TÉCNICO LISBOA

Large Scale VPNs

Ricardo Silva Bandeira

Thesis to obtain the Master of Science Degree in

Telecommunications and Informatics Engineering

Supervisor: Prof. Rui Jorge Morais Tomaz Valadas

Examination Committee

Chairperson: Prof. Ricardo Jorge Fernandes Chaves

Supervisor: Prof. Rui Jorge Morais Tomaz Valadas

Member of the Committee: Prof. Miguel Nuno Dias Alves Pupo Correia

November 2021

Abstract

The security of the network infrastructure is one of the key aspects of Internet security. One of the most important security challenges today is being able to access servers and resources located in private networks. This MSc Dissertation has as main objective to make a deep study of large scale virtual private network (VPN) technologies, especially those based on IKEv2.

In the topic of large scale VPNs, we approached two main subjects: FlexVPN, which is a conventional VPN solution integrating AAA, load balancing, monitoring, and fault tolerance, and SDWAN, which is a software-defined solution to deploy and manage VPNs. Public key infrastructures (PKI) were also studied because they add scalability and security to these solutions. Routing security was also studied because routing protocols make the base of networking, and securing these protocols makes the solutions that run on top of them safer.

Using a virtual environment based on GNS3, a network emulation tool, the Dissertation work demonstrates features of FlexVPN, the focus being on load balancing, AAA, PKI Integration and fault tolerance. Three SDWAN solutions from different vendors were also studied and explored, using their own Web Interfaces. Regarding PKIs, two infrastructures were implemented, one simple, single-level PKI, and another two-level, more complex, infrastructure to demonstrate PKIs, and in relation to routing security, a RIPv2 attack and countermeasures were explored.

This MSc dissertation was supported by Instituto de Telecomunicações.

Keywords

VPN; FlexVPN; AAA; EAP; PKI; Routing Security; GNS3; Kali Linux; Scapy.

Resumo

A segurança da infraestrutura de rede é um dos aspetos fundamentais da segurança da Internet. Um dos desafios de segurança mais importantes hoje em dia é conseguir aceder a servidores e recursos localizados em redes privadas. Esta Dissertação MSc tem como principal objetivo fazer um estudo profundo de tecnologias de rede privada virtual em larga escala (VPN), especialmente as baseadas em IKEv2.

No tema das VPNs em larga escala, abordámos duas matérias principais: a FlexVPN, que é uma solução convencional de VPN que integra AAA, balanceamento de carga, monitorização, e tolerância a falhas, e a SDWAN, que é uma solução definida por software usada para implementar e gerir VPNs. As infraestruturas de chaves públicas (PKI) também foram estudadas porque adicionam escalabilidade e segurança a estas soluções. A segurança dos protocolos de encaminhamento também foi estudada pois estes protocolos formam a base da infrastrutura de rede, e verificar a segurança destes protocols faz com que as soluções que funcionam em cima deles sejam mais seguras.

Utilizando um ambiente virtual baseado no GNS3, uma ferramenta de simulação de rede, o trabalho desta Dissertação demonstra características da FlexVPN, sendo o foco na distribuição de carga, AAA, Integração PKI e tolerância a falhas. Três soluções SDWAN de diferentes fornecedores também foram estudadas e exploradas, através das suas próprias interfaces Web; No que diz respeito aos PKI, foram implementadas duas infraestruturas, uma simples e de nível único, e outra infraestrutura de dois níveis, mais complexa para demonstrar PKIs e como funcionam. Em relação à segurança do encaminhamento, foi explorado um ataque RIPv2 e contramedidas.

Esta dissertação MSc foi apoiada pelo Instituto de Telecomunicações.

Palavras-Chave

VPN; FlexVPN; AAA; EAP; PKI; Routing Security; GNS3; Kali Linux; Scapy.

Contents

Image Index

Table Index

Acronyms

AAA	Authentication Authorization Accounting
AES	Advanced Encryption Standard
AH	Authentication Header
BGP	Border Gateway Protocol
CA	Certificate Authority
CLB	Cisco Load Balancing
DH	Diffie-Helmann
DMVPN	Dynamic Multipoint Virtual Private Network
EAP	Extensible Authentication Protocol
ESP	Encapsulating Security Payload
GDOI	Group Domain of Interpretation
GETVPN	Group Encrypted Transport Virtual Private Network
HSRP	Hot Standby Router Protocol

IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IP	Internet Protocol
IPSec	Internet Protocol Security
ISAKMP	Internet Security Association and Key Management Protocol
NAT	Network Address Translation
NHRP	Next Hop Resolution Protocol
OSPF	Open Shortest Path First
PKI	Public Key Infrastructure
PSK	Pre-shared Key
RA	Registration Authority
RADIUS	Remote Authentication Dial In User Service
RIP	Routing Information Protocol
RPKI	Resource Public Key Infrastructure
RSA	Rivest-Shamir-Adleman
SA	Security Association
SHA	Secure Hash Algorithms
SPI	Security Parameter Index
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VPN	Virtual Private Network

1. Introduction

1.1 Motivation

Network security and secure access is one of the Internet's most pressing problems. One of the most important things today is being able to access servers and resources located in private networks. For example, remote workers that need to connect to their workplace network, or students that want to connect to their college network to access college services. For this to be possible, the network to be accessed needs to allow connections from the Internet, but that comes with the risk of network attacks. If a network is not secure when allowing access to someone, its activity can be seen, inspected, and possibly changed by malicious actors. These malicious actors, hackers hoping to steal information or attack computer systems, try to explore vulnerable networks and systems. Because of this, there's an urgent need to secure the connections over the Internet.

