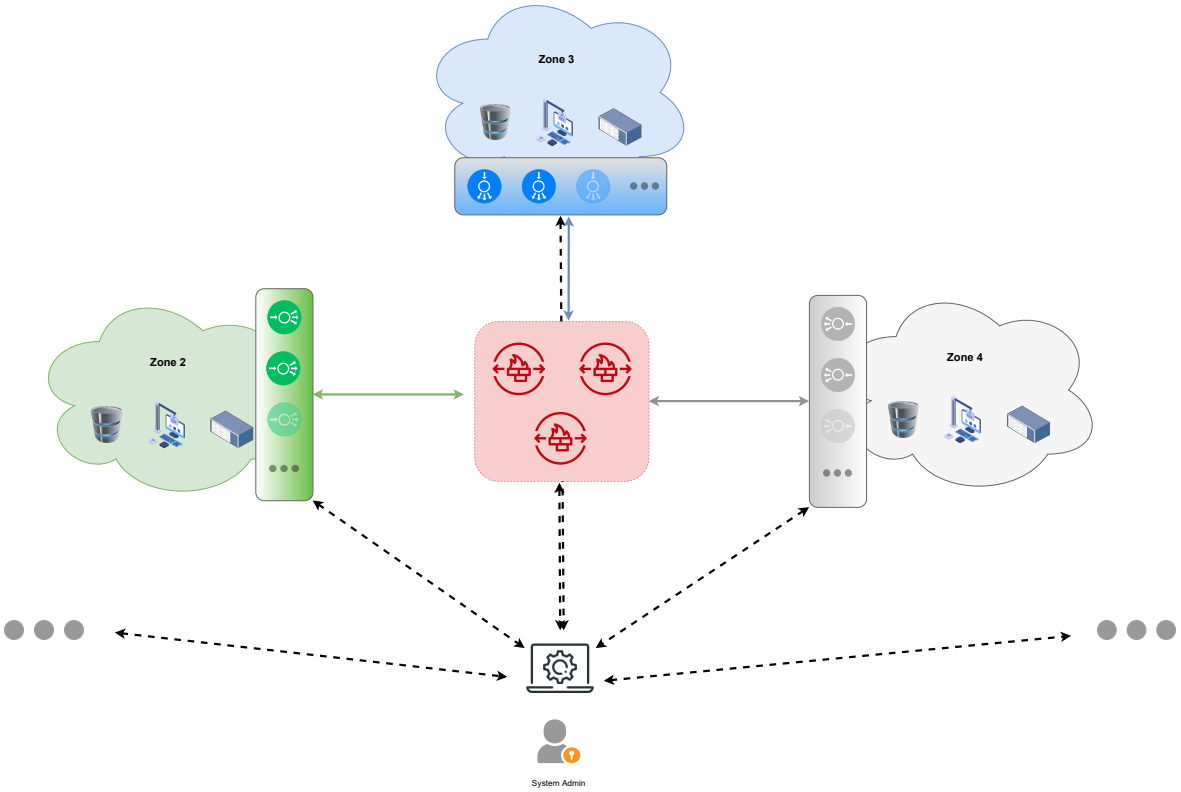


Figure 3.1: Common network management scenario

3.1.1 Functionalities

As the name implies, the HA system described in this work has the primary goal of maintaining the connectivity between the client networks and other services.

However, to achieve this goal while also guaranteeing scalability and flexibility to the system, many functionalities must be implemented.

Add or Remove Equipment

The solution must be able to work with various machines to implement in different network scenarios. It should adapt to the addition and removal of devices, along with a description of their interfaces, firewall zones and other details. The system must adjust its functioning when there are changes in the equipment with no interaction from the admin. This feature offers scalability to the system, provided it is intended to be a long-term solution and is helpful for small networks and clients with more complex infrastructures.

Add or Remove Zones

For a better definition of the equipment and the security infrastructure devices to which they connect, the system allows for removing and adding zones/networks.

Monitor Devices

To maintain the correct functioning of the network, the system monitors each device periodically. The data retrieved is then analyzed and may lead the system to act on the

devices, either via admin order or dynamically.

On-Demand Data Retrieval and Configuration

For the administrator to check the system's health in real-time, equipment and network data must be available at any time. This information, combined with the ability to re-configure routing or firewall rules on request, makes it an essential feature for preventing failures. This way, with little human interaction, the system provides failover capabilities as well as preventive measures.

Dynamically Configure Routing and Failover Techniques

The system configures every device without needing an administrator to implement routing policies more efficiently. This configuration initializes the solution but can also be triggered by analyzing the monitoring data mentioned above.

The solution can also detect network failures through said data and, with this information, guarantee failover in the scenario. It can perceive the malfunction in devices and automatically configure the flow control infrastructure to adapt to that malfunction in order not to lose any traffic and guarantee HA.

Attack Prevention

Besides adapting to failures, the system offers a way of preventing malfunctioning through manual control packet filtering rules. The ability to block and grant specific sources of traffic while the solution is operating makes use of the data provided through monitoring. It is crucial to prevent DDoS and other attacks.

3.1.2 Requirements

Flow Control Infrastructure

The core equipment in which the system will act is the security equipment. These can be both VMs or hardware to assure routing capabilities between other machines and still be able to mangle traffic packets. For this to happen, the equipment must be configured accordingly to the system requirements, including IPs, routing tables and interface names. Hence, these devices will be the main target of the system's functionalities.

Connectivity and Access

To supply the system with the ability to act and test features, it must have connectivity to the different devices of the traffic control infrastructure. Moreover, an administration VLAN is required to separate management traffic from data and grant exclusive access to the system admin.

Targeted Zones

For the system to configure the security equipment, it requires a definition of the zones to which it aims to provide HA capability to. While the system itself does not communicate with devices in those networks, it is responsible for configuring routing policies, packet filtering and mangling rules in the infrastructure.

Host Device

For the system to run correctly, a remote machine shall be configured to host the software and maintain connections with the control infrastructure to retrieve data and act accordingly to its functionalities.

A separate admin network must be configured for this connection to exist so that the system can operate the equipment through a different channel and thus not mix the control traffic with the data from other networks.

3.2 Implementation

This dissertation describes an approach to a system with the requirements and functionalities mentioned previously. The system designed to do so is described in the following subsections.

3.2.1 Infrastructure

The solution developed is hosted in a machine with connectivity to the security equipment, which must be detailed in a configuration file before the system startup, as shown in listings 3.1 and 3.2. Such equipment consists of devices with load-balancing capabilities and others dedicated to filtering packets, thus implementing On-Site load-balancers and firewalls. This method allows the administrator to provide all the data required by the system without requiring subsequent configuration. This relation is depicted in figure 3.2.

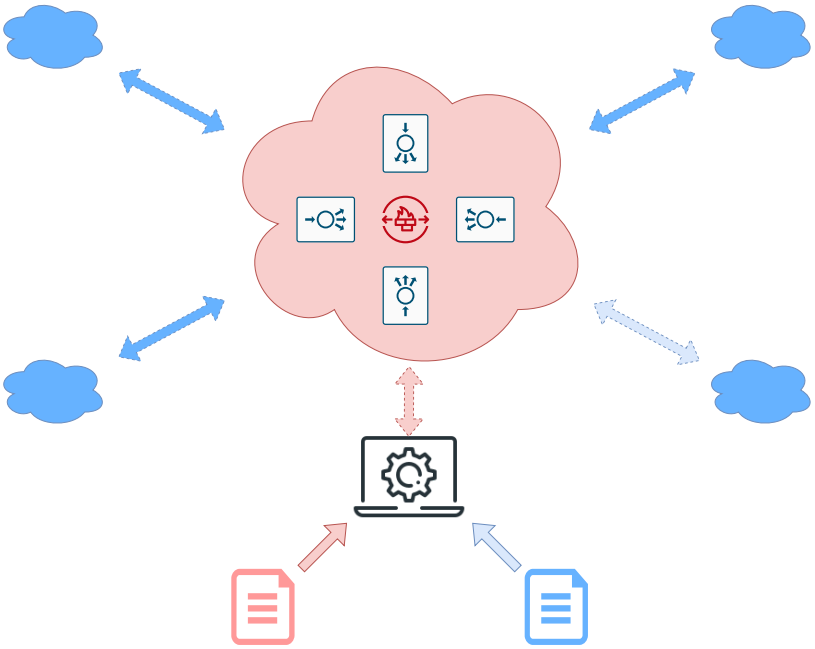


Figure 3.2: Startup of the system's connection to the infrastructure

The control infrastructure and the host device form a management-only IP network, whereby all the connections between the security devices and the solution must use the same interface (Ethernet zero) to automate network topology changes and expedite remote

configuration. In addition, it is an easy way to scale the scenario monitored by the HA system since it allows for a quicker configuration of new equipment and integration in the solution.

```
type: "LB",
name: "LB5",
ip: "192.168.10.7",
username: "debian",
password: "debian",
interfaces : {"0" : "100.0.1.1",
              "1": "100.0.11.1",
              "2": "100.0.12.1",
              "gateway": "100.0.1.10"},
zone: "dmz",
internal_networks : ["100.0.0.0"]
```

Listing 3.1: Example definition of a security device

The rest of the networks involved, mainly representing isolated zones or the internet in general, must also be detailed in a configuration file before the solution launch.

```
inside : "192.168.0.0/24",
outside : "200.2.2.0/24",
dmz : "100.0.0.0/16"
```

Listing 3.2: Example definition of a zone

While the physical implementation of each device is a requirement of the system, configuring its operation enters the scope of its features. Every machine is configured based on the rest of the equipment available. Following the configuration files mentioned above, the routing mechanisms are set up to guarantee connectivity with zones and respective interfaces of other equipment, as well as the networks it borders, regarding the load-balancing machines.

Regarding these devices, the LB capability is also assured by the HA system, considering the available firewalls that are or will be implemented in the startup process. Since many DoS attempts originate from a limited or unique traffic source, the distribution of packets by the Load-Balancers was implemented in a manner that different sources of traffic were redirected to distinct firewalls in order to ease identification and rejection of malicious data. These policies rely on the source address and source port of traffic, later building a *hash* that combines both parameters in a unique value that is redirected to a specific firewall. This mechanism is also implemented reversely, utilizing the destination IP address and destination source port to select a firewall that will analyze the traffic responding to a request originated by the protected network or even other security devices.

