TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technologies

Dominik Zoltan Kovacs 202092IVSB

Automated Vulnerability Scanning of the Network Segments of an Internet Service Provider

Bachelor's thesis

Supervisor: Tauseef Ahmed

PhD

Co-Supervisor: József Tanárki

BSc

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Dominik Zoltan Kovacs 202092IVSB

Haavatavuste automaatotsing internetiteenuse pakkuja võrgusegmentidest

Bakalaureusetöö

Juhendaja: Tauseef Ahmed

PhD

Kaasjuhendaja: József Tanárki

BSc

Tallinn 2023

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Dominik Zoltan Kovacs

29.12.2022

Abstract

A network automation solution is described in this paper to facilitate network auditing. Since an Internet Service Provider's infrastructure can be enormous and it is difficult to validate each access list manually on each device, a collection of processes was created to automate the handling of infrastructure access lists in an Internet Service Provider's infrastructure. Additionally, the research suggests a semi-automated solution based on closed loop automation, which allows a process controller to understand feedbacks. As a result, the more feedback a controller receives, the more intelligent the system will be.

Due to its not fully-automated nature, this thesis provides the basis for an open-source semi-automated system. This thesis aims to answer questions such as, what is the purpose of a semi-automated system, in what environment do they help, what are their key components and procedures, and finally, what role does it play in network auditing. Despite implementing the solution into this system, the author tried to emphasize the importance of network automation and closed loop automation. This thesis is written in English and is 47 pages long, including 7 chapters, 17 figures and 4 tables.

Annotatsioon

Käesolevas bakalaureusetöös käsitletakse võrguauditi korraldamiseks mõeldud automatiseerimislahendust. Kuna Interneti teenusepakkuja taristu võib olla väga laiaulatuslik ja iga juurdepääsunimistu käsitsi valideerimine on keeruline, loodi teenusepakkuja taristu juurdepääsunimistute haldamiseks rida protsesse. Lisaks sellele pakutakse välja suletud tsüklil põhinev poolautomaatne lahendus tagasiside arvessevõtmiseks. Seega on loodav lahendus seda arukam, mida enam tagasisidet laekub.

Käesolevas bakalaureusetöö pakutakse välja avatud lähtekoodi põhine poolautomaatne süsteem. Bakalaureusetöö püüab muuhulgas vastata järgnevateke küsimustele: mis on poolautomaatse süsteemi otstarve, millistes oludes neist kasu on, millised on nende peamised osad ja protseduurid ning viimaks, millist rolli nad võrguauditi juures mängivad. Lisaks lahenduse väljapakkumisele püüdis autor rõhutada võrgu automatiseerimise ja suletud tsükli kasutamise tähtsust.

See bakalaureusetöö on kirjutatud inglise keeles ja sisaldab teksti 47 leheküljel, 7 peatükki, 17 joonist ja 4 tabelit.

List of abbreviations and terms

ACL	Access List
AS	Autonomous System
AVS	Active Vulnerability Scanner
BGP	Border Gateway Protocol
CVE	Common Vulnerabilities and Exposures
DOS	Denial of Service
GUI	Graphical User Interface
HLD	High-Level Design
iACL	Infrastructure Access List
IGP	Interior Gateway Protocol
ISP	Internet Service Provider
LER	Label Edge Router
LDP	Label Discovery Protocol
LSR	Label Switching Router
MPLS	Multiprotocol Label Switching
OSPF	Open Shortest Path First
PVS	Passive Vulnerability Scanner
QoS	Quality of Service
SNMP	Simple Network Management Protocol
SSOT	Single Source of Truth
VRF	Virtual Routing and Forwarding
VPN	Virtual Private Network

Table of Contents

List of Figures

List of tables

1 Introduction

In any country's network infrastructure, Internet Service Providers (ISPs) play a crucial role. Compromising even one device can result in downtime and data leaks. Keeping networking devices secure is essential for ensuring integrity, availability, and confidentiality. Since an ISP's network infrastructure is growing rapidly, it comes with risks. Such risk can be a publicly visible device on the core infrastructure that can be a threat and attack vector for malicious activities if not defended properly. However, there are many ways to defend against threats. These are classified into proactive and reactive approaches, where proactive is about preventing risks before they occur and being prepared for most cases. As part of this thesis, proactive steps will be presented to mitigate risks. Network auditing is part of these proactive steps that consist of three parts: network scanning, vulnerability scanning and vulnerability analysis. Infrastructure Access Lists (iACLs) are one of the most critical security controls that can be implemented in every company. iACL only allows authorized access for infrastructure equipment, therefore adds an extra layer of security for hosts that are publicly visible. In this thesis, an automated way is proposed to test iACLs in a company's infrastructure. To achieve this goal, network scanners are compared and analysed by their performance in various use cases. This document aims to construct and present a system using a popular automation tool, called Ansible.

This chapter will describe the problem, goal, objective, research methodology and the organization of the thesis.

1.1 Problem statement

Main motivation of the thesis was to fulfil a request of an ISP, where the author spent his internship. The request was to help their network audit process, which is essential in every infrastructure. This thesis provides help for the first step of network auditing, which is network scanning and identifying devices, that allows to create a list of assets of the infrastructure and to understand which devices can be attack vectors in the future. An

ACL is an object of managing network traffic that drops or allows packets based on rules, but it is also used for hiding devices on infrastructures. Still, testing ACLs on an immense networks like an ISP can be challenging, since as a result of regular network changes, ACLs must be changed if their functionality in the network will change. The testing of these ACLs on different segments of the network must therefore be automated.

1.2 Goals and objectives

During the internship of the author, the ISP had a request, which was to find a way to test the functionality and correctness of iACL, sometimes also referred as stateless firewalls. This thesis intends to create a process for automatically scanning network segments on the ISP's network. In addition to identifying devices, protocols and open ports, there are side objectives such as comparing network scanners and explaining why Nmap is the best tool to solve this problem.

1.3 Research Methodology

To gain the most appropriate and comprehensive solution for the company's infrastructure, the following steps are followed in the research process, which consists of consultations with engineers, literature review and modelling.

- Understanding the infrastructure

Wide Area Network technologies and each segments of the network of an ISP are very complex. They have a very significant role to play in the network. For this reason, it is important to conduct research to better understand each protocol and technologies used.

- Investigation of the problem.

The problem defined in the problem statement is critical, therefore analysing and properly understanding the risks are essential. Second step of the research is to understand the role of this problem in the ISPs infrastructure and the potential damage it can cause, if the solution is not implemented correctly. Main method is to consult with the engineers.

- Discovering technologies

It is the literature review part of the research, that helps to determine the best solution, by analysing number of technologies based on their performance, accuracy and usability. Additionally, to achieve this, close collaboration with the ISP is required since a decision has to be made based on the difficulty of implementing the system.

- Constructing the system

There must be comprehensive documentation of the solution's development process, which is used as a guide during the development. Besides that, this part of the research is also important in terms of the project's time and cost. The research method is modelling, where a testing environment has to be created to understand the necessary steps of the processes.

- Development of the inventory and the playbooks

The case study of the thesis is the development. The process controller is implemented with the help of Ansible, that is the connection between all events and orchestrates the processes. The scope of the thesis is only to construct the automated process, however an alpha version of the developed components can be found in Appendix 2.

2 Background information

In this section, the author aims to provide enough information to gain a deeper understanding of the current situation and the problem. It is impossible to understand the results of this thesis without understanding ISPs, their networks and roles, configuration management tools, and ACLs.

2.1 Internet Service Providers

According to the United States Cybersecurity & Infrastructure Security Agency, the Communication Sector is a critical infrastructure, as the Information Technology sector plays a vital role in the nation's security, economy and public health and safety. Businesses, governments, academia, and private citizens depend increasingly on IT sector functions. [1]

ISPs are companies that route internet traffic, resolve domain names, and maintain network infrastructure to provide internet access for a fee. [2]

There are different ISP Tiers, and the top of the pyramid is tier 1, like Deutsche Telekom. Few of the key attributes of being a tier 1 ISP as follow, based on External Publication of IDC Information and Data: [3]

- They have access to the entire Internet routing table through peering.
- They have at least one Autonomous System (AS) Number per continent.
- Owning a lease international fibre optic transport.
- They deliver packets between customers and peers.

Tier 2 and tier 3 ISPs do not have these attributes, however they peer with other tier 1 and tier 2 ISPs to have access to almost the entire Internet, so they can provide packet delivery and reliable connections. In all tiers, the common technology used is Border Gateway Protocol (BGP). BGP is used between Autonomous Systems (AS) to communicate host reachability information. Since there can be many valid paths to reach endpoints, external BGP (eBGP) learn routes of external prefixes from other ASes. Internal BGP (iBGP) is then used for propagating routes among routers inside the

AS. It is necessary for routers within an AS to form a full mesh of sessions to be able to announce routes via iBGP. [4]. An example can be seen on Figure 1.

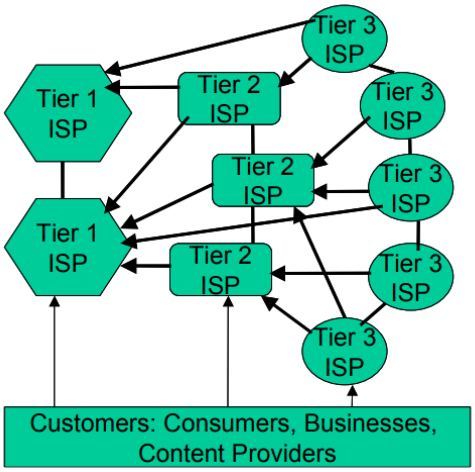

Figure 1: Hierarchy of the Global Internet [3]

2.2 Network Characteristics

There is a representation of a traditional MPLS VPN network in the following figure. Compared to the legacy hub and spoke model, MPLS overcomes many limitations. It allows for the virtualization of routing and forwarding tables and it uses labels to forward packets instead of network addresses. Through this technology, a single piece of hardware can be shared and used by several independent peers at the same time.

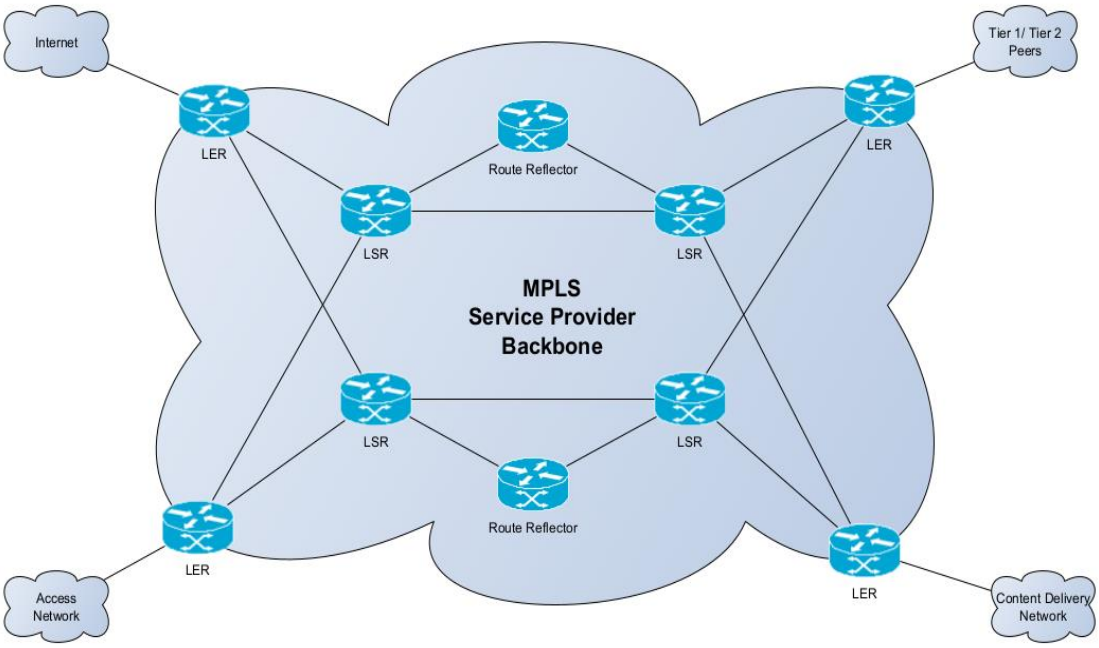

Figure 2: MPLS Service Provider Backbone model

2.2.1 Core Network

In the model, each type of router has a different purpose in the network. By knowing from Cisco's official MPLS VPN introduction documentation [5], Label Switching Router (LSR) is responsible to transport traffic across the MPLS backbone. It does not have any Border Gateway Protocol (BGP) peer or knowledge about the customer routes, which in this case the peering networks. LSR runs Open Shortest Path First (OSPF) protocol as the interior gateway protocol that creates a layer 3 connection between the devices and chooses the best path to forward traffic.