VPNs solve this problem. They grant access to networks, and can be used with a number of tools and protocols to create connections to private networks, or for several networks to be connected as one over the Internet, with no one being able to inspect or change the traffic. With all the tools and VPN technologies, and solutions to deploy them, there's no reason for a large network, that needs to be accessed, to not allow secure connections from the Internet.

1.2 Objectives

This dissertation has as main objective to make an in-depth study of the most important and most recent virtual private network technologies (VPNs), and Software-Defined WAN (SDWAN), namely those based on IKEv2. We will study two approaches differing on whether the configuration and management are distributed or centralized: conventional VPNs and SDWAN. SDWAN was not in the initial objectives for this dissertation but was added due to its relevancy to the topic.

The aim is also to study and demonstrate public key infrastructures, which are used to support VPNs by performing key distribution, and to study routing protocols, which form the basic infrastructure of networking, and therefore, of VPNs

Regarding the first objective, FlexVPN technology will be studied, namely aspects related to FlexVPN client/server architecture, EAP authentication, AAA, load balancing, failure resilience, and monitoring. FlexVPN will be compared with previous technologies, namely DMVPN and GETVPN, considering their integration with IKEv2. SDWAN solutions and their way to deploy VPNs will also be studied and there will be a comparison between different SDWAN solutions, and a comparison between SDWAN in general and conventional VPNs. Finally, a detailed study of PKI technologies will be made, including their design and how they can be integrated to improve VPNs and SDWAN

solutions. Finally, attacks affecting routing protocols will be studied, namely RIP, as well as possible countermeasures.

Most studies related to this dissertation will be carried out in a virtual environment based on GNS3. In certain cases we will use proprietary solutions and infrastructures for demonstrations. For the study of different technologies, network scenarios will be designed as close as possible to the reality of organizations/companies. Security attacks use existing tools (for example, Kali Linux) or programmed in Python (using, for example, scapy).

The final objective is to compare the solutions approached in the dissertation and understand when they should be used instead of other technologies.

1.3 Organization of the Document

This dissertation is organized as follows. Chapters 2, 3, 4, 5, and 6 cover the theoretical concepts and techniques about VPNs, FlexVPN, PKI, SDWAN, routing protocols and network attacks that will be used in the dissertation. Chapters 7 through 15 describe the experiments developed so far, mentioning the ways and tools to perform an attack as well as the used countermeasures to prevent it.

Chapter 16 presents the conclusion for the MSc Dissertation.

2. VPNs

A virtual private network (VPN) allows secure connections to its private network, through a public network. It doesn't require physical connectivity to the remote sites, only a virtual connection. This means users can exchange data securely across unsecure networks, as if their devices were directly in the private network.

This is done by creating a connection between two, the VPN, and the user/client, this tunnel is then used to transport every packet from one end to the other. Although encryption was not always supported in the early days of VPNs, nowadays almost every VPN technology uses IPsec to handle confidentiality, and every data packet passing through the tunnel is encrypted before leaving, and decrypted when it arrives. It's also possible for the VPN to require authentication before allowing someone to connect to it and access the private network.

There are two basic ways of using a VPN:

- Remote-access: When there isn't a fixed connection, and a user can connect from their host, in any network with access to the Internet.
- Site-to-site: When the VPN connects different sites in a shared network. In this way all hosts in the network of one site can access the network on the other site, and the hosts aren't even aware of the VPN.

Remote-access is mostly used by employees that need to use some resource on their company network, or students that need to use the school's network. As long as the user has the correct credentials, he can connect from any network.

Site-to-site is used to connect networks in different locations. It can connect multiple client sites or multiple company sites to the main network. In this case, a VPN tunnel is established between two gateways (usually two routers), one is the main gateway (Hub), which waits for incoming connections, and the other is a Spoke, which initiates the connection. Even with multiple sites connected, there is only one Hub and the remaining are Spokes. After receiving a data packet from the VPN peer, the data packet will be validated, unencrypted, and routed to its destination. Both Site-to-Site and Remote Access are exemplified in Figure 1.

There are a lot of VPN technologies being used these days, but in this document, the focus will be on technologies using IKEv2.