Due to possible security threats and resource managing policies, the system accounts for the existence of all configured devices, despite their status. This attention enhances the ability to re-integrate devices that were previously malfunctioning and operating correctly later on since the configuration of that device is already detailed in the system.

Concerning firewall devices, their configuration is also made by the monitoring system. Both routing policies and packet filtering rules are implemented on startup, as depicted in figure 3.3. While the first has in mind the rest of the security infrastructure, similar to the load-balancing devices, the implementation of rules works differently. On startup, the firewall equipment accepts every packet, and changes can only happen after a revision by the admin. Even though the system can help identify malicious targets and allows for a very simple blocking of said sources, this action requires administrator handling since automatic traffic control could induce unwanted packet loss and service denial. Thus, the firewall rules are defined via a user-friendly interaction service, which does not require any network knowledge and works with the solution to apply new filtering rules dynamically and instantly.

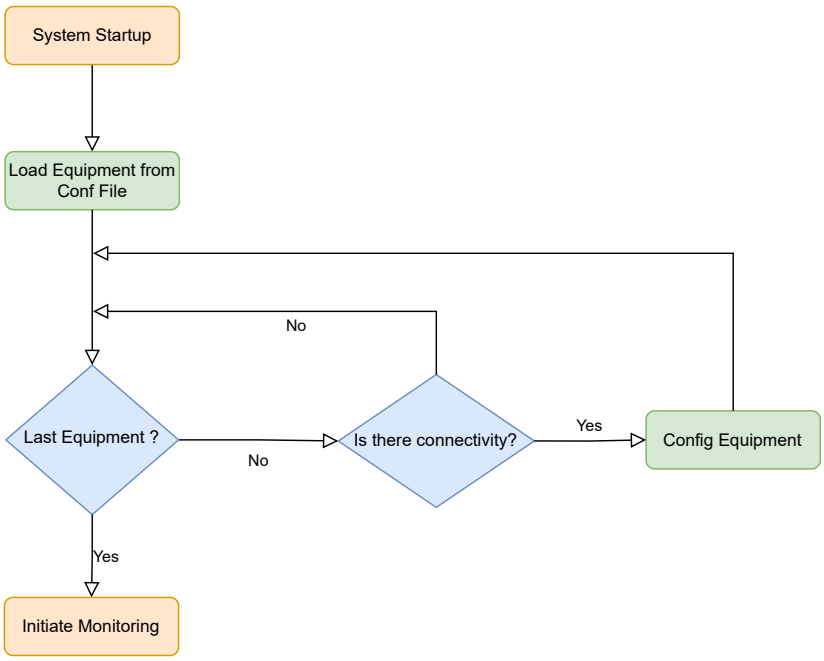


Figure 3.3: System startup flowchart

3.2.2 Monitoring

Threats to a service network can occur at any given time and, particularly regarding DDoS attacks, very quickly. This factor makes it crucial for a system to be regularly supervised so that harmful traffic is prevented, or its effects to be reduced to minimize the network clients' impact.

Regarding the solution described, monitoring the system capabilities is made periodically with the contribution of the infrastructure defined previously. Therefore, the solution must be informed of the state of the equipment to ensure it is functioning.

As such, the quickest way to conclude the operation of the machines is by verifying the

connection to the HA solution, as referenced in figure 3.4. If there is connectivity between the host and a security device, it only shows that the equipment is up, not reporting its actual functioning. However, if said connectivity does not exist, it shows that the device monitored is malfunctioning, shut down, or out of reach by the solution. All of the cases pose a threat to the network. Since all infrastructure equipment is responsible for either load-balancing traffic or filtering packets, its malfunctioning will likely redirect packets to incorrect destinations. This event defeats the load-balancing purpose. Moreover, even if the distribution is correctly done, accepting and outputting traffic that should be filtered and possibly rejected might harm other devices and impact the network users. If the equipment is shut down, it cannot perform the security features intended, which stands as the biggest threat to the network. Finally, even if the device is running and functioning well, if there is no connectivity to the host device, the HA solution cannot act, stripping the possibility to monitor and configure the equipment, ending up as a regular network with a basic level of security.

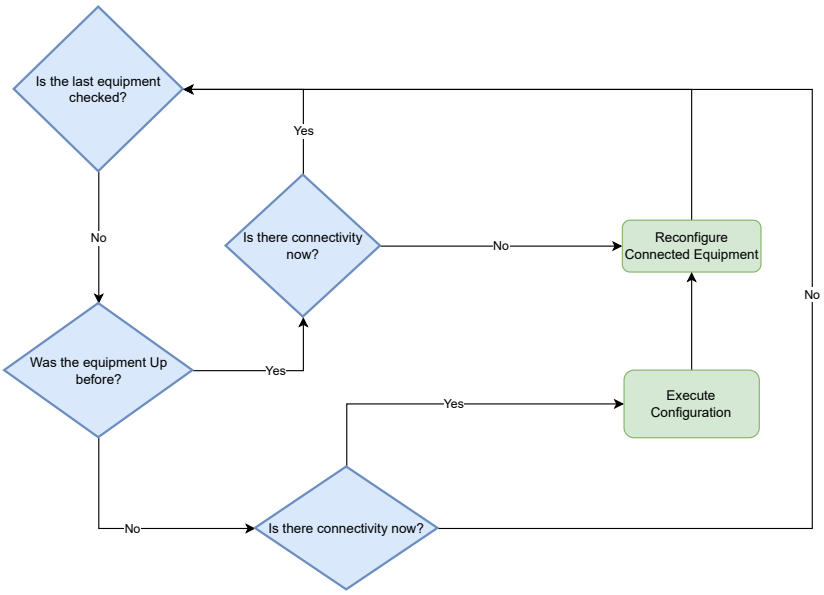


Figure 3.4: Initial monitoring algorithm

If the security devices can communicate with the solution, the latter needs to ensure the correct functioning of the network with different measures. To meet this need, two other data-collecting methods were implemented.

The first method involves querying the device's hardware status (figure 3.5). The most popular DoS attacks attempt to sabotage a network by overwhelming a specific machine, causing its physical components to fail. Therefore, the execution of said attacks reflects on the hardware status of the machine, which can be monitored to act beforehand and prevent failures. Essentially, these features are the core of this dissertation's system in this dissertation. In cases where the connectivity to a given piece of equipment is ensured, periodical requests for the equipment's hardware CPU occupation and memory are made. Should these values exceed a predefined threshold, an alert occurs, and the network administrator is notified.

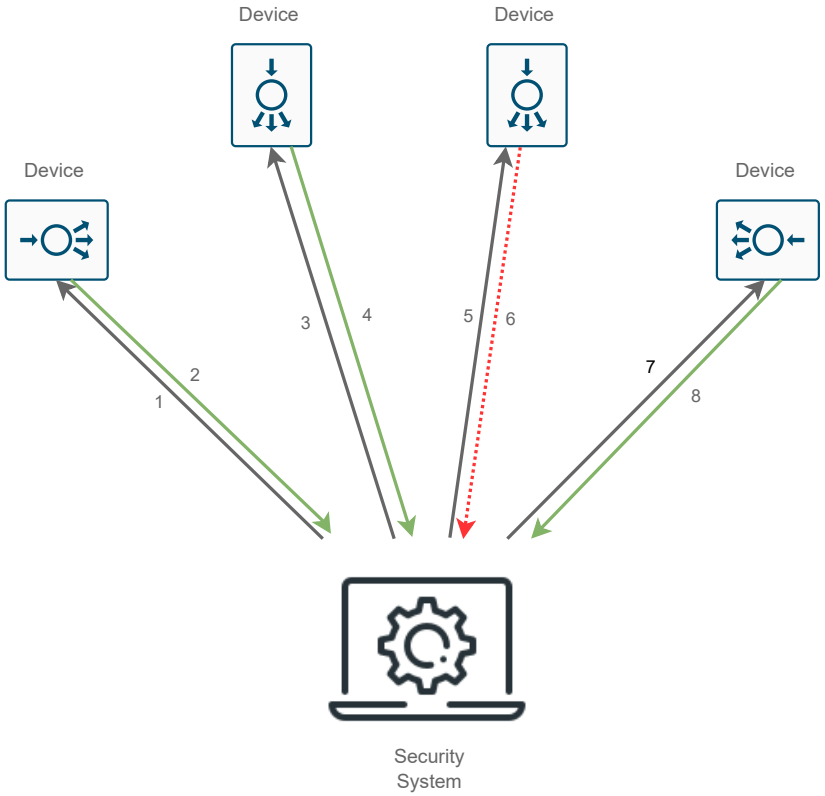


Figure 3.5: Automatic monitoring diagram

The second method relies on the rest of the security infrastructure to provide information regarding the equipment being analyzed. The system obtains data regarding the connectivity between the target device and all the other flow control devices to ensure the target is not isolated, as depicted in figure 3.6. Since every device must be connected physically through different interfaces, this test provides the critical information to detect interface anomalies and attempts to isolate specific devices without the noticing of the monitoring software. This method helps prevent more targeted DDoS threats in case only a specific interface is under attack, which can happen as a more discrete offense, an attempt to disrupt a particular service or part of the network.

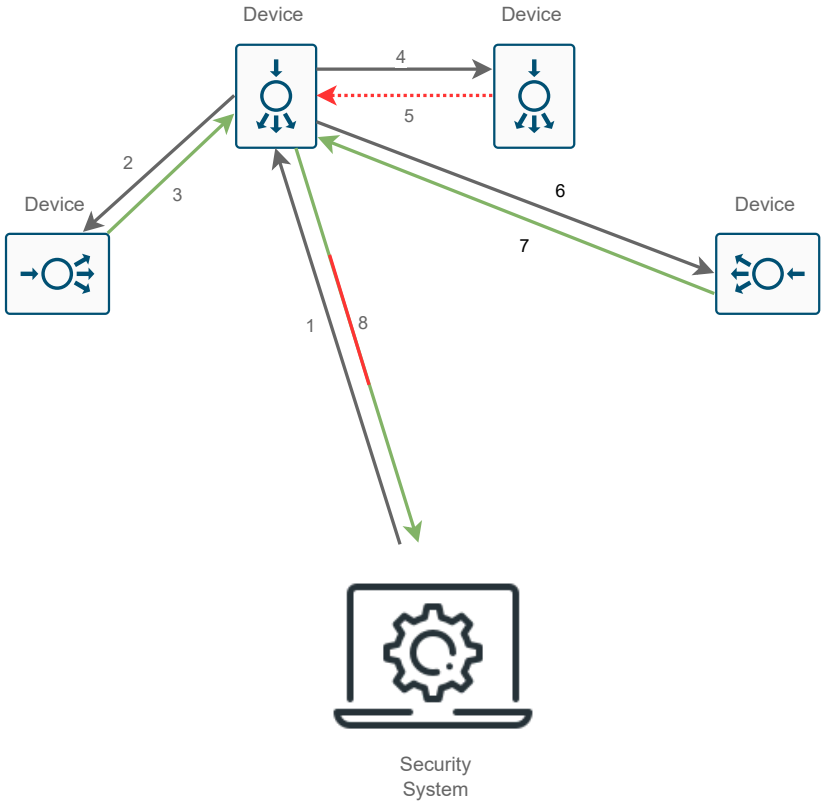


Figure 3.6: Monitoring as querying diagram

3.2.3 Fail Handling

One of the most significant assets of the system described in this document is the ability to adapt a network to changes in topology. In an optimal scenario, these will only happen with human interaction, mainly planned by the network administrator. However, the system's real strength is reacting to unforeseen failures and adapting to cyber attacks.