Label Edge Router (LER) is the device that has knowledge of the customer routes and peers with the customer equipment, usually with external BGP. Each LER maintains a Virtual Private Network (VPN) virtual routing and forwarding (VRF) table, that is independent from every other routing table. These VRFs contain the directly connected VPN sites only. When LER receives ingress traffic, it performs classification, assigns a label, and uses the responsible VRF to forward IP packet, where each VRF is mapped to a customer's VPN. After the traffic crosses the backbone, labels from egress traffic must be removed by LER and forward the traffic for the peers using native IP addresses instead of labels. [5]

To make MPLS Backbone scalable, route reflectors are used, which are basically dedicated routers only to announce routes for internal BGP peers. This implementation is important, because without Route Reflectors each LER must be in full mesh to exchange routes. As a result, each LER only needs to be BGP neighbours with the route reflectors. Moreover, Route Reflectors allow LERs to exchange VPN-Ipv4 updates through Multi-Protocol iBGP sessions. As part of the updates, VPN-Ipv4 addresses and labels are located. Additionally, as part of the Multi-Protocol BGP, a route distinguisher is added to every Ipv4 route, which makes VPN-unique addresses also unique in the MPLS core. [5]

Then, by knowing the path from the route reflector, OSPF will have the responsibility to deliver the packets to the correct router. Through OSPF, both LERs and LSRs have BGP next-hop reachability. Labels then are distributed via the label discovery protocol with their corresponding BGP next hops. As the packet travels across the provider

backbone, two labels are used. Label one is used to direct the packet to the appropriate Egress LER. The second label indicates how that egress LER should forward the packet.

Additionally, MPLS VPNs require security measures such as Address Space Separation, Routing Separation, and hiding the MPLS core structure. VPN-Ipv4 addresses can be used to achieve Address Space Separation. For routing separation, LERs must maintain a separate VRF for each VPN connected. [5] The thesis aims to hide MPLS core structure by hiding each VPN's VRF on each LER. Due to the fact that every VPN results in a separate VRF, there is no interference between VPNs on LERs. While MPLS does not reveal unnecessary information to the outside, the infrastructure can contain public IP addresses for several reasons. Since this Tier 2 ISP was established quite some time ago, these reasons are historical in nature. At that time, public and private networks were separated, and public IP addresses were plentiful. Engineers did not considered building a public network using private IP addresses. Additionally, it was not possible to create an outband management network, so it was necessary to create a management backdoor from the side of the internet. It is a huge security risk, however, after years, it would be very challenging to change all the public IP addresses, because so many services and systems rely on them. Using ACLs to hide these addresses from the external network is an easier solution. It is necessary, since by knowing the IP addresses, it is much easier to carry out direct attacks, such as Denial-of-Service attacks on LER and LSR. In order to completely mask the MPLS core, packet filtering and using firewalls are required.

2.2.2 Access Network

In telecommunications, an access network is a user network that connects subscribers to a service provider and, via the carrier network, to other networks. Connections such as Ethernet, Wireless LAN, Fibre Optic and ADSL are options to connect to a service provider. In the presented model on Figure 2, access networks mostly consist of small office home office (SOHO) networks and are considered untrusted.

2.2.3 Content Delivery Network

In Cisco's Annual Internet Report (2018-2023) [6], it is evident that bandwidth demand is increasing, mainly due to video traffic. Additionally, there is a need for convergence,

so that all network services can be delivered over the same network with consistent performance. According to the report, Wi-Fi and cellular speeds will triple by 2023, while broadband speeds will double. Content delivery networks help in this matter. In order to maximise speed and connectivity, CDNs place servers at the crossroads between different networks to work together to distribute content quickly, cheaply, reliably, and securely. There are a number of Internet exchange points (IXPs) where different Internet providers connect and share traffic originating on their different networks with each other. High speed data delivery can be reduced in cost and transit time when a CDN provider has access to these highly interconnected, high-speed locations. [7] In terms of security, since the ISP is not managing the infrastrucuture, it cannot be considered as secure.

2.2.4 Peering network

As described in section 2.1, ISPs are categorized into three tiers based on different properties and size. There must be a network where other ISPs are peering with each other and it is called Peering network in this document. In this research, the author was part of a tier 2 ISP. Since any traffic can come across this network, it is also considered untrusted, therefore traffic filtering and inspecting mechanisms must be implemented on this network segment.

2.3 Security requirements

Today's ISPs have high operational security requirements, so they must follow a framework that specifies profiles that collect all these requirements. The RFC3871 [8] was created in September 2004 and is being updated, which defines the security requirements for large ISP IP network infrastructure. The motivation of this document is based on four conditions: Ability to Control Service Bindings for Listening Services, Ability to Filter on Protocols, Identify Services That May Be Listening and Basic Filtering Capabilities, which are the key takeaways of the RFC regarding this research.

- Identify services that may listening

Vendors must identify protocols and ports on which services are listenning, as well as provide a list of all these services, that are active on devices. It mentions there may be

reasons why disclosing specifications of proprietary protocols isn't necessary, but in those cases their existence must be disclosed. [8]

- Ability to filter based on protocols

A requirement in this section is that the traffic must be filtered according to the protocol in the IP header. As a result, policies can be implemented, and operations can be made more secure. Furthermore, it discusses Internet Control Message Protocol (ICMP) flood attacks and proposes a way to mitigate Denial of Service (DOS) attacks by dropping all ICMP traffic. Concluding this section, it is essential to know which protocols and ports are in use to prevent these DOS attacks. [8]

- Ability to control service bindings for listening services

This requirement is about restricting access to management services. It is a good practice that management services are bound only to the loopback interfaces, that are only routed between managed devices and authorized management networks or hosts. By following this requirement, the number of ports listening will be reduced and complex filters will be less needed. Therefore, it is necessary to know what services are running and what ports are open on the device interfaces. It can also bring an advantage that since the management interface will be always up, till the device is running, loopback interfaces never go down until the device is alive. [8]

- Basic filtering capabilities

Filtering IP packets on any interface, which is a filtering mechanism to control traffic that goes inside and outside of any interfaces without significant performance degradation. It is the goal of this thesis, to test these filtering capabilities by fulfilling the first three requirements. [8]

2.4 Network Auditing

Enterprises must understand their infrastructure and audit it regularly, based on company policies. Network auditing consists of three parts: Network scanning, vulnerability scanning and vulnerability analysis. The purpose of network scanning is to identify which hosts are alive within a computer network, what operating systems they run, and what

services they provide. By using this information, a list can be created of the devices, which can then be used to conduct vulnerability analyses. As part of vulnerability analysis, signatures are compared to information collected from network scanning, and vulnerability scanning may also attempt to exploit vulnerabilities to verify their existence. Finally, vulnerability assessment is the part where remediation and patching take place by understanding the risks and acting based on the risk assessment. [9]

2.5 Stateless Firewalls

The purpose of stateless firewalls is to protect computers and networks. Stateless Firewalls, also referred as ACLs, is a security mechanism that filters traffic that is performed on incoming packets based on their UDP/TCP port numbers, source addresses, and destination addresses. Generally, stateless firewalls are unable to view packets as part of wider traffic and inspect them individually. They are also not capable of inspecting application-level traffic types (such as HTTP, HTTPS, FTP, VoIP, SSH, etc.). This makes them vulnerable to attacks that spread across many packets rather than being hidden within one. Additionally, stateless firewalls do not keep track of the network's status or its connections. Due to the fact that stateless firewalls only inspect the header portion of an inspected packet, they are quicker and more efficient. [10]

2.6 Infrastructure Access List

From Cisco's best practices [11], it is known there are two types of traffic. Transit traffic, which passes through a router to reach its destination, and traffic destined for the router, like SSH or ICMP. To mitigate direct router attacks, it is necessary to restrict access to infrastructure equipments from external sources. Excessive traffic is likely to raise CPU usage and result in packet loss, that can cause DOS. There are, however, several filtering techniques to prevent direct router attacks, including Receive ACLs, which filters traffic destined for the router without affecting transit traffic. However, Receive ACLs must be deployed on every router, which is a drawback. In hop-by-hop router ACLs, only authorized traffic is permitted to the router interface, while all other traffic is denied. Transit traffic is affected, causing forwarding performance issues. Using iACL edge filtering, the ACL is applied at the edge of the network. In the case of an ISP, at the edge

of the AS. iACLs filter traffic specifically destined for infrastructure services. This technique is the most flexible, since it only has to be deployed on the edge devices and does not affect transit traffic.

In case of using iACLs, there are address spaces that must be denied, such as the private address and special-use address. Additionally, anti-spoof filter must be applied, since the infrastructure's address space must never be the source of the packets from the external network. [11] An example can be see on Figure 3.

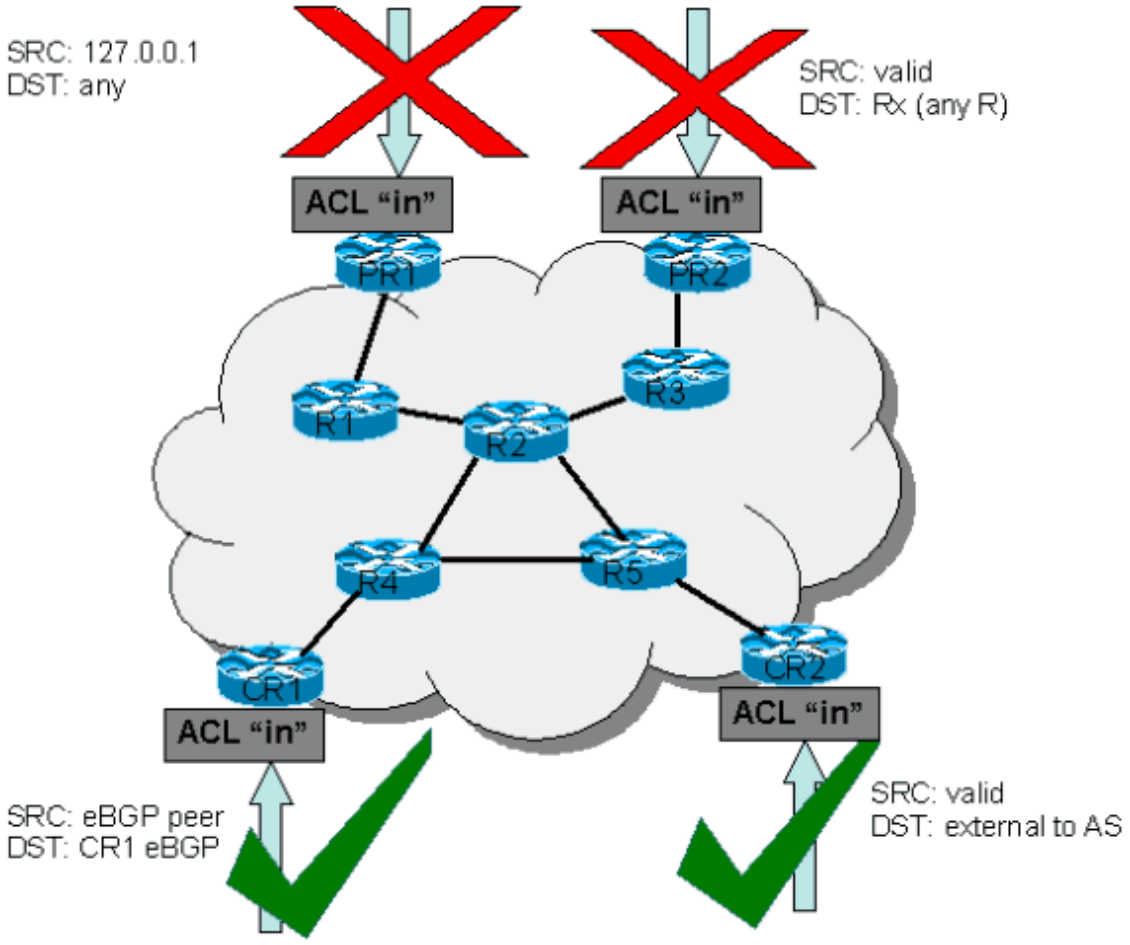

Figure 3: Applying filter on the edge devices [11]