Regarding the system startup, the data in the configuration files is needed to perform the initial configuration of the security infrastructure, just with the initiation command. Concerning the dynamic configuration of devices, the ability to react to changes is possible thanks to the data gathered through the monitoring mechanisms implemented. Both configuration types are detailed in figure 3.7.

For the connectivity test, more detailed in Chapter 4, the solution adapts the network to changes in device status after the startup. When a device stops or starts working, unlike when the system starts, an algorithm re-configures the equipment connected to it. For changes in Load-Balancers, the system is prompted to discover the now active LB equipment. After deleting all forwarding rules, it implements the routing policies in every firewall to connect to those devices. The packet filtering instructions are preserved to keep preventing more possible failures. When a Firewall changes its state, a similar process happens with Load-Balancers, where all of them discover active firewalls, and routes are updated only to interact with those devices. In addition, the traffic distribution policies are updated to consider the topology change. In these cases of device failure, packet filtering rules are saved by the system so

that when the firewall is back running, the rules are re-established, and all packet filtering equipment is synchronized. Even though the filtering devices are not directly connected, the system is responsible for updating rules so that traffic suffers the same filtering process if distributed to different targets.

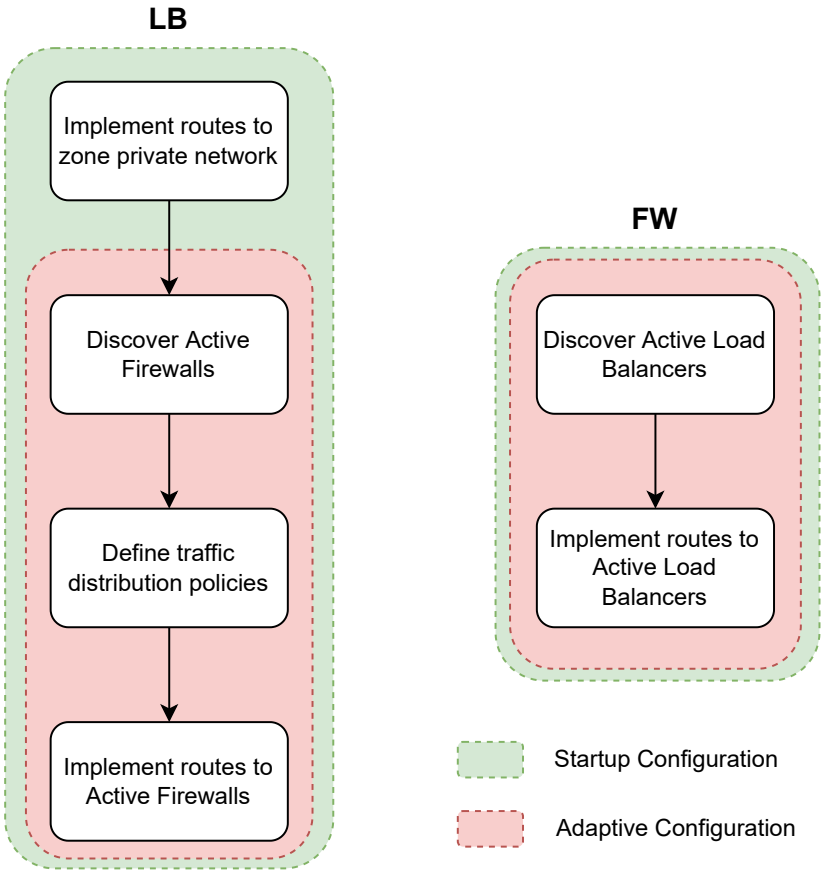


Figure 3.7: Configuration of devices algorithm

In the dynamic configuration of equipment, preventing and adapting to failure must follow a method that allows fast recovery of the previous state of devices. The system complies with these characteristics thanks to the rule-saving mechanism and active device discovery. Every packet filtering instruction implemented in a Firewall is saved by the HA system, so when it reboots or starts functioning after an attack, the flow control is re-established without the need for manual input by the administrator. This feature allows the network to remain protected from previously identified attacks before and after another failure. Moreover, the setup of devices after status changes induces a discovery of active equipment by the system so that only the functioning equipment is considered when configuring a new one. This way, due to possible concurrent equipment failures, there is no need to execute the updates that occurred when the equipment was obsolete. A method was developed to store the commands that disabled machines could not execute due to being inoperative, but more topology configurations could overwrite these commands. Therefore, regarding routing policies, the system provides the information and routes to active machines, reducing configuration changes to the essentials

and minimizing the overall processing of the solution and security infrastructure.

The fact that the solution is responsible for saving the information regarding the state of the devices is essential to protect against DoS attacks. Furthermore, this architecture allows the devices to reduce their workload to the essential networking features, instead of communicating to update their states, which would propagate an attack to other machines in case of failure. This way, striving for maximum protection, the system's host machine should be the most powerful and reliable instead of increasing the power of every security equipment, which would significantly increase the costs of implementing a secure infrastructure.

Another critical aspect of the dynamic configuration method is the network's reliability in the process. When it comes to the consistency of the connections, the security devices' configuration after a topology change follows a thorough method. The settings are individually updated only after other devices are already functioning. As an example, a Firewall failure causes changes in the Load-Balancing policies. However, to guarantee the network remains operating, these changes occur in one Load-Balancer at a time, and more machines are only updated after the previous is functioning correctly. This way, minimal traffic is lost during the topology change process and there is close to no impact on the user.

3.2.4 Admin Interaction

As mentioned in section 3.2.2, the monitoring capabilities allow the solution to react to failures in more flagrant cases. However, network data is constantly being monitored and made available to the system administrator to prevent said failures, which can take measures to alter the traffic flow. Providing this data can be done through alerts of the system or requests. Hence, two different approaches were implemented to arrange the display of information (figure 3.8).

The first approach is through the platform running the security system with an automatic data display. Provided it is started in the host machine, with graphic support, the system shows feedback on its actions during its workflow. Every device configured is linked to a display of data in the host machine with information regarding its configuration and feedback, whether it was successful or not. Periodic warnings of down equipment are also shown, being that if this equipment suffered a change of status, an alert is raised and displayed with more prominent visibility, to raise awareness for the dynamic topology change that is starting or to induce the admin to act accordingly.

The second approach of providing data is through a platform of interaction between the admin and the HA security system, where the first requests the data to analyze. This platform was implemented as a Representational State Transfer (REST) Application Programming Interface (API), available from the system startup. In this fashion, the network manager can exchange data with the system in the shape of requests as long as it is connected to the management VLAN. Said requests include commands to obtain the configuration data, status, and the possibility to re-configure a single device or a variety of equipment. Another key feature of the system, powered by administrator interaction with the API, is the option to manipulate packet filtering to the zones mentioned in the configuration files based on

addresses and ports. The implementation of filtering rules occurs instantly, causing sequential re-configuration of the Firewalls. This packet filtering feature is based on the premise that newer rules are prioritized over older ones since the system's security needs may vary over time. This way, it is possible to allow specific sources of a generally blocked network to access other zones if a port is specified, or the opposite scenario: blocking a specific traffic source even if connected to a previously allowed network.

Finally, it is essential to emphasize the lack of security mechanisms when handling this API, which requires no type of authentication, based on the fact that only users with administrator privileges can connect to the management VLAN, which already narrows the access to this interaction platform.

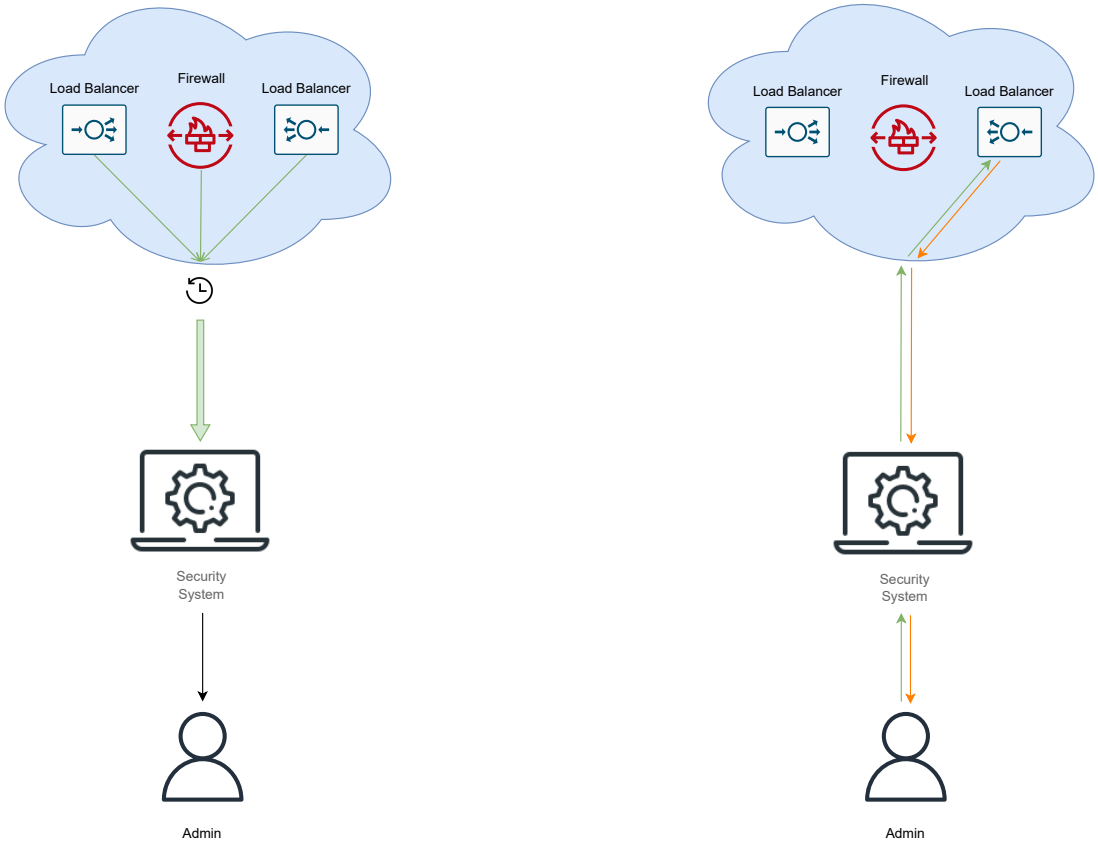


Figure 3.8: Data displaying methods

3.2.5 Tools and Frameworks

The system's core functioning was developed in Python[1], primarily due to the ease of use and the streamlined possibility of connecting to remote devices. While programmed with Python, this connection is executed via Secure Shell (SSH) with the help of *paramiko* library. Every configuration executed remotely by the system uses this library, and requires an opening of an individual session between the host machine and the device to be handled.

[1] `https://www.python.org/`

Said device is planned to run a Linux distribution in view of implementing Linux kernel commands through the HA system, increasing the solution's compatibility and making it more valuable for anyone to integrate it into their network. As such, the system implements NFTables commands in the configuration process. This framework is used due to its high efficiency and easy usability, as described in section 2.2.6.

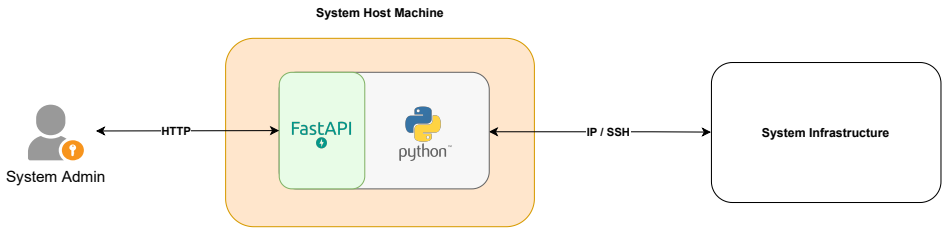


Figure 3.9: Technical Architecture of the High Availability System

For the interaction with the admin, an API was developed using the *FastAPI*[2] framework. This choice was based on the straightforward integration, intuitive interface and easy compatibility with the system's algorithms since both were developed in Python. Furthermore, this tool works as a separate module that provides the admin with various methods depending on the data needed or the action to perform. Both the connection between all frameworks and the methods available to the admin are depicted in figures 3.9 and 3.10.

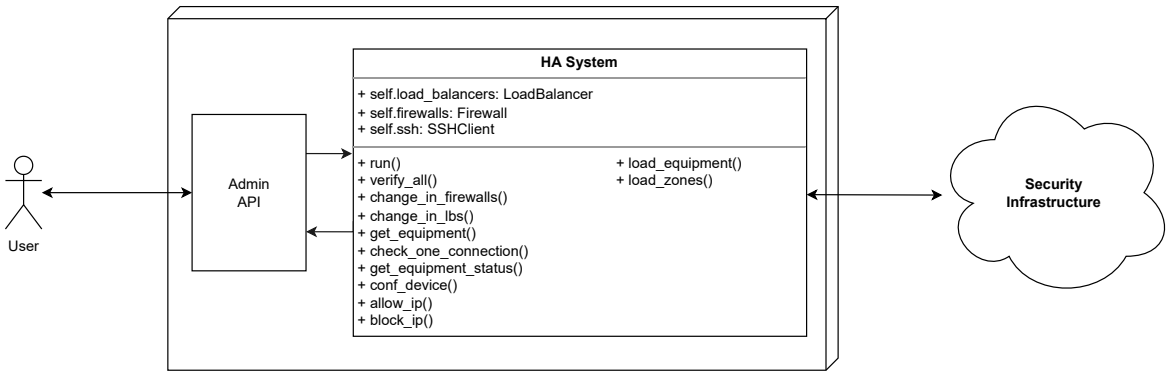


Figure 3.10: User interaction model

[2] `https://fastapi.tiangolo.com/`

CHAPTER 4

Tests and Results

This chapter presents the methods implemented to test the system's functioning, depicting all the tools and frameworks used to implement the solution in a network scenario. The technical architecture is also featured, as well as the results of the system operating in a virtualized environment. As such, a scenario was built in which one could easily integrate the HA system being developed, containing three different zones hosting network equipment, Firewall and Load-Balancing machines and a host machine to run the system.

4.1 Testing Scenario

To test the system's features there was a need of integrating it into a realistic network environment, demonstrated in figure 4.1. This should be able to provide a traffic flow similar to what can be expected in a domestic or corporate network, be it regular communication or harmful data transfers. For every zone, network or test depicted in the next sections, all IP addresses are fictional and are not to be compared to what they represent in the real world.

4.1.1 Test Network

The modern-day DDoS attacks can target any kind of equipment, from Layer 2 switches to *Datacenter* servers. Thus, assuming different levels of security and access control, three zones were defined:

- **Internal Network** - containing the small work equipment, like laptops, smartphones, and other IoT devices. It can either represent a domestic network - for ordinary users and all personal computers and home equipment - or part of a company network - including work or entertainment equipment, personnel and network devices.
- **Datacenter Network or Demilitarized Zone (DMZ)** - featuring equipment that needs external access and is not directly connected to the private network to assure extra security. This is most common in scenarios where the system is integrated into a corporate environment.

- **Internet** - representing everything that the system client can access outside their protected network. This includes all entertainment services, social networks, platforms, websites and everything that requires an internet connection and is not hosted in the client's network.

Besides the defined zones, another subnetwork was established between the security infrastructure and the host machine. This subnet acts as the Management VLAN and allows for the security equipment to connect with the HA system. Said VLAN covers all the traffic control infrastructure, whose devices will be monitored and be subject to handling by the system. The implemented devices were two central Firewalls to analyze packets and filter data flowing between the three zones defined, and six Load-Balancers, two for each connection between a zone and the Firewalls. This infrastructure establishes a total of twelve connections between all of its eight devices, all of them peer-to-peer from a Load-Balancing machine to a Firewall.

It is important to note that the Firewalls could be classified as Stateless. The packet filtering rules implemented with the equipment analyze individual packet data, having little consideration for the flow and are not connected between themselves, therefore not intentionally sharing any state or similar configuration. This method was selected aiming to reduce the impact of a possible DDoS attack on the network, since the overwhelming of a Firewall could lead to the failure of the other, provided the two machines were connected and shared rules and state information.

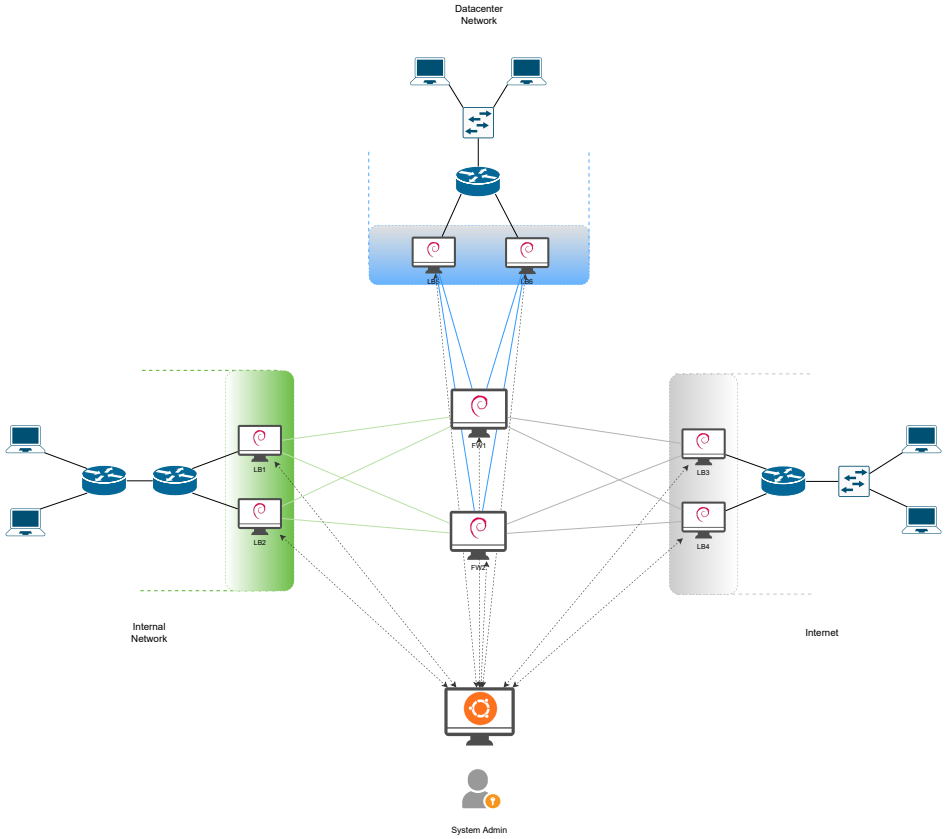


Figure 4.1: Supportive network implemented to test the system

4.1.2 Tools and Frameworks

The scenario was implemented virtually, using *Graphical Network Simulator-3 (GNS3)*. The choice of designing a virtual testing solution was mainly based on the straightforward deployment and configuration, added to the cost savings and easy maintenance. The connection between virtual machines is also via virtualized cable connections and the machine configuration is planned in a hardware-compatible way. Meaning that if the system is integrated with a functioning network with dedicated hardware, the functionalities are preserved. Even though the system was only tested in virtualized environments, the performance should be identical.