2.7 Ansible

Developed by Red Hat Enterprise, Ansible is an automation tool that is widely used to deploy Infrastructure as a Service (IaaS). The original purpose of Ansible was to manage configuration files on Linux systems before it was expanded to network applications.

Among the greatest advantages of Ansible is the ability to structure scripts, allowing a collection of tasks to run simultaneously on several devices. One of the unique features of Ansible is that it is a push-based configuration management tool. It means, the main server pushes configuration to the target node. Additionally, Ansible is agentless, meaning that no agent is required to be installed on each managed node. Due to the research's flexible aim, this criterion played a critical role. This is because the scan has to be conducted from any hosts device from any networks. Due to these two reasons, other configuration management tools, like Puppet or Chef cannot be used. It is because they are both push-based configuration tools, and agents must be present on managed devices, therefore they would provide less flexibility.

There are a wide variety of pre-written modules for Ansible, that were written in Python and are an easy to implement solutions for many problems. On the other side, Python offers libraries such as netmiko and paramiko, which are also handy tools for network automation. The advantage of Ansible over Python, however, is that it describes the desired state of computer systems and services using a state-driven resource model, which will be very important in this thesis. Additionally, it uses YAML, which is a human readable format and easier to learn than Python.

Ansible uses four key components. Inventory that consists of nodes to manage by Ansible. Modules do the actual work and are getting executed in each playbook. Playbooks consist of tasks, called Ansible modules, which then orchestrates, configures, administers, or deploys. Lastly, Ansible configuration, where the working configuration of Ansible is located.

REST API and Ansible are frequently mentioned in this document. As a default state, Ansible does not offer API functionality, but the AWX project provides a web-based user interface, REST API, and task engine, making it one of the upstream projects for Ansible Automation Platform. [12]

2.8 Python-Django

This open-source framework was released in 2005 and uses Python to create web applications, supported by Django Software Foundation [13]. Their aim is to encourage

rapid development while maintaining a pragmatic design. The Django web framework is one of the leading frameworks for Python web applications. It uses the Model-View-Template software design, which is a variation of the widely spread Model-View-Control design pattern. Django offers frontend, backend, and security solutions right out of the box. During this study, Django was used to create the backend and provide the API for the stored resources, like IP prefixes/addresses, devices and their interfaces. There are plenty of popular projects that use Django, including Instagram, Mozilla Firefox, Spotify, and even YouTube.

Organizations can use the single source of truth (SSOT) concept as part of their information architecture to ensure that all employees in the organization are using the same data when making business decisions. By adopting a SSOT, employees will have access to a federated view of data, which is also known as a golden record or single version of the truth [14].

According to their official documentation, Netbox is the leading solution for modelling and documenting modern networks. It is an ideal SSOT option for network automation, since it was designed specifically for network engineers and operators. Racks, devices, device components, cables, wireless connections, data circuits, virtual machines, prefixes, ranges, addresses, VRFs, L2VPN overlays, VLANs, etc., are just some of the network technologies covered in the Netbox project. Also, Netbox is a Python-Django-based open-source application that can be extended by adding custom fields, custom models, templates, plugins, and REST APIs [15]. This project relies heavily on Netbox as the SSOT, since it can store prefixes, providing REST APIs, and supports webhooks. According to RedHat, Webhooks enable lightweight, event-driven communication between two application programming interfaces (APIs) [16].

Figure 4 shows the attributes of each prefix object stored in the inventory. Initially, it provides prefixes to be scanned by the REST API for Ansible, and webhooks can be added to create trigger points, so it can react when a database change occurs.

The author was part of the development team that extended the functionality of Netbox by implementing REST API endpoints. These endpoints provide the necessary information about network segments and edge routers.

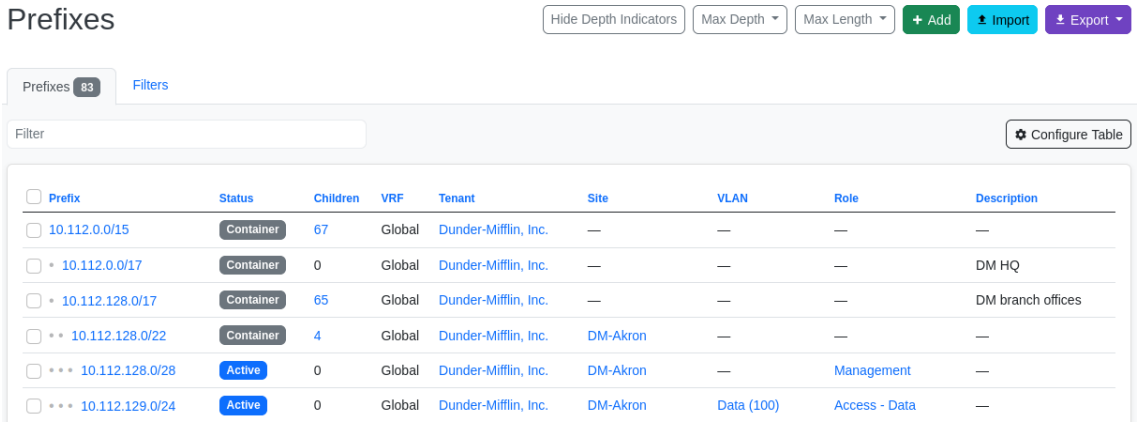

Figure 4: GUI of Netbox storing prefixes [17]

3 Background Research

The objective of this section is to perform a comparison of various technologies based on available researches. Properties, such as active and passive vulnerability scanners, performance, accuracy, precision, and difficulty to implement, were considered important criteria in order to identify the right tool for this thesis.

Normally, network vulnerability scanners audit a network in three parts: network scanning, vulnerability scanning, and vulnerability analysis. Network vulnerability scanners as their name states, are both network and vulnerability scanners. The scanning of a network involves identifying which computers in the network are alive, what operating systems they are running, and what services they are providing. In most scanner tools, vulnerabilities are verified by carefully constructed queries, which help confirm the vulnerability's existence without interfering with the tool's operation. [9] The goal of this thesis was to find a network scanner that would best suit testing the functionality of the stateless firewall. The research methodology emphasized the need to find a tool that could test the infrastructure of the ISP from the outside, which is already running a vulnerability scanner, called Nessus. Tenable's Nessus vulnerability assessment solution, according to them, is the industry's most trusted solution. In addition to having the lowest false positive rate and six-sigma accuracy, Nessus has the deepest and broadest vulnerability coverage in the industry with over 2 million downloads. [18] There are a few criteria to in order to find a cost-effective vulnerability scanner. A vulnerability scanner is classified into two main types based on the method it uses to identify vulnerabilities: Active Vulnerability Scanners (AVS) and Passive Vulnerability Scanners (PVS).

3.1 Active and Passive Vulnerability Scanners

To detect vulnerabilities, AVS sends traffic to hosts on scanned networks and analyses their response to determine if any vulnerabilities exist. As a result, active vulnerability scanning is considered intrusive. In PVS three methods are possible to identify vulnerabilities. First option is to capture and analyse traffic to determine which devices and services are running in the infrastructure. Alternatively, one can analyse logs, operating systems, and configurations to learn more about installed software and services.

Another alternative is to run sophisticated queries against known vulnerability databases to identify the services and nodes affected. [19]

One of the leading industry vulnerability scanners, Nessus is an AVS tool, and they claim more than 70.000 Common Vulnerabilities and Exposures (CVE) are covered by their scanning scripts. However, as of September in 2022, the total number of CVE's are more than 185.000, which gives only an 38% coverage. On the other hand, PVS can cover all the known vulnerabilities that are contained at the vulnerability database or repositories [19].

AVS also has a side effect since it needs to send packets on the network, so it increases network traffic, affecting performance and it varies by the size of the network. In addition, scripts run by AVS can disrupt services on hosts. In contrast, PVS is not affected by this problem. When the whole network is not visible, such as when separated by a firewall, AVS may block some TCP/UDP traffic and produce false negatives. AVS and PVS can both produce false positives and negatives, but their rates differ according to the factors involved.

An AVS and PVS test are conducted for three main operating systems, covering the most recent 100 CVEs. This research is conducted by Harun Ecik. Detected vulnerabilities are considered accurate if all of them have been discovered within a target. Detecting only existing vulnerabilities within a target (excluding false negatives) is called precision. According to the study, accuracy, and the ability to detect all vulnerabilities were the most important criteria for detection. The researchers conclude that PVS returns more accurate and complete results. The result can be observed on Figure 5.

OS/APP	SCANNER	Active Vulnerability Scanning			Passive Vulnerability Detection		
		Total CVEs	Accuracy	Precision	Total CVEs	Accuracy	Precision
Windows XP	NESSUS	120	34.70	53.60	208	46.30	56.00
	NEXPOSE	80	41.90	77.50		46.70	
	OPENVAS	448	29.90	23.00		46.30	
OS/APP	SCANNER	Total CVEs	Accuracy	Precision	Total CVEs	Accuracy	Precision
Ubuntu Linux	NESSUS	767	15.50	20.00	1658	98.00	98.00
	NEXPOSE	398	10.90	14.00		92.50	
	OPENVAS	310	16.20	19.00		89.10	
OS/APP	SCANNER	Total CVEs	Accuracy	Precision	Total CVEs	Accuracy	Precision
Windows 7	NESSUS	438	37.60	70.00	299	76.30	90.00
	NEXPOSE	252	69.00	89.00		81.10	
	OPENVAS	1850	0.00	0.00		90.00	
OS/APP	SCANNER	Total CVEs	Accuracy	Precision	Total CVEs	Accuracy	Precision
Windows XP VLC P.	NESSUS	88	55.70	67.00	65	78.50	95.40
	NEXPOSE	56	44.60	73.20		77.50	
	OPENVAS	61	62.40	86.90		78.50	
OS/APP	SCANNER	Total CVEs	Accuracy	Precision	Total CVEs	Accuracy	Precision
Windows XP M.Player	NESSUS	1	20.00	100.00	7	71.40	71.40
	NEXPOSE	3	42.90	100.00		55.60	
	OPENVAS	6	0.00	0.00		71.40	
OS/APP	SCANNER	Total CVEs	Accuracy	Precision	Total CVEs	Accuracy	Precision
Ubuntu Linux Curl	NESSUS	5	11.60	100.00	53	86.00	86.00
	NEXPOSE	5	11.60	100.00		86.00	
	OPENVAS	4	9.30	100.00		86.00	
OS/APP	SCANNER	Total CVEs	Accuracy	Precision	Total CVEs	Accuracy	Precision
Ubunutu Linux Samba	NESSUS	15	27.90	80.00	49	71.20	75.50
	NEXPOSE	16	35.70	93.80		69.80	
	OPENVAS	26	33.30	64.00		72.50	
OS/APP	SCANNER	Total CVEs	Accuracy	Precision	Total CVEs	Accuracy	Precision
Windows 7 M. Player	NESSUS	7	63.60	100.00	9	33.30	55.60
	NEXPOSE	7	63.60	100.00		33.30	
	OPENVAS	1	50.00	100.00		55.60	
OS/APP	SCANNER	Total CVEs	Accuracy	Precision	Total CVEs	Accuracy	Precision
Windows 7 Putty	NESSUS	11	61.10	100.00	18	100.00	100.00
	NEXPOSE	0	0.00	0.00		100.00	
	OPENVAS	8	30.00	75.00		100.00	

Figure 5: Accuracy and precision result [19]

Ron Gula summarizes the differences between AVS and PVS as follows: "Passive vulnerability scanning is not a substitute for active scanning." PVS is a separate technique that allows network and security engineers to use distributed network monitoring tools for troubleshooting purposes. Still, PVS will not be able to generate the same amount of

raw data as AVS can. [20] From this summary, it is evident, using both PVS and AVS is the best practice to secure any enterprise's network.

Additionally, Nessus as an AVS performs poorly in network scanning compared to other network scanners. It is true that Tenable Network Security Inc. offers a PVS, however the author needed to consider an easier, more robust, and open-source solution if it is available. For this reason, to identify hosts that are not hidden by ACLs on the infrastructure, Nmap is proposed as a network scanner in this thesis.

3.2 Nmap

In the official documentation, Nmap refers to their software as a Network Mapper, an open-source tool for auditing networks. It is designed for scanning large networks rapidly, however, can also be used on single hosts. Raw IP packets are used by Nmap in novel ways to determine which hosts are available on the network, what services those hosts offer (name and version of the application), which operating systems (and version of the operating system) are used, what type of packet filters/firewalls are used, and several other aspects. Besides network audits, Nmap is often used for routine tasks like creating network inventory, service upgrade scheduling, and monitoring host and service availability. [21]

A passive vulnerability detection solution is proposed by Ron Gula [20] in his research by using passive network monitoring. For example, in the case of client-based vulnerabilities, the client itself can be detected rather than the vulnerability itself. He mentions, this solution also helps to reduce the scope of the hardware related vulnerabilities, because if a company never bought, for example, a HP-UX server, there is no need to search for HP-UX related vulnerabilities in the future. Additionally, he discusses how to test firewall vulnerabilities with network scanners.

3.3 Nessus and Nmap performance comparison

In this research from 2016 [22], on a network consisting of 40 devices, the authors compared the performance of Nessus and Nmap. The accuracy, precision, and scan time of 20 Linux and 20 Windows hosts were measured regardless of whether a firewall was

present or not. Precision and accuracy were calculated based on the ratio of false negatives, false positives, and true positives to all devices. They calculated it as follows:

Classificaiton	**Description**
True Positive – Identification success	Then scan result is equal to the actual operating system
False Positive – Misidentification	The scan result is not equal to the actual operating system
False Negative – Identification Failure	The scan failed

Table 1: Classification on scanning results

Accuracy	**Precision**
TP/(TP + FP + FN)	TP/(TP +FP)

Table 2: Calculation of accuracy and precision

Tool	**True Positive**	**False Positive**	**False Negative**
Nmap	8	12	20
Nessus	3	0	37

Table 3: Comparison of scan results when firewall was present

Tool	**Accuracy (%)**	**Precision (%)**
Nmap	20	40
Nessus	7.5	100

Table 4: Comparison of accuracy and precision

Based on their summary of the results, Table 4 shows that Nmap has been able to identify operating systems more efficiently than Nessus. Moreover, it is apparent Nessus had more difficulty identifying Operating Systems on the devices, however its precision was 100% accurate when it did. Table 3 shows that, Nmap was able to give less False Positives and False Negatives, while giving more True Poisitives, when firewall was present, which is considered a huge advantage in this study.

In the same study, their average scan time was also compared on both Windows and Linux machines.

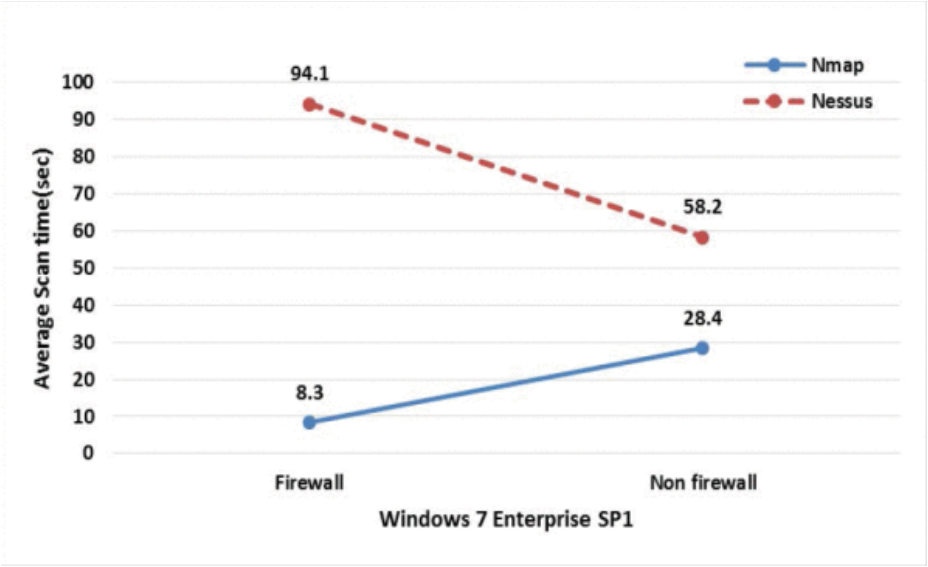

Figure 6: Average scanning time on windows host

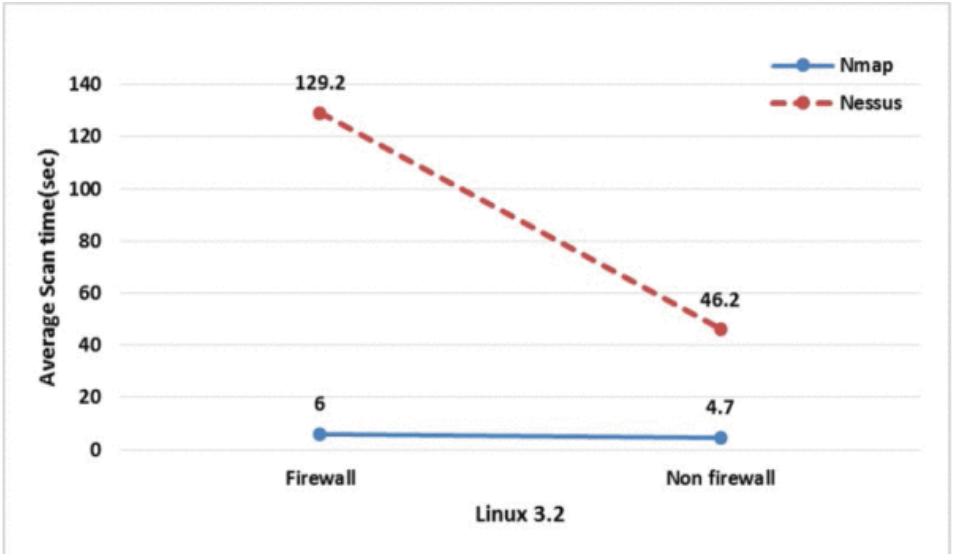

Figure 7: Average scanning time on Linux Host

This means that Nessus takes longer on average to scan the operating system than Nmap.

3.4 Comparison of OpenVAS and Nmap

The OpenVAS vulnerability scanner is a comprehensive tool. In addition to unauthenticated and authenticated testing, it supports high-level and low-level internet and industrial protocols, performance tuning for large-scale scans, and a powerful internal programming language for implementing all types of vulnerability tests. [23]

This comparison mentioned in the research made by Nikita Y Jhala [24], OpenVAS and Nmap are compared by different properties, such as vulnerability assessment, complexity to install, configure, report generation and scan duration.

In conclusion, Nmap offers less scripts for the Nmap Scripting Engine and vulnerability scan results have a higher number of false positives than OpenVAS. As Nessus takes on the role of AVS in the infrastructure, the author examined these tools as network scanners, that can be later used as a PVS.

OpenVAS is complicated to setup and configure since it has a client-server architecture over SSL, and users need to be created on the server where scans are performed. It means OpenVAS has flexibility limitations, which results in a disadvantage over Nmap.

Due to the fact that both tools can generate an XML file or any other processable output format, report generation ability is not considered decesive.

However, it takes significant time for OpenVAS to conduct vulnerability scans. Since vulnerability scanning is out of scope, OpenVAS does not work well by conducting only network scans without performing vulnerability assessments, since it is a vulnerability assessment tool rather than a network scanner.

As a result of the comparison, Nmap is a more robust, lightweight network scanning tool, whereas OpenVAS is a more complex solution with much more GUI output and useful features. However, this isn't an important factor in this study, so Nmap is the perfect solution.

4 Solution

Chapter 2 and 3 should give enough background knowledge for the reader in order to understand the problem and solution in more depth. Hence, to conclude the problem, in order to hide infrastructure devices, iACLs must be created to prevent access from external networks. An ISP's infrastructure is constantly changing and has many devices around the world. It is therefore necessary to have a process which automatically detects if ACLs do not function as expected

The internal network of an ISP is responsible for transporting traffic between its customers, so optimal and undisturbed behavior is essential. If an iACL does not function correctly or is missing, devices are visible from the Internet, making it possible to conduct direct attacks. In addition, scanning infrastructure devices will be possible, which can provide valuable information for an attacker. Moreover, an edge device should drop unverified traffic when it enters the internal network, in order to save resources for the core devices.

4.1 Open and Closed Loop Automation

During the thesis, the author had the opportunity to choose from two automation methods, which are open and closed loop automation systems. Processes in closed loop systems are able to provide feedbacks for the controller, that helps to maintain the desired state. In contrast to open loop systems, which does not provide this possibility. Feedbacks were essential in this project, since it is how the components communicate with each other inside the system. Additionally, the process controller is able to understand its current state by processing the feedbacks. It is important, since if the current state differs from the desired state, some action must be taken in order to recover it. By using open loop systems, it would be harder to maintain the desired state. Ansible is an excellent tool, since it uses state-driven model, which means, it is possible to define the desired state of the system by applying playbooks. At each run, Ansible tries to regain the state by executing the playbooks, and a feedback is provided in case of the current state differs from the desired state during the execution, or not. In case, there was a change in the

system, the process controller understands some action must be taken. On Figure 8 the differences between Open and Closed Loop systems can be observed.