Regarding equipment that is part of a zone, every device is implemented using Cisco Internetwork Operating System (Cisco IOS) appliances. Some of these devices exist in the network to generate traffic, being able to simulate both regular and harmful users' behavior. The other devices are installed as switches and routers to make sure there is connectivity between the zones. These only have routing mechanisms to ensure traffic flow inside the network and with the security infrastructure, for they are network dependent and thus are not included in the scope of the HA system.

Concerning security infrastructure equipment, the Firewall and Load-Balancing capabilities were implemented through software running on Linux-based machines. The distribution used was *Debian 11*[1] in view of using Linux kernel commands and having a lighter desktop

[1] `https://www.debian.org/index.pt.html`

environment that allows configuration of network features. Even though possible, packet filtering is a less powerful tool in these devices. The traffic distribution option, while effective, doesn't allow for easy implementation of more complex distribution rules and packet marking. Moreover, Linux kernel commands stand identical among all Linux distributions, which allows for a wider variety of users to implement this solution in their networks, using developer-friendly OSs. This ensures a OSS character to the system developed.

Finally, the host machine is running an image of *Ubuntu 22*[2], this time to allow for easy code development and connectivity with the security devices. As mentioned in section 3.2.5 the core of the systems' functioning was developed in *Python* due to the familiar syntax and easy implementation of remote communication mechanisms.

Since the host machine is a physical appliance, the connection between the system and the infrastructure is also virtualized. A specific network adapter of the host was configured to connect with the virtualization environment and later represented in the network as a cloud, which itself connects to the security devices using their virtual adapters. As mentioned above, the systems' functionalities would still be available in a hardware-only implementation, since these adapters represent the Ethernet ports of the equipment.

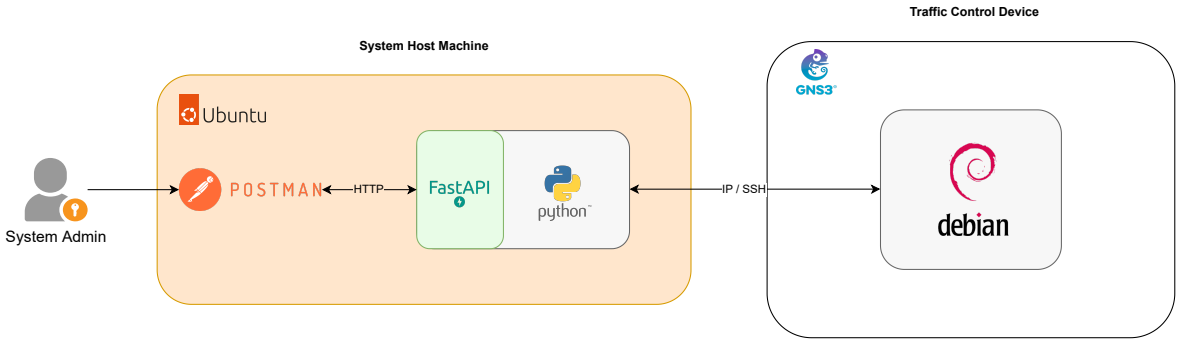


Figure 4.2: Technical Architecture of the testing scenario

The interaction with the admin was done through web browsers and *Postman*[3] to execute Hypertext Transfer Protocol (HTTP) requests to the API. These requests trigger algorithms in the system that handled security infrastructure equipment and return feedback on these actions, as depicted in figure 4.2.

4.2 Functional Tests

To assess the system's features, the evaluation was made through functional tests. Performance, usability or compatibility tests are not in the scope of this dissertation.

With the network scenario referenced in the previous section, the developed tests to assure the solution were defined as follows:

1. **Request device's status, connections and execute re-configuration** - Guaranteeing the possibility of retrieving data on-demand;

[2] `https://ubuntu.com/`
[3] `https://www.postman.com/`

2. **Turn off and reactivation of devices after system startup** - Testing the configuration process adapting to topology changes;
3. **Block specific IP address and allow specific address and port combination access to zones** - Verifying the capability of blocking specific sources while the system is operating;
4. **Device and Zone Addition and Removal** - To verify the ability of the system to work accordingly with a different topology;

All the tests were executed by the author of this dissertation through the system's host machine.

4.3 Execution and Results

4.3.1 Test 1 - Request device's status, connections and execute re-configuration

Testing On-Demand data retrieval and configuration changes, a set of HTTP requests were defined on *Postman* to interact with the system through the API. These requests were executed targeting different devices with different goals, in order to test the scope of the status querying methods. The tests were defined not only to obtain data but also to order the configuration of specific machines, all after the system was initiated. More specifically, considering the testing scenario referenced in section 4.1.1, the requests examined the status of "Firewall 1" and the network's connections, while also prompting a re-configuration of "Load-Balancer 4".

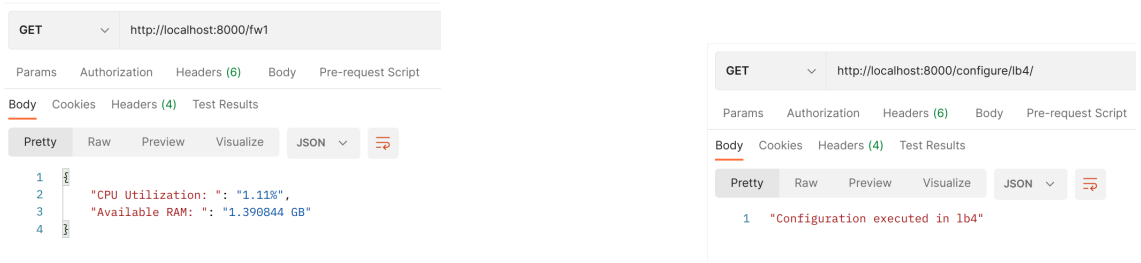


Figure 4.3: Display of Firewall 4 status and example of Load-Balance 4 configuration request

```
LB5 is down
LB6 is down
Configuring LB4 with debian  and debian
ERR: RTNETLINK answers: File exists
ERR: RTNETLINK answers: File exists
ERR: RTNETLINK answers: File exists
INFO:     127.0.0.1:56648 - "GET /configure/lb4 HTTP/1.1" 200 OK
LB5 is down
LB6 is down
LB5 is down
```

Figure 4.4: Load-Balancer 4 configuration feedback

Regarding the configuration requests, it was possible to conclude that these induce more processing from the system since they cause changes in device status. Because the configuration

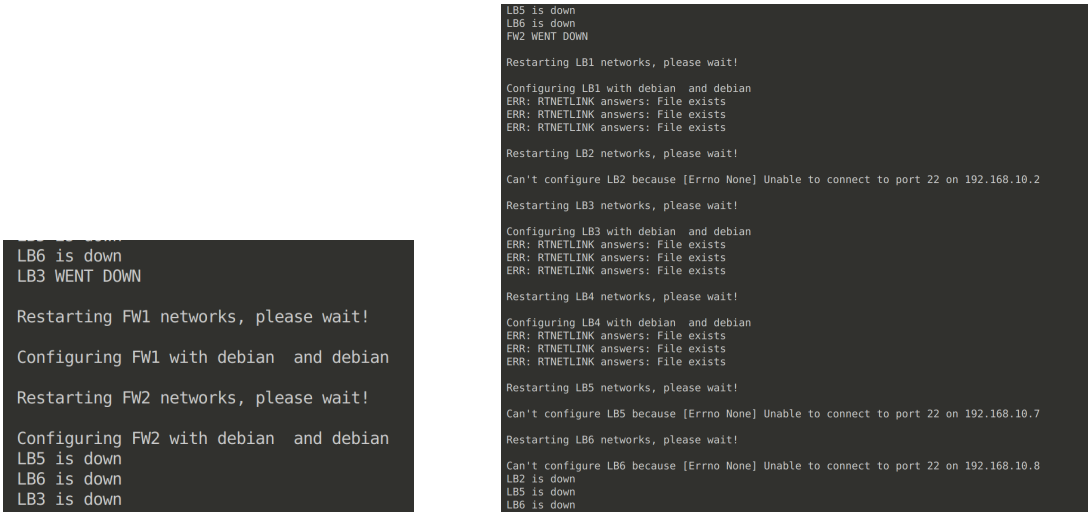
commands alter routing or packet filtering packets, the target equipment can not be reached for brief moments due to the redefinition of its network services. This triggers the system to reconfigure the rest of the infrastructure, even if only for a short period, in order not to lose any traffic.

Nevertheless, the execution of the HTTP requests was successful. All of them produced an immediate effect on the targeted devices and the system, so no other functionalities were questioned. The feedback of the configuration process delivered out of order prints (figures 4.3 and 4.4), related to the terminal software, and warnings related to the duplicated configurations, this because no changes were made to the topology of the network between system startup (initial setup of "Load-Balancer 4") and the manual setup executed afterward. Still, the rules applied overwritten the previously defined, therefore having no negative impact on the functionality of the device.

4.3.2 Test 2 - Turn off and reactivation of devices after system startup

The biggest effect a DDoS attack can cause in a network is a device failure. Regarding functioning, this process is similar to having the equipment turned off. Thus, the second test consists of manually shutting down security devices, simulating sudden failure, in an unknowable way to the system. The target machines used for this test were "Load-Balancer 2" - assigned to distribute traffic from the Internal Network - and "Firewall 2", and connectivity between the Internal Network and the Internet was verified. After testing the reaction of the system to these failures, the devices were made available again, testing the ability to re-integrate active machines.

After shutting down the devices, the system showed a quick reaction to topology changes, shown in figure 4.5. For each device change, the solution performed automatic configuration of the security infrastructure to maintain connectivity between the zones. Even though the adaptation is not instantaneous, the process lasts a short time and provides feedback on the re-configuration of the traffic control equipment.



(a) "Load-Balancer 2" **(b)** "Firewall 2"

Figure 4.5: Display of systems feedback after detecting device failures, prompting re-configuration of the security infrastructure

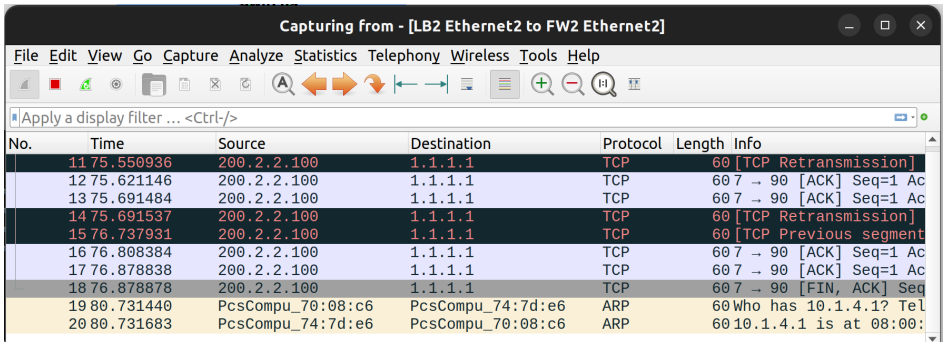


Figure 4.6: Traffic flowing through "Load Balancer 2" and "Firewall 2" before their failure.