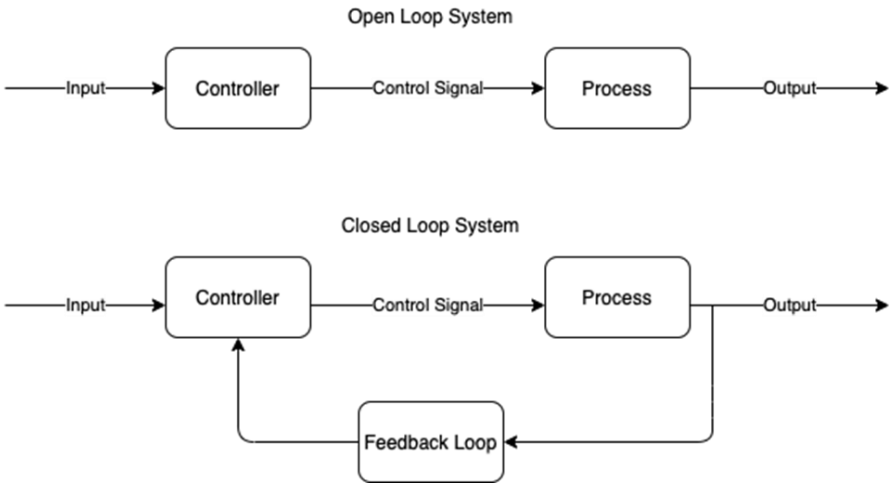

Figure 8: Open and Closed Loop Systems [25]

4.2 Possible solutions

Beside open and closed loop automation, other approaches had to be considered during the research. Manual, fully automated and semi automated approach each have different advantages over each other.

4.2.1 Manual approach

In this approach, engineers must manually check the visibility of the infrastructure using network scanners. If a problem is detected, it must be rectified manually. As a result, if a new device is added to the infrastructure, manual configuration must be created and validated. Manual configurations have a high risk, as human errors can occur, such as typos. Automation can prevent these errors, so a better approach is needed.

4.2.2 Fully-automated approach

Using this approach, infrastructure changes are detected automatically and the system knows where to make the changes and on which devices they should take place. It is entirely automated, and is controlled by an artificial algorithm. In addition, it can perform scans, verify device configurations, and correct them as necessary. It is very close to a software-defined network, however they are not identical. As Red Hat Enterprise

describes it, this is a self-healing infrastructure based on closed loop automation. [25] In closed loop automation, an algorithm is used to ensure a system maintains the desired state. In contrast to open loop automation, closed loop automation provides feedback to the controller to assist it in understanding and monitoring its states. This approach is more expensive than manual approach and requires a great deal of resources.

4.2.3 Semi-automated approach

Semi-automated systems are hybrid systems that combine the properties of manual and fully-automated approaches. In other words, the system cannot recognize every change in the infrastructure, such as the implementation of a new device. However, it can react to it by knowing where and what needs to be changed. The system must, therefore, be notified if a device is implemented, so that it can make the necessary changes to keep the desired state. Due to this, some human interaction is necessary. However, a major advantage of this system is that, unlike the fully-automated approach, it does not eliminate the possibility of manual changes and also reduces the chance of human error. As well as providing a sufficient solution, this system is the most cost-effective solution of all three. A simple example of this system is an air conditioner, which uses a themometer as feedback to maintain a desired room temperature, however the temperature must be manually set. On the other hand, a fully-automated approach would remove this manual step, since it is able to specify the desired state without human intervention by understanding the resident's habits from additional feedbacks.

4.3 Implementation requirements

As part of this research, a semi-automated system is proposed as a solution. In order for the system to work undisrupted and optimally, some requirements must be met.

- Valid and up-to-date data must be used in a system.

Data from un-updated databases should never be used in the system, since addresses, rule constructions, and device reachability rely on that information. The system must be able to detect if the data stored in the database is not consistent, so it can be validated for the future. Netbox can be used for this purpose, to store addresses and prefixes.

- Communication with all networking devices must be possible.

The system needs an undisrupted, reliable communication in order to make changes on the devices. The risk cannot be prevented if only half of the devices configuration can be modified.

- Changes must be validated by the system.

Suppose, an engineer reserves a prefix for a new device in the inventory, but the address configured on the device is completely different from the reserved prefix. Therefore, the first requirement is not met, which results in the edge device configuration being generated incorrectly. That is why, each change must be validated by the system, since it cannot trust in manual modifications.

- System must know the current and desired states.

A desired state must be explicitly stated, that the system attempts to achieve. In addition, the system must know the current state through various monitoring and scanning tools. As a result of not knowing both states, the system is unable to make a difference and then take the appropriate steps to recover.

- A possibility of manual modification must exist.

Depending on the situation, the system must offer the option of manually modifying the configuration or database. Although manual modification is always available on networking devices, the system must be capable of recognizing these changes and responding accordingly. That cannot happen, after a manual configuration change the database is not updated, because of human error. This must be updated and validated by the system if needed.

4.4 Implementation

The proposed semi-automated system consists of three main and a helper component.

The first component is an inventory, which is an application that provides valid data inside the system. This application is implemented using a web framework, like Fast API or Django. It has API functionality, database capability and allows to create logic using its backend. For this project, REST API was used, that allows other components to communicate with it, using get, put, patch, and delete requests and to modify database contents. On the other hand, the content can be also changed by a GUI, since it is where engineers can interact with the system.

Next, is the iACL management component, that has the responsibility to generate and load the configuration into the correct devices. It also has a helper component, that can detect the manual configuration change on the edge devices. Network scan is always triggered if a configuration change is applied.

Finally, the network scan component that is an Nmap scan to check the exposed hosts from the external network. It also has the responsibility to notify the inventory if any hosts are visible.

4.5 Desired State

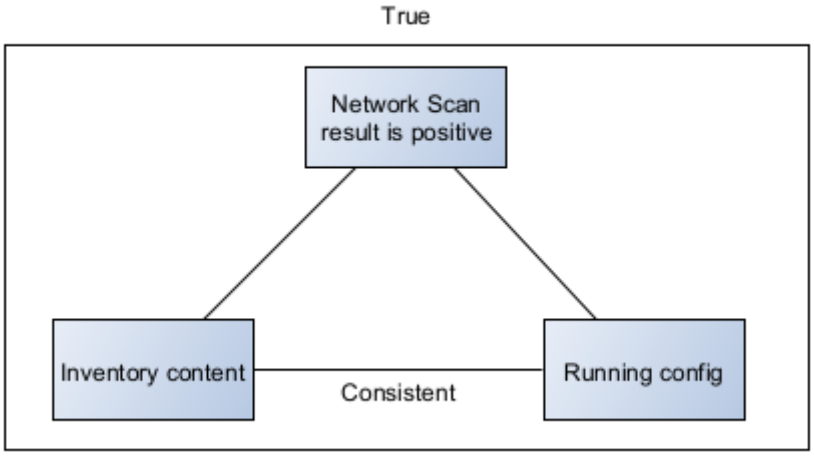

Figure 9: Desired State to be maintained

On Figure 9, the requirement for the system to be in desired state can be observed. It means, the database must store all prefixes that are in the running configurations. Additionally, the running config has to include all prefixes that are stored in the database. Consistency can only be achieved through this approach. Additionally, network scanning must always return a positive result, meaning there are no internal hosts visible from the external network. The connection between the states is important, as this is the reason for the closed loop nature of this system. Network scanning and the inventory content validates the ACL configuration. Network scanning and the actual running configuration validates the inventory content. The network scan, however, does not need to be validated, since it is a component for verifying configuration and database errors. In the following sections these connection between states are described.

4.6 Practical implementation

It was the aim of this thesis to find a way to implement the solution into the ISP's infrastructure. In practice, three processes are demonstrated and these constitute the main components of this practical implementation. Ansible has the responsibility to call the necessary playbooks based on the feedbacks. Feedbacks are positive and negative results, where positive means the system is in desired state, however negative result does not neccessarly indicates undesired state. It can also indicate connection errors. Therefore, the system is a collection of playbooks that are executed based on the state of the system. It is based on the property of idempotence, since if the system is in desired state, no matter how many times the playbooks are executed, the state must not be affected.

4.6.1 Idempotence

The property of idempotence in programming and mathematics is that a particular operation produces the same result no matter how many times it is executed [26].

4.6.2 Network scanning component

In the first component, a network scan is carried out from an external network toward the MPLS backbone using Nmap. The external network was explained in Chapter 2, and it consists of a number of networks such as access, CDNs, and peering. Therefore, the first step is to connect via SSH to a host on the external network using Ansible. As long as the connection is successful, which is a basic requirement for implementation, then Ansible tries to install Nmap. However, if it is already installed, nothing is changed. In the next step, Ansible then asks for a list of devices that needs to be scanned from the inventory. As a result, Ansible creates a get request for the inventory via its built-in get_url module. Then, it reformats the received JSON file and converts it to a txt, which is then copied to the target hots, via the built-in copy module. If the copied txt content matches the target node's txt, it can be concluded that no changes have occurred since the last scan. Therefore, the process returns with a positive result, since the system is in desired state. Alternatively, Ansible uses its shell module to perform an Nmap scan, which generates an XML report afterwards. In the ideal case, no hosts are visible from the external network, which means, in practice, the XML file contains the "hostup=0" tag, which can be easily identified by regex search. This component is mostly used for validation, which

was an implementation requirement. If the regex appears in the file, the scan returns a positive result, however, if it is not, the component returns with negative result. A flowchart of this process can be observed on Figure 10.

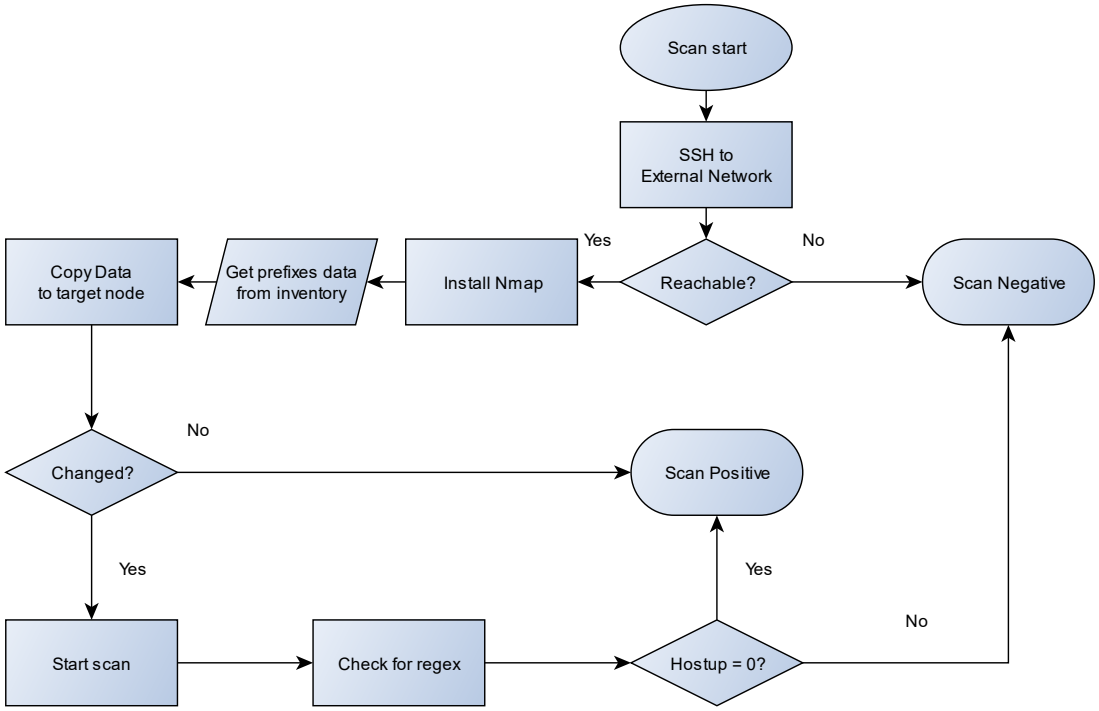

Figure 10: Process of network scanning component

4.6.3 Access List management component

The second component is the management of ACLs on devices. In this process, Ansible is triggered by the inventory via REST API. Ansible must be able to connect to the networking devices, or else it will return a negative result. ACL can be managed using several Ansible modules, including Juniper's junipernetworks.junos.junos_acls [27], Cisco's cisco.iosxr.iosxr_acls [28], and Huawei's community edition [29]. The built-in CLI module can also be used to achieve vendor neutrality in many cases. This process, starts with Ansible pulls the current iACLs from the running configuration and compares them to the inventory contents. If they do not differ, that means no changes have been made, therefore the database and running configuration are consistent with each other, so the component returns a positive result. Two possibilities exist in the other case. Firstly, if the database is not updated, which is critical since it is one of the basic conditions, then Ansible must be capable of reformatting the missing data and sending it to the inventory

via REST API. It can be done using the third component and will be described in more details. If an engineer updated the ACL of the device but not the inventory, then this solves the problem, since manually making changes if necessary was also among the basic conditions. Another possibility is that, the inventory contains a network that is not hidden by the edge devices. In this case, Ansible is responsible for generating the ACL configuration and place them into the correct devices. Then, it must be validated through the network scanning component, since the process can only return a positive result if network scanning is also positive. It is how the generated configuration is validated and the process sends feedback. If the network scanning result is negative, then manual changes are needed, since the system is not yet prepared for that scenario. It would mean, the generated ACL is wrong. This process can be observed on Figure 11.

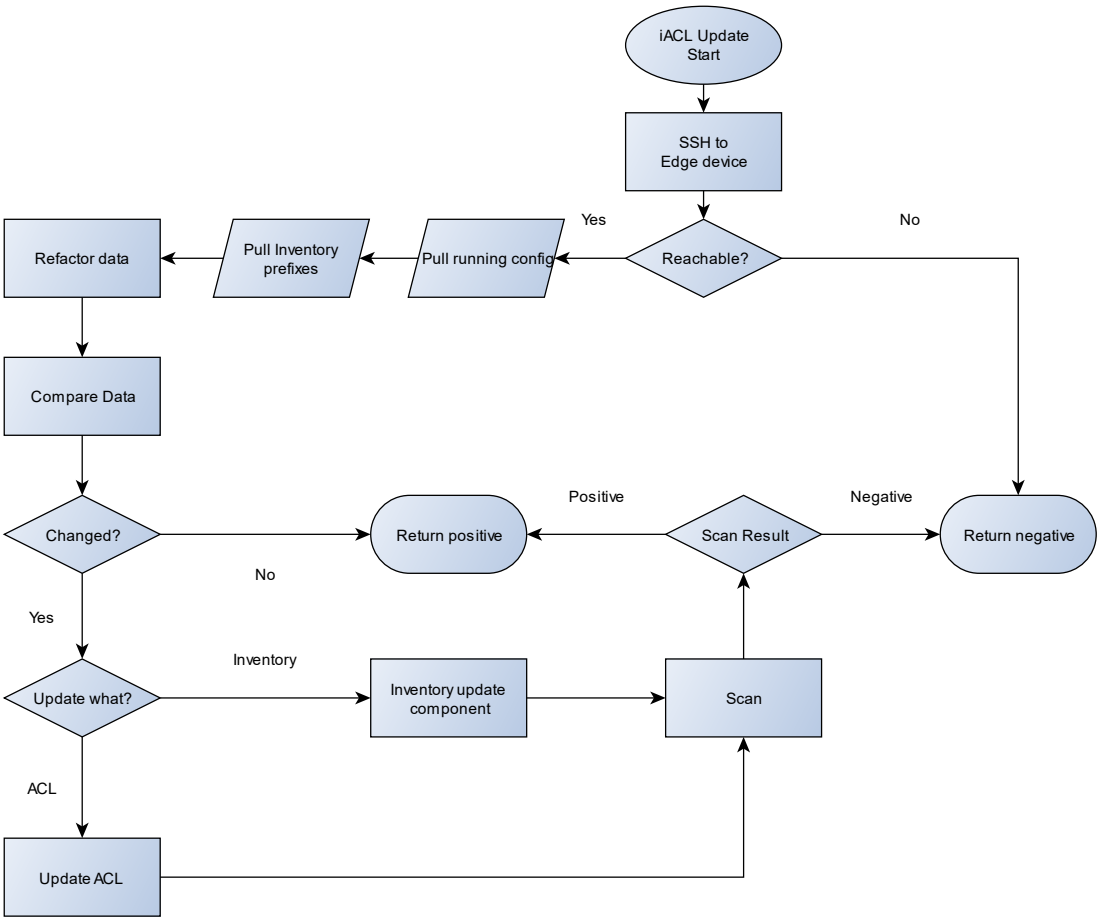

Figure 11: Process of access list management component

4.6.4 Configuration change detector helper component

It is more difficult to implement this process in practice since Ansible is not able to react when ACLs are manually changed, since by default it cannot detect when and what was the change. One solution would be to continously pull the running configuration of the devices, however an even better solution is to use a configuration change detector program.

Located on a log collector node, this helper component is a simple progran that can analyze logs and notify the iACL management component. It can be achieved by using Simple Network Management Protocol (SNMP). One option is to use Cisco's Management Information Base (MIB) [30]. Using this MIB allows to create SNMP traps, so devices can send logs for the colletor node. In that case, the node can understand if the modification affects the ACL configuration, so it triggers the ACL management component, by creating a HTTP request. However, in some cases these MIBs are not supported. Another solution is to use the log message that is always generated after commit. A part of the generated log message on a Cisco IOS XR looks the following:

%MGBL-CONFIG-6-DB_COMMIT : Configuration committed by user test.

In practice, if the "configuration commited" string can be found in the logs, the system can start comparing the content of the inventory and the running configs, by triggering the ACL management component. In case of the configuration did not change the ACL config, the component returns with positive result, since the system is in desired state already.

4.6.5 Inventory update component

The last component is the communication between Ansible and the Inventory. This process manages the database and provides data between Ansible and the Inventory through their REST API. The solution must allow engineers to manually interact with the system, therefore they should be able to modify the database content using a Graphical User Interface (GUI). When the modification is saved in the GUI, it triggers the ACL management component via HTTP Request.

On the other hand, inventory update is also capable of receiving new prefixes through its API. It can happen, as an example, when a network scan component finds an exposed network or when the running configuration has networks that are not yet stored in the Inventory. It starts, by the component receives the prefix and the action, which can be either delete, put or post. Then, Ansible process this data and sends it for the web application. In any cases, it will return with a positive result, however if the prefix is not in the inventory yet, then the web application updates its database content. The process can be seen on Figure 12.

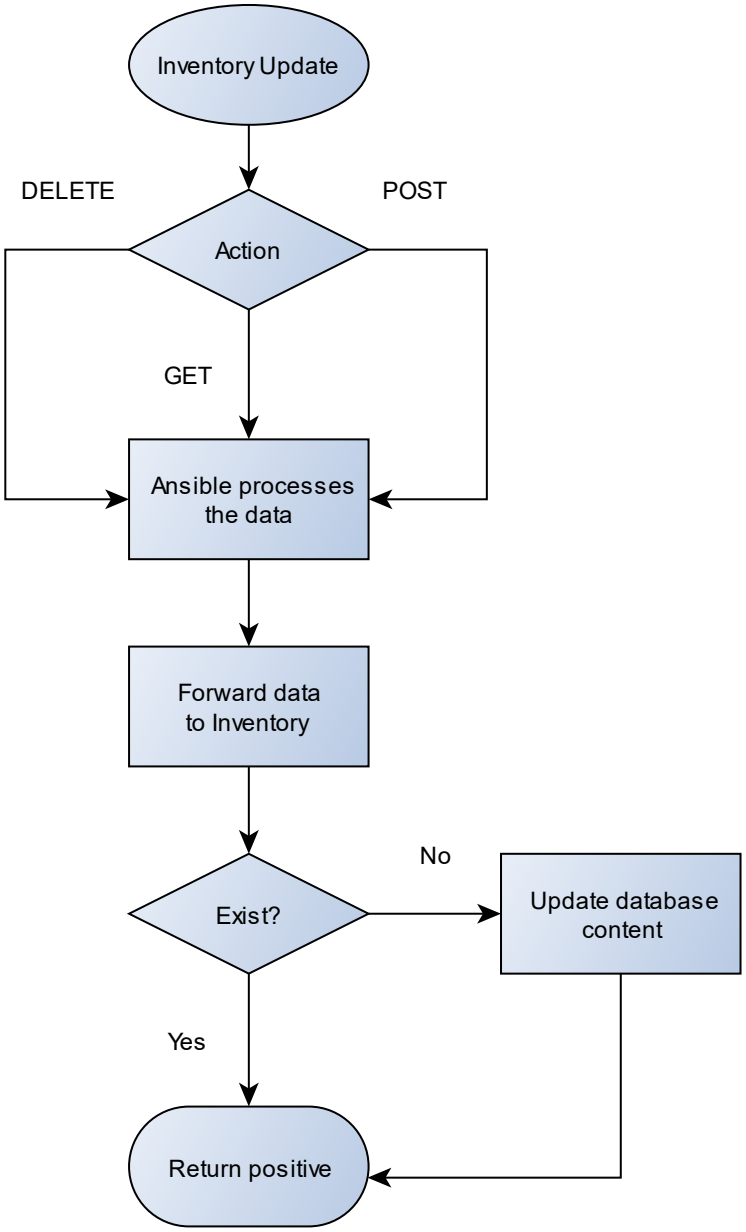

Figure 12: Process of the inventory update component

4.7 Connecting the components

After connecting the components, the system can solve the problem defined in the problem statement, since the generation of ACLs and their validation is automated. Connected components can have three triggers points. If the inventory is updated, if manual configuration change is detected or when a network scan returns with negative result. The system is able to catch the trigger and call the necessary processes accordingly. Each process of the connected components can be considered successful if they return with positive result.

The main purpose of the system was invented for the case, when an engineer reserves a perfix in the inventory and saves the modification in the GUI. In that case, Inventory notifies the ACL management component, that understands there is a new prefix, which is not in the running configuration of the devices. Therefore, it generates the config and loads it. Later on, the generated configuration is validated with the network scan component, that is always called after commits. In case of the all components return with positive result, the process finishes. Otherwise, manual steps are required, since some cases cannot be handled because of the nature of the hybrid system. Such case could be if a device is not reachable.

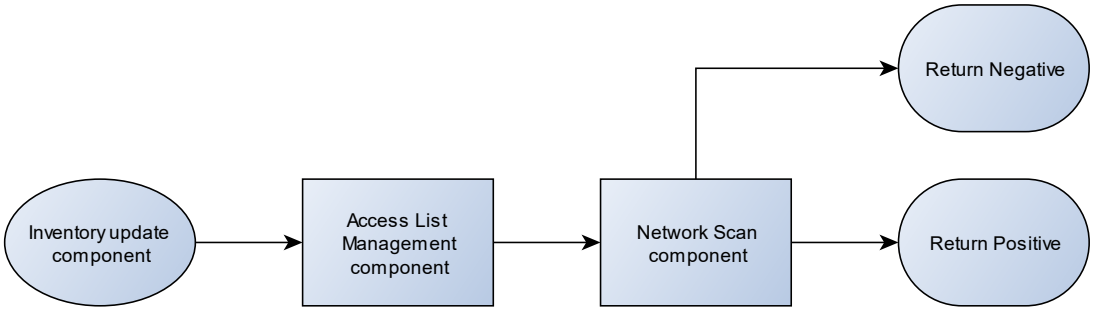

Figure 13: Inventory update with connected components

Engineers must be able to manually modify running configurations if needed. Therefore, by implementing the ACL management helper component, Ansible can be notified if there was a change in the edge devices. In this case, the helper program will notify the ACL management component, that understands by comparing the running config and the inventory content, that they are inconsistent, therefore it must call the inventory update component. In the end, a network scan is conducted, since that can happen, ACL is not

updated on each LERs. In case of negative result, manual steps are required. This process can be observed on Figure 14.