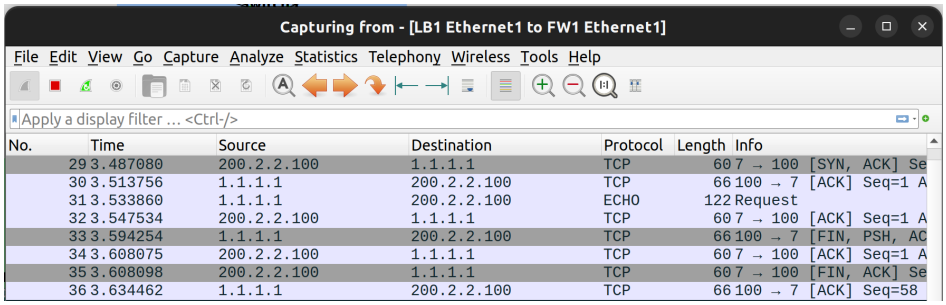


Figure 4.7: Identical traffic flowing through "Load Balancer 1" and "Firewall 1" after changes in the security infrastructure.

Before turning the devices back on, it is possible to conclude that the system correctly implemented new routing and filtering policies for the available equipment in order to maintain connectivity. The example shown in figure 4.6 and figure 4.7 shows that the same traffic flow was established using different machines, adapting to the infrastructure resources, thanks to the system's intervention. Due to the deactivation of one of the Load Balancers assigned to

the Internal Network, being the fictional address *1.1.1.1* covered by that network, both the request and response of the ping were distributed by the available machine, now that "Load Balancer 2" was down. Similarly to the previous tests, LB machines from the Datacenter Network were not functioning, hence the feedback to the admin and the errors of connection to "Load Balancer 5" and "Load Balancer 6".

Subsequently, the devices in failure were made available and the HA system rapidly detected these new changes, re-configuring the rest of the security machines. Similarly to the reaction to failure, the reaction to reactivation was effective. All zones managed to still be able to communicate with each other using all the available devices (figures 4.8 and 4.9). Not only the traffic flow was secured, but also packet filtering rules persisted since the malfunctioning Firewall implemented the previously defined packet filtering rules as soon as it was integrated back into the network. It's safe to say that the system provides failover ability even when multiple pieces of equipment fail simultaneously. Through the feedback provided to the admin, it is possible to conclude that the solution handled concurrent malfunction episodes, detecting all failures and processing each one individually.

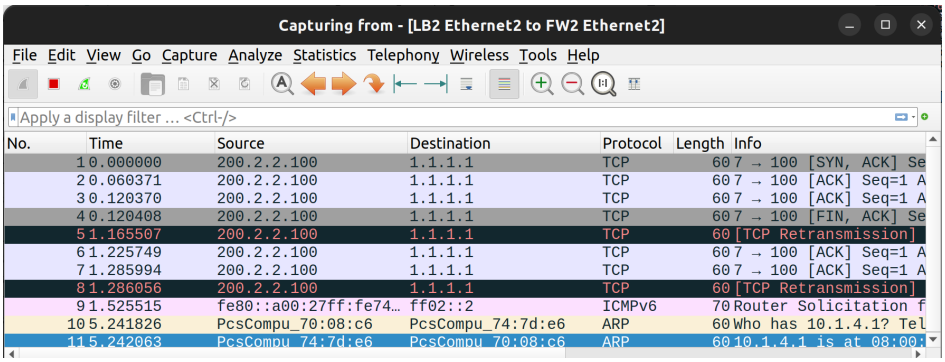


Figure 4.8: Identical traffic flow example using "Load Balancer 2" and "Firewall 2" again after their reactivation.

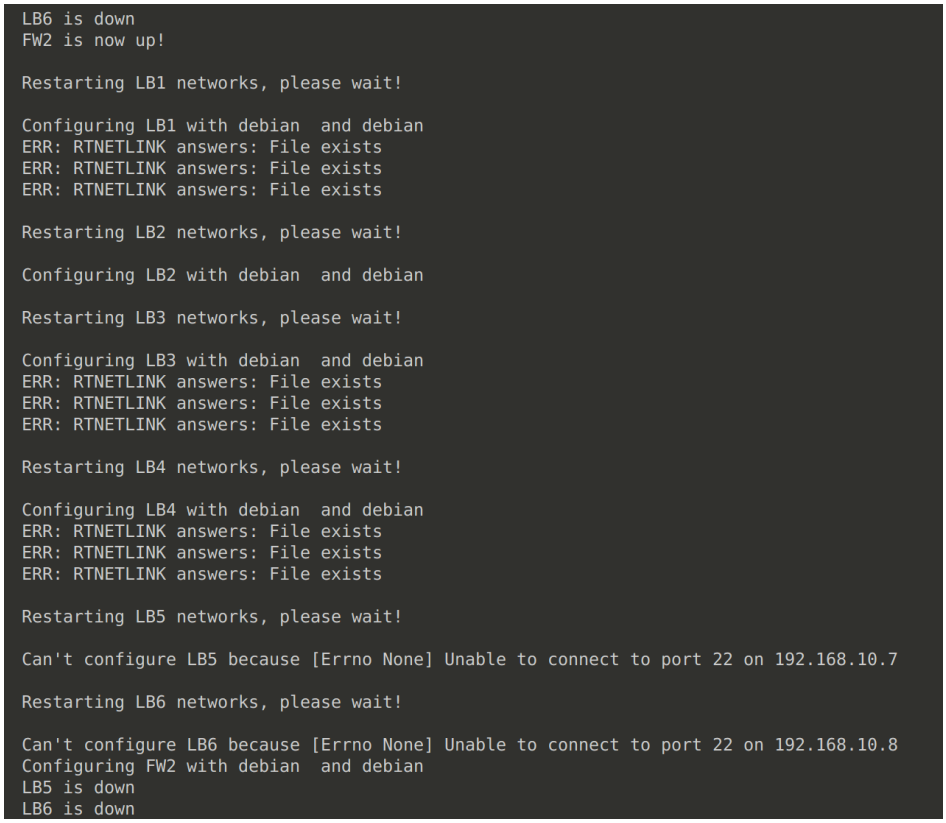


Figure 4.9: Similarly to 4.5, detection of availability of "Firewall 2" and reconfiguration of infrastructure"

4.3.3 Test 3 - Block specific IP address and allow specific IP and port access to zones

Despite the need for admin interaction to filter traffic in the system, this feature is crucial to suppress DoS threats. This way, the connectivity between the two zones was put to test before and after the implementation of filtering rules on the security infrastructure. More specifically, the connectivity between the Internal Network and the Internet was verified on startup, after blocking the IP address to access the same zone (figure 4.10), and after allowing access from a source with the previously blocked IP address but a defined port 22 (figure 4.11), representing the exclusive allowance of SSH traffic.

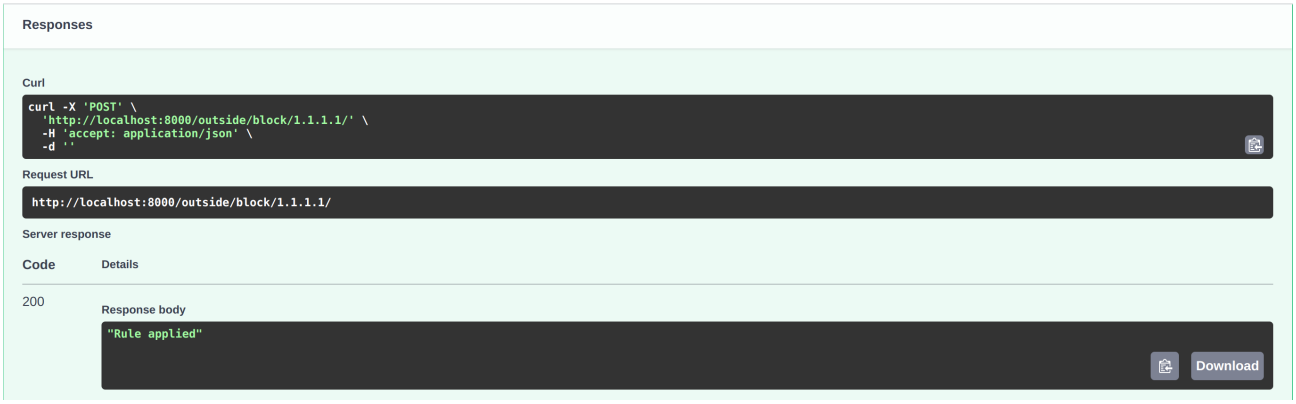


Figure 4.10: Rule for Blocking specific IP from accessing the Internet, applied via web browser

Figure 4.11: Rule allowing access from specific IP and port to the Internet applied via web browser with respective feedback

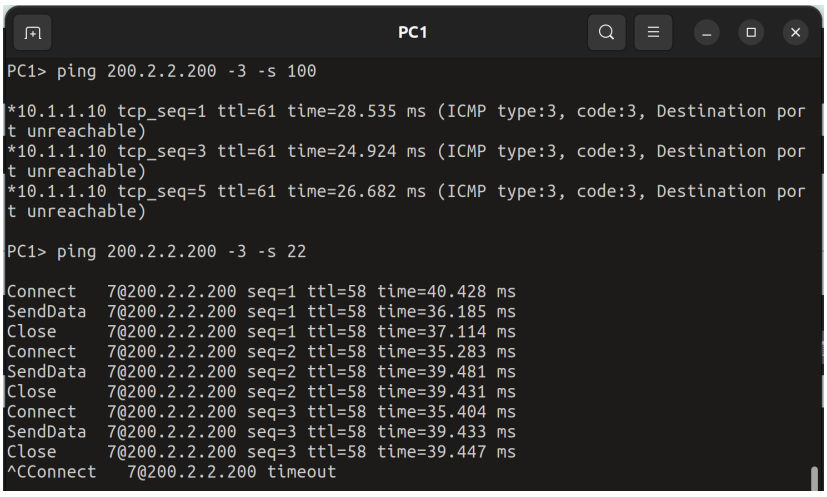


Figure 4.12: Effects of 4.12 rules on traffic flow

The connectivity was tested via TCP pings between VPC covered by the different zones. Immediately after the system startup, communication between these two peers is guaranteed, due to the initial configuration process. Afterward, the IP address 1.1.1.1 was blocked and the same connection was tested. Since the packet filtering rules are applied to all Firewalls covered

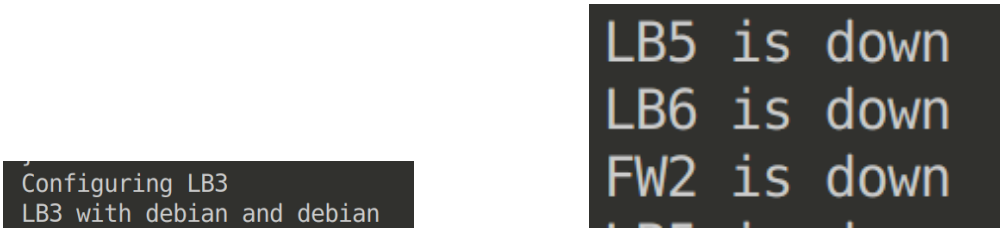
by the system, the connectivity was lost and a "Destination port unreachable" message was displayed after seeing the traffic rejected by the Firewalls. Next, a new rule was implemented, this time allowing a specific combination of IP and port number, being that the address used was the same one that was previously blocked. After the input of the new rule, connectivity was reestablished when using the combination of IP address and port specified, while a combination with any other port number remains blocked, as can be seen in figure 4.12.

4.3.4 Test 4 - Device and Zone Addition and Removal

To execute this test, three network scenarios were planned. The first details two zones (Internal Network and Internet), two Load-Balancers between each zone, the packet filtering equipment, and a single Firewall device. It represents a smaller network from what could be a small company or a domestic structure. The second scenario depicts three zones (Internal Network, Internet and Datacenter Network), two Load-Balancers at the border of each zone and two Firewalls connecting all the Load-Balancers, representing a more complex corporate network with more significant security needs. Finally, the third scenario is similar to the second, but now the devices that distinguish the first two scenarios are deactivated or malfunctioning. This means that network equipment from the Datacenter Network and one of the Firewalls are present, but not operational. All the scenarios can be checked in figure 4.16.

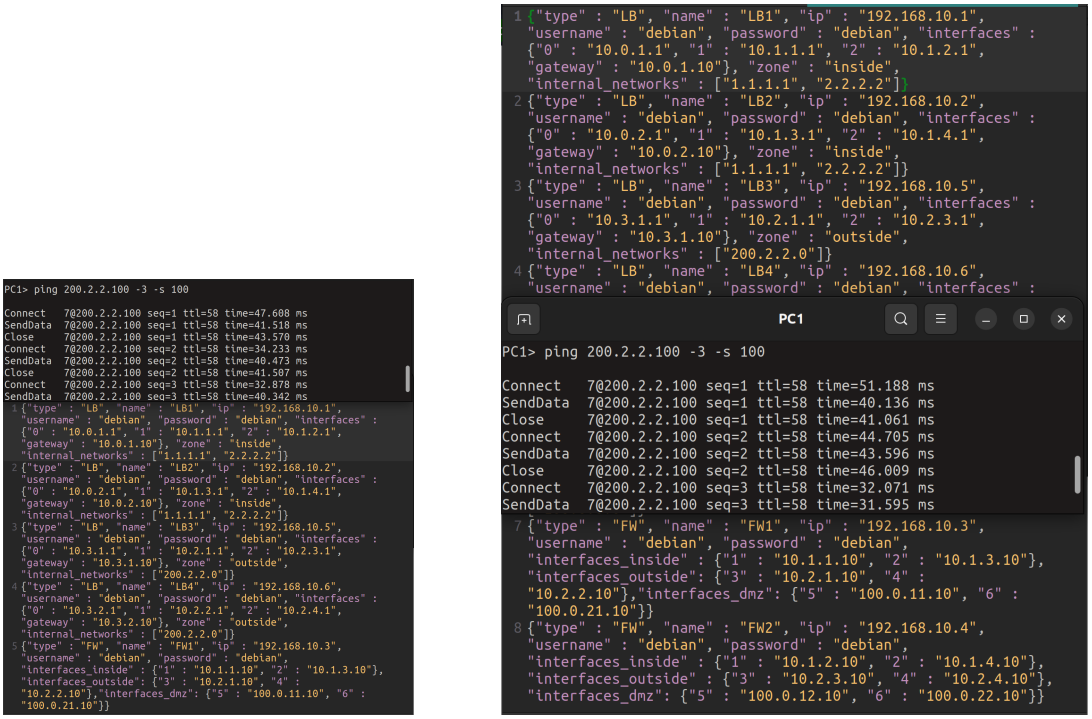
For all scenarios, the system started with virtually no delay, meaning that the process of loading different configuration files did not impact the solution. However, considering the difference in equipment between both networks, the startup time was affected. Since connectivity is ensured only after every device is properly configured, and the configuration process involves testing connectivity, the scenarios with down equipment take longer to achieve a point of correct functioning.

As for the operation of the system, the other functionalities were all present in the three scenarios (figure 4.14). The monitoring of the equipment delivered accurate status changes (figure 4.13) as well as malfunctioning equipment that had been turned on at the startup.



(a) Scenario 1 **(b)** Scenario 3

Figure 4.13: Examples of monitoring while configuring scenarios

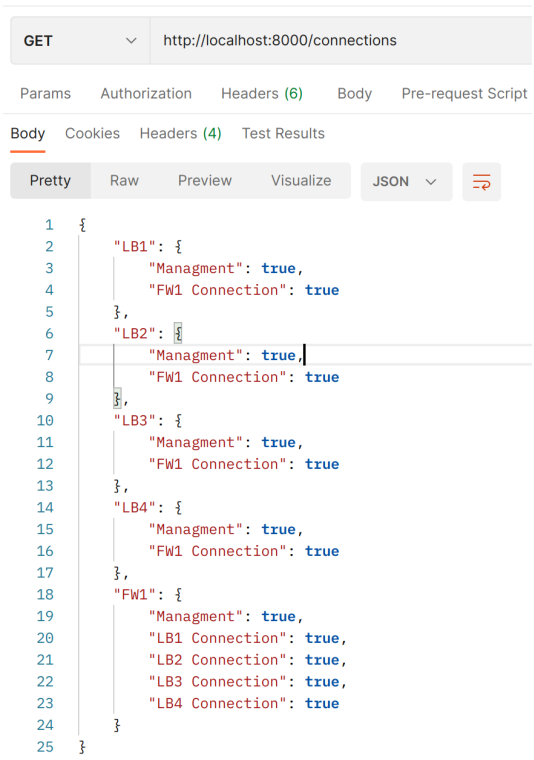


(a) Scenario 1 **(b)** Scenario 3

Figure 4.14: Examples connectivity with different scenario connectivity

The API for retrieving data from the devices was also functional, differing only in the amount of data returned due to the different number of machines, as can be seen in 4.15.

Finally, configuring filtering rules and adaptive configurations remained functional since these features only depend on the target device, obtained via admin input, and the host system whose functioning was not altered.



(a) Scenario 1

(b) Scenario 3

Figure 4.15: Connection data retrieval examples

None of the results presented include Scenario 2 due to the similarity with the other scenarios, since it establishes all the connections of Scenario 3, with all the links functioning well, similarly to Scenario 1.

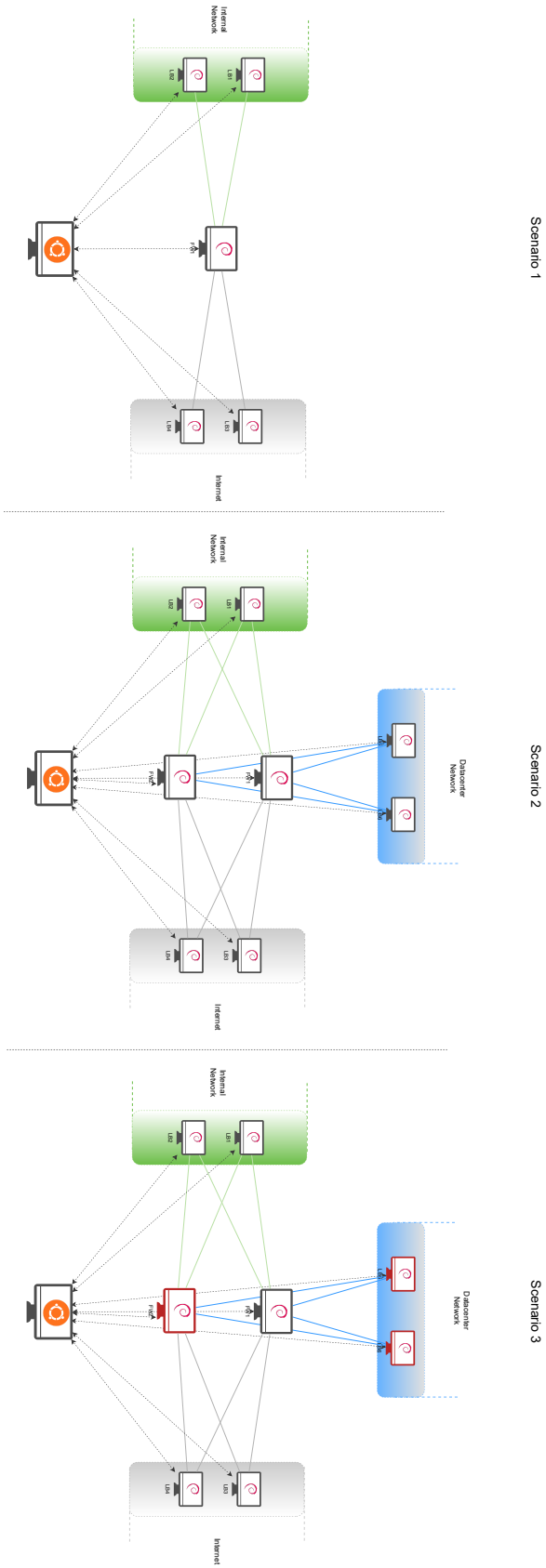


Figure 4.16: Test 4 Implemented Scenarios

The results shown in the previous section represent the correct functioning of the system described in this dissertation. In order to achieve these results, execute the tests mentioned, and take advantage of all the features, the solution must be correctly deployed and used.