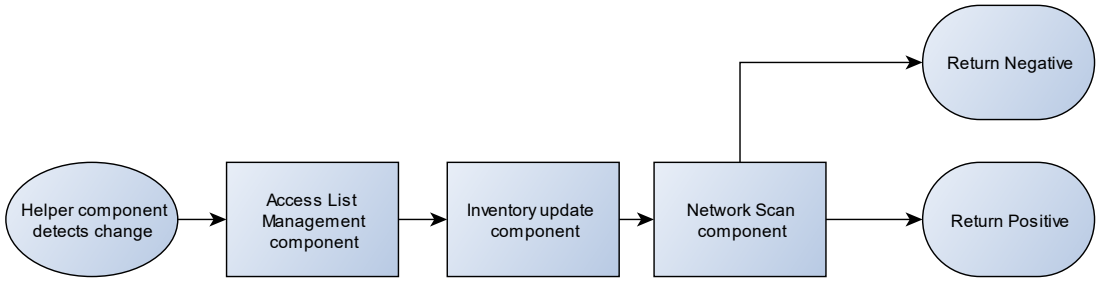

Figure 14: Access list update with connected components

In order to help the network audit process, it is important to eventually start a network scan, which is a passive vulnerability detection mechanism. In case of the scan result is negative, it is possible to find the prefixes in the generated XML report and send it for the inventory through its API. Then, the process from Figure 13 starts from the beginning. An example diagram of the process can be seen on Figure 15.

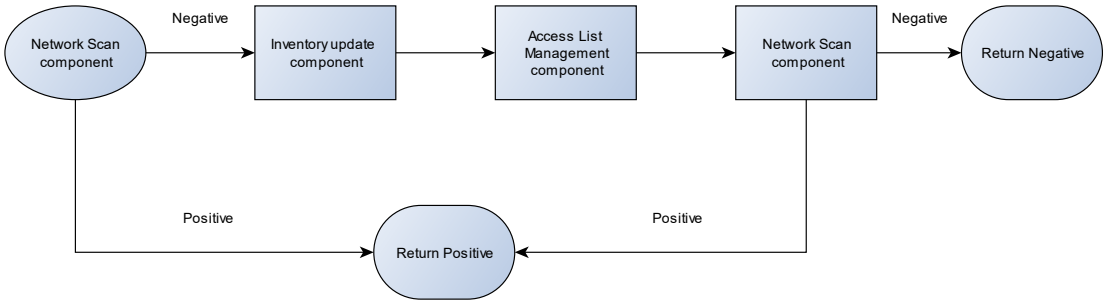

Figure 15: Network Scan with connected components

4.8 Error Handling

Semi-automated systems have many advantages, but they also have some significant disadvantages. As an example, if ACL generation was incorrect, then the scan process would return negative results. Means, the system is not able to recover its desired state, so manual changes must be made. The easiest solution is to create an email sending component that alerts engineers if a process returns with a negative result, however there can be many solutions, depending on the complexity of the processes and infrastructure.

4.9 Environments

To develop and debug software, a testing environment is required. Testing environments can sometimes differ from production environments, since in some cases not all the functionality of production environments is needed. This was also the case in this research, since Netbox in this enterprise is quite advanced and only API Endpoints and database functionality were critical for testing.

4.9.1 Test Environment

In order to replace Django and Netbox, the author used FastAPI, that functions the same way as Django API, since both are written in Python. Therefore, FastAPI [31] were handling API calls. In order to simulate the database, a simple JSON file was created, that allowed to read, write and modify values as dictionary objects in Python. This side project can be also found on the author's github [GitHub FastAPI project], but is still in development state. As an edge device, the company provided a Cisco CSR 1000V Router's virtual image. Additionally, a third machines was created to run Ansible to orchestrate the processes. All applications and the router were running in a virtual, hypervised Vmware vSphere environment that allowed them to communicate with each other using its virtual switch methodology [32]. This test environment requires a lot of resources.

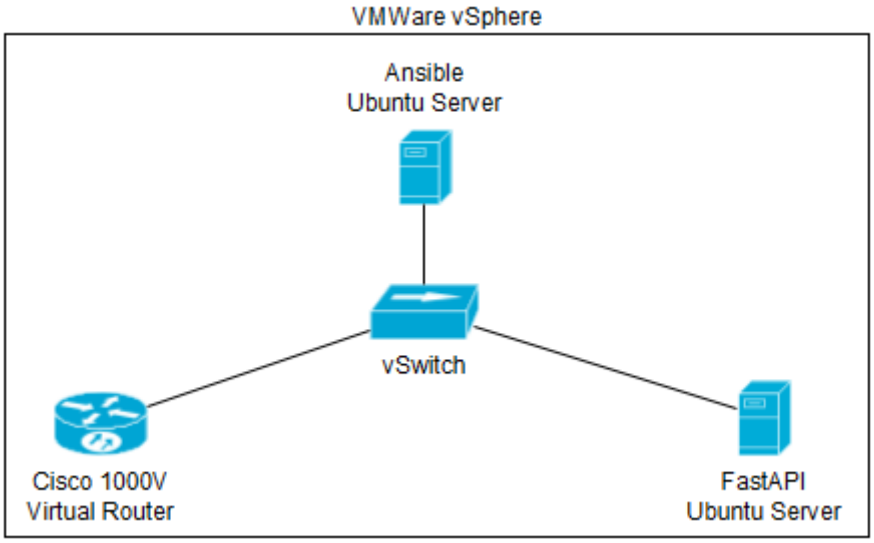

Figure 16: Test environment

4.9.2 Production Environment

Although, the production environment is different. Ansible AWX, must be placed into the infrastructure where it can reach every edge device and the Inventory. In this environment, Django is used as the Inventory with MySQL database behind it. Additionally, SSH reachable external nodes are connected to the infrastructure through the edge devices.

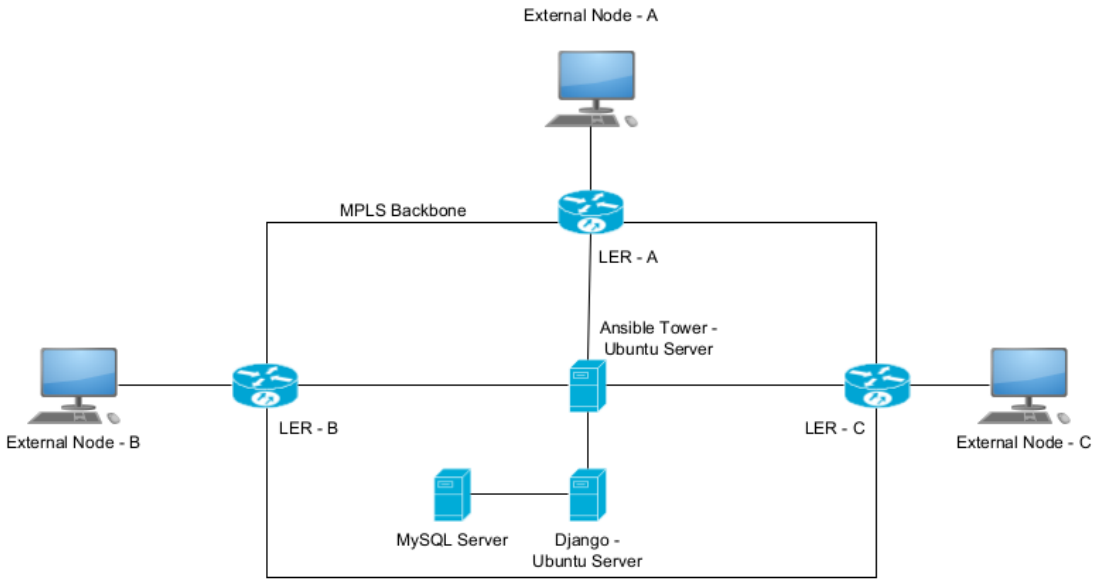

Figure 17: Production environment

5 Results

In this research, there are sub-objectives, such as comparing network scanners and describing the role and value of Nmap. Consider these as theoretical objectives, since the entire project would not use the right tool for the solution without these comparisons.

Based on the research conducted, it is known that there are two major groups of vulnerability scanners, Active and Passive vulnerability scanners. Network scanners are a smaller group within these vulnerability scanners. According to reports, Nessus, a market-leading AVS, is only able to cover about 38% of available CVEs. Therefore, additional vulnerability scanning is needed, which allows engineers to broaden their understanding of their infrastructure. According to Ron Gula [20], a PVS can be used in conjunction with an AVS for hardening network infrastructures. He also mentions, network scanners can be used as PVS in many cases. As a result, this research aimed to find the right network scanner to function as a PVS in the enterprise infrastructure. In section 3.4 of the thesis, a comparison has been conducted between OpenVAS and Nmap. OpenVAS is more efficent in terms of vulnerability scanning, however Nmap is more effective in terms of network scanning. As a result, for this research, Nmap is the right choice. The main goal, was to create a process for automatically scanning network segments on the ISP network. The author described the process and presented the solution in chapter 4.

5.1 Outcome

Closed loop automation is the basis of the proposed system. The author created a hybrid system that combines the advantages of closed loop automation with manual operations, since closed loop automation itself is quite a costly implementation. It means sometimes manual operation is required to restore the system to its desired state. Therefore the system has disadvantages in error handling, since unexpected errors and combinations may occur that the process controller does not yet understand. The system thus significantly lowers the risk of human error while still providing the opportunity for manual changes, and it has a lower implementation cost than a fully-automated system.

5.2 Capabilities

The system is capable of updating the content of the inventory, create or modify the ACL configuration on edge devices, start a network scan and understand its result. Moreover, it can check if the Inventory and the running configuration are consistent by comparing their content. It can understand the feedbacks, by checking if the result is either negative or positive and take necessary steps accordingly. In case of the database content is incomplete, the system can send the necessary prefixes through API. Additionally, if the running configuration is incomplete, the system can create or modify ACLs.

After connecting the components, three major use cases can happen. First, is to reserve a prefix in the database and the system autonomously create the necessary ACLs and validates it with a scan. Second, in case of the manual configuration change, the system validates the config and sends missing prefixes for the inventory if it is incomplete. Lastly, network scan can be conducted to check if each iACLs are working as expected.

Therefore, the system accomplishes much more than just checking for exposed hosts with Nmap. It has validation and self-healing capabilities.

5.3 Usability

As the thesis title states, the scanning can cover different network segments, not just the MPLS backbone. Due to the fact that the created process is not tied to the MPLS infrastructure, it is possible. Edge routers are the prerequisite, since they play a crucial role in the process as a whole, if the goal is to create iACLs. In the event that the purpose of this process is not to manage iACLS, then this process may also be used to manage any ACLs and any router may be used. Implementation can be handled even within Small Office Home Office (SOHO) environments.

5.4 Security and performance

As the iACLs are in place and correct, which are validated by the content of the inventory and the network scan, the result is increased performance and security enhancements. It can be concluded, since if the network scan does not find any exposed hosts from the

external network, it means, those devices cannot be accessed, therefore conducting direct attacks are less possible. Moreover, by generating the ACLs, human errors are less likely. On the other hand, by dropping unverified traffic on the edge of the infrastructure, the core devices does not have to process that traffic. As a result, less unwanted packet enters the internal network, so increased performance can be expected on the core devices.