4.4.1 Deploy and Startup

Integrating the system into a client's network is a simple task. Regarding the host machine, it is required that Python3, FastAPI and Uvicorn[4] are installed to run the API for admin interaction and the system itself. Moreover, the library paramiko[5] is also required for the connection with the security infrastructure. The startup command, shown in 4.1, is the only action required to initiate both the API and the security system, including automatic configurations and normal workflow.

```
$ sudo python3 main.py
```

Listing 4.1: Startup command

It is important to notice that the system is dependent on the host machine network and its characteristics. While the solution does not require heavy computational power to run, the network connection to security devices must be ensured, as well as permission and configuration for remote access. Said configuration process was not included in the document since it is beyond the scope of this dissertation.

4.4.2 Usage

Much like the deployment process, taking advantage of the HA system is also quite simple. Most of the benefits come from automatic monitoring and configuration, thus not requiring any interaction or knowledge of the system itself. The feedback on the configurations is automatically displayed in the terminal used to start up the solution. To use the features that require admin interaction, the command (4.1) initiates a platform running on port 8000 of the host system and different actions can be put in motion through the same link via the web browser or an API platform like Postman. All the manual actions are described within Section 3.1, but can also be explored in detail at *http://<localhost_ip>:8000/docs*.

Throughout the system's functioning, configuration errors or alerts will be displayed immediately so that the admin can take action on the remote or host machine, depending on the alert. Said errors will not impede the correct operation of the solution, since failures in the security infrastructure are expected and their handling is one of the system's main features. As for incorrect usage of the API, non-existing routes will display an error as feedback of the admin interaction, but this will not impact the monitoring and failover capabilities. Finally, it is important to highlight the impact of the filtering traffic ability. Depending on the admin interaction, this action can cause connectivity issues between the host system and the security infrastructure, which may strip the system of its functionalities by refusing its connection to the filtering devices.

[4] https://www.uvicorn.org/
[5] https://www.paramiko.org/

CHAPTER 5

Conclusion and Future Work

5.1 Conclusion

With the current ease of access to information technologies, more and more cyber attacks can be expected. The increase of Information Technologies (IT) solutions used and the digitalization of everyday life aspects makes the Internet a very valuable place, where one can gain access to privileged information, even digital assets or just cause harm to the competition, striving for personal growth. This makes it essential to have defense mechanisms, no matter the dimension of the protected environment. Meaning, domestic network appliances and small corporate networks must integrate defense mechanisms, which can be costly and hard to manage.

The study conducted in this document allowed the development of a solution that can help defend a network from DDoS attacks while striving to be an Open-Source system. This not only allows for more customization by the end user but also permits a significant cost reduction compared to the available commercial solutions at the time of writing. The architecture of the product was designed to be light and developer-friendly, using mostly Linux kernel commands, familiar to the IT community, while allowing said developers to tweak the functionalities to their liking. The system is easy to implement out-of-the-box, which offers network security features to users with little knowledge of the area.

Regarding functionalities, the HA system has proven to be a competent automatic tool for handling network monitoring and applying failover techniques. It provides DDoS protection with a short learning curve and little configuration, being simple to startup and having an intuitive API for interacting with the network manager.

The tests developed revealed the ability of the system to perform correctly in ordinary cases. Although, due to limitations in testing equipment, it was not possible to test the reaction of the system to an actual DoS/DDoS attack. However, since the main end goal of these attacks is to cause failure in the service by sabotaging devices, one can reproduce the same effect with spontaneous shutdowns, which the solution dealt well with. The testing scenario, while limiting possible tests, allowed for faster implementation and management of a

network to support the system. In general, it is possible to conclude that the research done led to a solution capable of handling modern-day problems, widely available and budget-friendly.

5.2 Future Work

While it is true that the solution developed fulfills the functionalities planned, it does not offer full automatic protection to DDoS attacks since the packet filtering rules are applied via admin interaction. The human analysis can not only be faulty but also very time-consuming, which suggests that traffic flow analysis algorithms could be used to benefit the client. Said algorithms can be simple conditions for blocking traffic, like using the status of hardware resources of the filtering equipment and linking it to the volume of packets per source. Additionally, a machine learning method could be applied to detect malicious network data, classify the packets or perform anomaly detection.

The system was tested in virtual environments only, which disregards the possibility of physical problems and limits the performance of the security infrastructure. While other issues might occur, like electric failures or manual sabotage, specialized hardware outperforms virtual machines running on a server or personal computer. In the future, implementing the solution in a network with dedicated equipment for the security infrastructure could prove to be more reliable, provided that these devices would handle more complex attacks and bigger volumes of malicious traffic. Furthermore, if integrated with specialized hardware, the system could become more easily deployed with a connection detection mechanism and DHCP, so that the networks between security equipment would be automatically set up by the system, and require less initial configuration by the admin.

Finally, in order to maximize the potential of the software developed, performance tests should be implemented to conclude the delay in configurations and the system's reactions. Lastly, code optimization or automation frameworks can be implemented in order to increase the efficiency of the solution, as well as help its possible integration as a cloud service.

References

[1] S. Grand-Clement, "Digital learning: Education and skills in the digital age.," *RAND Europe*, 2017.

[2] K. Hernandez and T. Roberts, "Leaving no one behind in a digital world acknowledgements suggested citation copyright," 2018.

[3] *Check point research: Third quarter of 2022 reveals increase in cyberattacks and unexpected developments in global trends - check point software.* [Online]. Available: https://blog.checkpoint.com/2022/10/26/third-quarter-of-2022-reveals-increase-in-cyberattacks/ (visited on 10/10/2022).

[4] *Alarming cyber statistics for mid-year 2022 that you need to know.* [Online]. Available: https://www.forbes.com/sites/chuckbrooks/2022/06/03/alarming-cyber-statistics-for-mid-year-2022-that-you-need-to-know/?sh=130639b7864a (visited on 09/20/2022).

[5] *Reliability: An imperative for digital business.* [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2021/08/16/reliability-an-imperative-for-digital-business/?sh=2bcf1ed4b835 (visited on 06/13/2022).

[6] *How the internet works, part i - the internet infrastructure | vahid dejwakh.* [Online]. Available: https://vahid.blog/post/2020-12-15-how-the-internet-works-part-i-infrastructure/ (visited on 03/21/2022).

[7] *Creating and using vlans for uk offices, data centres - activereach.* [Online]. Available: https://activereach.net/support/knowledge-base/connectivity-networking/creating-and-using-vlans/ (visited on 04/02/2022).

[8] N. Agrawal and S. Tapaswi, "Defense mechanisms against ddos attacks in a cloud computing environment: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3769–3795, 2019. DOI: 10.1109/COMST.2019.2934468.

[9] A. Munshi, N. A. Alqarni, and N. Abdullah Almalki, "Ddos attack on iot devices," in *2020 3rd International Conference on Computer Applications & Information Security (ICCAIS)*, 2020, pp. 1–5. DOI: 10.1109/ICCAIS48893.2020.9096818.

[10] *Syn/dos/ddos protection - routeros - mikrotik documentation.* [Online]. Available: https://help.mikrotik.com/docs/pages/viewpage.action?pageId=28606504 (visited on 09/13/2022).

[11] A. Agarwal, G. Manisha, R. N. Milind, and S. S. Shylaja, "Performance analysis of cloud based load balancing techniques," in *2014 International Conference on Parallel, Distributed and Grid Computing*, 2014, pp. 49–52. DOI: 10.1109/PDGC.2014.7030714.

[12] A. Johansson, J. Zaxmy, and T. Fischer, "Http load balancing performance evaluation of haproxy, nginx, traefik and envoy with the round-robin algorithm," 2022. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1678660&dswid=-6489.

[13] *What is load balancing and how does it work?* [Online]. Available: https://phoenixnap.com/kb/load-balancing (visited on 05/28/2022).

[14] G. Liyanage and S. Fernando, "Firewall model for cloud computing," in *2013 IEEE 8th International Conference on Industrial and Information Systems*, 2013, pp. 86–91. DOI: 10.1109/ICIInfS.2013.6731960.

[15] S. Chaitra and R. Sharma, "Integration of software router with wi-fi for enhanced security," in *2017 IEEE 7th International Advance Computing Conference (IACC)*, 2017, pp. 33–36. DOI: 10.1109/IACC.2017.0022.

[16] S. Staroletov, "Software architecture for an intelligent firewall based on linux netfilter," in *2022 25th Conference on Innovation in Clouds, Internet and Networks (ICIN)*, 2022, pp. 160–162. DOI: 10.1109/ICIN53892.2022.9758121.

[17] *Iptables processing flowchart (updated often) – phil hagen's scratch pad.* [Online]. Available: https://stuffphilwrites.com/2014/09/iptables-processing-flowchart/ (visited on 06/20/2022).

[18] C. Diekmann, L. Hupel, J. Michaelis, · . M. Haslbeck, G. Carle, and M. Haslbeck, "Verified iptables firewall analysis and verification," *Journal of Automated Reasoning*, vol. 61, pp. 191–242, 2018. DOI: 10.1007/s10817-017-9445-1. [Online]. Available: https://doi.org/10.1007/s10817-017-9445-1.

[19] S. Suehring, *Linux firewalls: Enhancing security with nftables and beyond.* Addison-Wesley Professional, 2015.

[20] D. Melkov, A. Šaltis, and Š. Paulikas, "Performance testing of linux firewalls," in *2020 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, 2020, pp. 1–4. DOI: 10.1109/eStream50540.2020.9108868.

[21] P. Likhar and R. Shankar Yadav, "Impacts of replace venerable iptables and embrace nftables in a new futuristic linux firewall framework," in *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, 2021, pp. 1735–1742. DOI: 10.1109/ICCMC51019.2021.9418298.

[22] P. Blazek, T. Gerlich, Z. Martinasek, and J. Frolka, "Comparison of linux filtering tools for mitigation of ddos attacks," in *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, 2018, pp. 1–5. DOI: 10.1109/TSP.2018.8441309.

[23] J. A. Donenfeld, "Wireguard: Next generation kernel network tunnel," 2017. [Online]. Available: www.wireguard.com.