5.5 Network outages

As the management of access list is automated, network outages can be rectified in time, since it has a strict validation procedure. In case of error, the system can notify the enginers or tries to recover into desired state. [33] According to this article, 80% of unplanned network outages are caused by changes to network configurations. These outages cause $46 million in damage each year, and only 3% are rectified before they cause disruptions. It is therefore, an automated validation mechanism should be implemented into most infrastructures.

5.6 Cost

This project relies on only open-source tools and libraries, that are, Ansible AWX, Nmap, Django, and MySQL. The whole system can be implemented without any cost, which can be an advantage over other solutions, like software-defined implementations.

6 Future works

In chapter 4, a solution that is proposed is a semi-automated system and in chapter 5 some of its disadvantages were described. Thus, error handling and process calling mechanisms have difficulties in this system, since it has to be programmed to handle every scenario. There is a straight line, where the starting point is a manually managed system, and the ending point is a fully-automated system. The semi-automated system sits in the middle of this straight line. In other words, the more developers adjust the system and prepare it for unexpected errors, the closer the system gets to being fully-automated.

Nevertheless, every company must find the most cost-effective solution, since in many cases, a fully-automated system is not even required. Still, logging, security and availability solutions should be a high priority. These aspects were out of the scope of this research, however it is relevant to discuss them in this section.

6.1 Development

The scope of the research was to construct the scan process that will provide the base for the described Tier 2 ISP, where the author spent his internship. For this reason, the case study of this research is not fully done yet, since the actual code is not yet implemented into the production environment. However, an alpha version can be found in Appendix 2, that was used for testing purposes and is still under development.

6.2 Logging

Logging solutions must be implemented into the system. Despite the fact that Ansible provides feedbacks, sometimes its log messages do not explain the issue. To teach the system in the future, the process path must be logged. This means that the logs should indicate which step failed. As an example, Ansible may try to push configurations into Juniper edge devices using a Cisco or Huawei module. In this case, the log message simply tells the device that the configuration is invalid. In the event that the SSH connection is logged, it is immediately apparent that the configuration has been generated using the wrong module, since the IOS XR connection log message differs from the Junos

log message. By identifying the problem, the system can be prepared to handle it correctly in the future.

As part of the configuration process, configuration files can also be saved for further inspection to determine what went wrong. It is also important to record every action taken by the Inventory so that engineers can eventually review the put, post, and delete requests and take appropriate action if necessary. Specially, since it is hybrid system, so its functionality must be eventually supervised.

6.3 Security

It is extremely important to protect the Ansible AWX server, since it has access for most of the devices and has permission to change their configurations. Moreover, hardcoding any login credentials into playbooks or variables is must be avoided at all cost. It should be set up on every possible device to use public key authentication and a separate user should be created for Ansible. This can also be automated where Ansible puts its key file on each device and creates its own user.

Despite being able to reach external networks, this server must only be accessible by the external hosts that start the scan, which is why an ACL is needed for an extra layer of security.

It is also critical to protect the Inventory, since it contains sensitive information about the infrastructure. Even though if access lists are in place and attackers cannot directly attack the Ansible AWX server, however other attack vectors can be used, such as deleting a prefix with an HTTP delete request. If that's the case, Ansible will also delete the iACL creating a security hole in the infrastructure. Additionally, it is also important to give reasonable expiration times for the API tokens, since API requests can cause significant damage if used for attacks.

6.4 Availability

The least significant aspect of all is availability, since manual configuration is available if an urgent change needs to be made. Nevertheless, availability is vital to the efficient

functioning of the system. Hence, redundancy between servers should be provided, and it is mandatory not to run all services on the same server. If one server is down, all services can go down. In addition, Netbox is a source of truth and other services rely on it, so high availability is a must. Having separated the servers, a redundancy solution needs to be implemented between them to handle failovers. Furthermore, creating backups, such as snapshots in a virtual environment, is a good practice.

6.5 Connecting services

The main idea of the enterprise was to use a real-time and federated source for every automation process, which is the SSOT. By connecting each automation processes, they use up-to-date information. It is efficent, since many departments can use different data source and not update them correctly. When the SSOT fully developed, it can serve as the basis for automation of other processes, such as, managing firewall access policies, summarise routes, create reports and so on. Using real-time and federated data source is an essential part of network automation.

7 Conclusion

Security, economics, public health and safety rely heavily on Internet services and Telecommunication. Network auditing is an important proactive approach to mitigate risks. Since auditing a network can be difficult in a large network, automated methods must be implemented. Due to its advanced network scanning capabilities and its ability to act as a passive vulnerability scanner, Nmap was selected as a suitable network scanning tool in this study. However, active vulnerability scanners should also be used in every infrastructure. Closed loop automation is an approach that paves the way for self-driving networks. This research was created based on this approach, and a semi-automated process was developed for handling iACLs on edge devices, and then validating their configurations and the central database content. As a result, the created process is free, increases network security and performance, helps network auditing and lowers the cost of network outages. There are still many future works available to advance the process into a fully-automated approach, including logging, security, and availability requirements. Moreover, connecting most of the automation solutions within the ISP to the inventory would be an excellent addition. As a member of the ISP, the author was involved in this project, and the process presented provides a basis for further development. This project's alpha version, made only for this study, can be found on the github link in the appendix.

Bibliography

[1] CISA, "https://www.cisa.gov," [Online]. Available: https://www.cisa.gov/communications-sector. [Accessed 29 December 2022].

[2] WhatIsMyIP, "WhatIsMyIP," [Online]. Available: https://www.whatismyisp.com/articles/what-is-an-isp. [Accessed 29 December 2022].

[3] M. Winther, "Tier 1 ISPs: What They Are and Why Are Important," 2006.

[4] J. M. A. S. M. R. N. B. Ashley Flavel, "Computer Communications," *BGP route prediction within ISPs*, vol. 33, no. 10, pp. 1180-1190, 2010.

[5] Cisco, "Cisco," 3 August 2007. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/net_mgmt/vpn_solutions_center/2-0/mpls/provisioning/guide/PGmpls1.html#wp1018933. [Accessed 29 December 2022].

[6] Cisco, "Cisco Annual Internet Report (2018–2023) White Paper," Cisco, 2020.

[7] Cloudflare, "Cloudflare," [Online]. Available: https://www.cloudflare.com/learning/cdn/what-is-a-cdn/. [Accessed 29 December 2022].

[8] G. J. Ed, "https://www.rfc-editor.org/," September 2004. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3871. [Accessed 29 December 2022].

[9] H. Holm, "Performance of automated network vulnerability scanning at remediating security issues," in *Computers & Security*, 2012, pp. 164-175.

[10] Lanner-America, "Lanner-America," [Online]. Available: https://www.lanner-america.com/blog/stateless-vs-stateful-packet-filtering-firewalls-better/. [Accessed 29 December 2022].

[11] Cisco, "Cisco," 21 October 2008. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/ip/access-lists/43920-iacl.html. [Accessed 29 December 2022].

[12] R. Hat, "github.com," Red Hat, [Online]. Available: https://github.com/ansible/awx. [Accessed 29 December 2022].

[13] "geeksforgeeks," 22 February 2022. [Online]. Available: https://www.geeksforgeeks.org/difference-between-django-and-node-js/. [Accessed 29 December 2022].

[14] T. Contributor, "techtarget," [Online]. Available: https://www.techtarget.com/whatis/definition/single-source-of-truth-SSOT. [Accessed 29 December 2022].

[15] netbox-community, "github," [Online]. Available: https://github.com/netbox-community/netbox. [Accessed 29 December 2022].

[16] RedHat, "RedHat," 1 June 2022. [Online]. Available: https://www.redhat.com/en/topics/automation/what-is-a-webhook.

[17] netbox-community. [Online]. Available: https://docs.netbox.dev/en/stable/.

[18] Nessus, "Tenable," Tenable, [Online]. Available: https://www.tenable.com/products/nessus. [Accessed 29 December 2022].

[19] H. Ecik, "2021 International Conference on Information Security and Cryptology," *Comparison of Active Vulnerability Scanning vs. Passive Vulnerability Detection,* pp. 87-92, 2021.

[20] R. Gula, "Passive Vulnerability Detection," 9 September 1999. [Online]. Available: https://vodun.org/papers/net-papers/gula_passive_vulnerability_detection.pdf. [Accessed 29 December 2022].

[21] nmap, "nmap.org," nmap, [Online]. Available: https://nmap.org/book/man.html. [Accessed 29 December 2022].

[22] S.-H. a. K. Y. R. a. B.-h. R. Sun-young Im and Shin, "Performance evaluation of network scanning tools with operation of firewall," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2016, pp. 876-881.

[23] OpenVAS, "openvas.org," [Online]. Available: https://www.openvas.org/. [Accessed 29 December 2022].

[24] N. Y. Jhala, *Network Scanning & Vulnerability Assessment with Report Generation,* 2014.

[25] S. B. Jerome Marc, "Redhat," 28 September 2021. [Online]. Available: https://www.redhat.com/en/blog/self-healing-infrastructure-red-hat-insights-and-ansible-automation-platform. [Accessed 29 December 2022].

[26] I. Wigmore, "techtarget," [Online]. Available: https://www.techtarget.com/whatis/definition/idempotence. [Accessed 29 December 2022].

[27] D. Mellado, "docs.ansible.com," [Online]. Available: https://docs.ansible.com/ansible/latest/collections/junipernetworks/junos/junos_acls_module.html. [Accessed 29 December 2022].

[28] N. Chakraborty, “docs.ansible.com,” [Online]. Available: https://docs.ansible.com/ansible/latest/collections/cisco/iosxr/iosxr_acls_module.html#ansible-collections-cisco-iosxr-iosxr-acls-module. [Accessed 29 December 2022].

[29] Q. Pan, “docs.ansible.com,” [Online]. Available: https://docs.ansible.com/ansible/latest/collections/community/network/ce_acl_module.html. [Accessed 29 December 2022].

[30] Cisco, “github,” [Online]. Available: https://github.com/cisco/cisco-mibs/blob/main/ME1200-MIBS/15.4_3_SN/ME1200-ACL-MIB.mib. [Accessed 14 December 2022].

[31] FastAPI. [Online]. Available: https://fastapi.tiangolo.com/. [Accessed 29 December 2022].

[32] VMware, “docs.vmware.com,” [Online]. Available: https://docs.vmware.com/en/VMware-Smart-Assurance/10.1.0/ip-manager-delopment-guide-101/GUID-C7E7752E-D122-44EA-83DE-33F690B22313.html.

[33] D. Krisha, “anutanetwork,” 22 January 2019. [Online]. Available: https://www.anutanetworks.com/5-real-world-use-cases-of-closed-loop-automation/.

[34] U. Team, “https://www.upguard.com/breaches/out-of-pocket-how-an-isp-exposed-administrative-system-credentials,” UpGuard, 2019.

[35] D. Cramer, “hackertarget,” 22 August 2012. [Online]. Available: https://hackertarget.com/nessus-openvas-nexpose-vs-metasploitable.

[36] M. Martinsson, “What is closed-loop automation?,” 9 April 2019. [Online]. Available: https://www.ericsson.com/en/blog/2019/4/what-is-closed-loop-automation.

[37] R. H. Enterprise, “ansible.com,” [Online]. Available: https://www.ansible.com/products/awx-project/faq. [Accessed 29 December 2022].

Appendix 1 - Non-exclusive licence for reproduction and publication of a graduation thesis

I, Dominik Zoltan Kovacs,

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis Automated Vulnerability Scanning of the Network Segments of an Internet Service Provider, supervised by Tauseef Ahmed and József Tanárki
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

05.01.2023

Appendix 2

Github link: dokova/ansible_thesis_project (github.com)