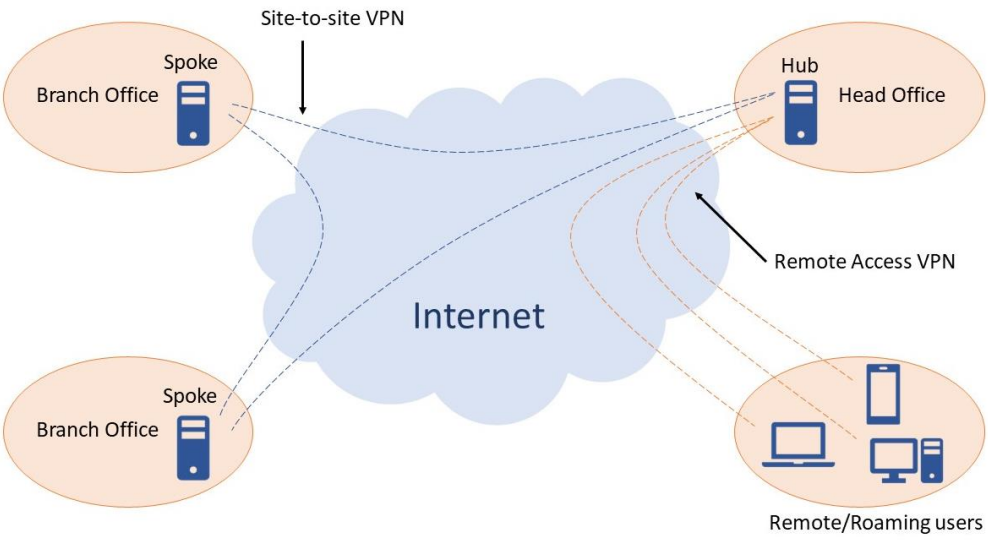


Figure 1 - Remote Access and Site-to-Site VPNs

2.1 IPsec

IPsec[1] is an IETF standard for security in communications. In recent VPN technologies, IPsec is used to secure the tunnel between peers.

When trying to establish a connection, first, the VPN peer will establish an IPsec security association (SA), which is stored in each peer's database and contains the security parameter index (SPI), the identity of the other peer, the sequence number currently being used (to stop replay attacks) and what cryptographic services (encryption, authentication, and integrity algorithms) are being used for the connection. This information is mandatory for the maintenance of the connection.

To use IPsec, there are two protocols, AH (Authentication Header) and ESP (Encapsulating Security Protocol):

- AH provides authentication and integrity protection only, it does not offer encryption.
- ESP offers authentication, encryption, and integrity protection.
- It's possible to use both AH and ESP at the same time.

There are also two modes of operation, Tunnel Mode and Transport Mode. In Transport Mode. The IPsec information is inserted inside packet, while on Tunnel Mode, the information encapsulates the packet, and a new IP header is added, as shown in Figure 2.

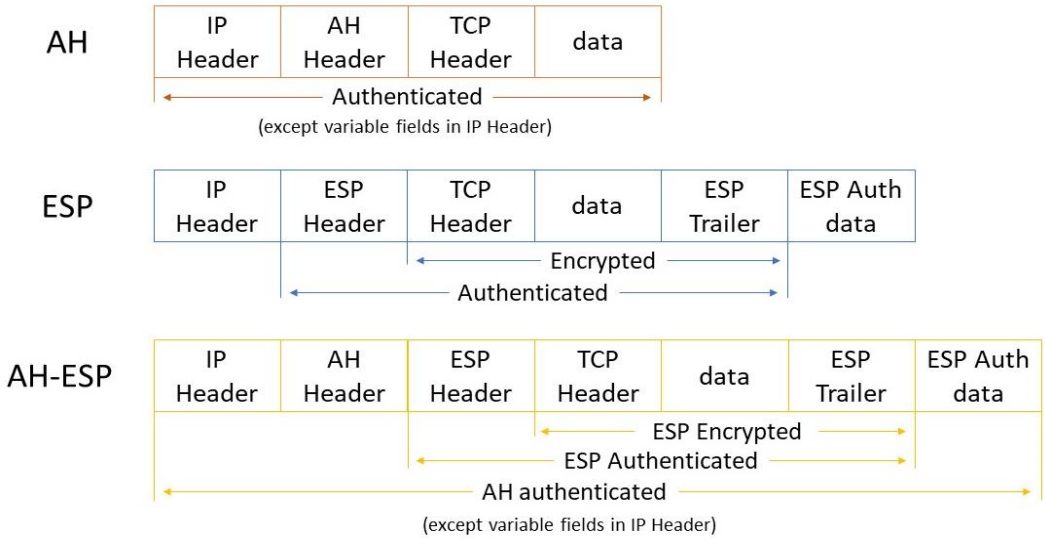


Figure 2 - Packet structure in Transport Mode

The IPsec information is added between the IP header and the rest of the packet. In this way, usually only the payload of the IP data packet is encrypted or authenticated. The routing remains unchanged because the IP header is neither modified nor encrypted. However, when the authentication header (AH) is used, the IP addresses cannot be modified through NAT (Network Address Translation), as this invalidates the hash value used to assure integrity.

Figure 3 shows how the packets are built in Tunnel Mode.

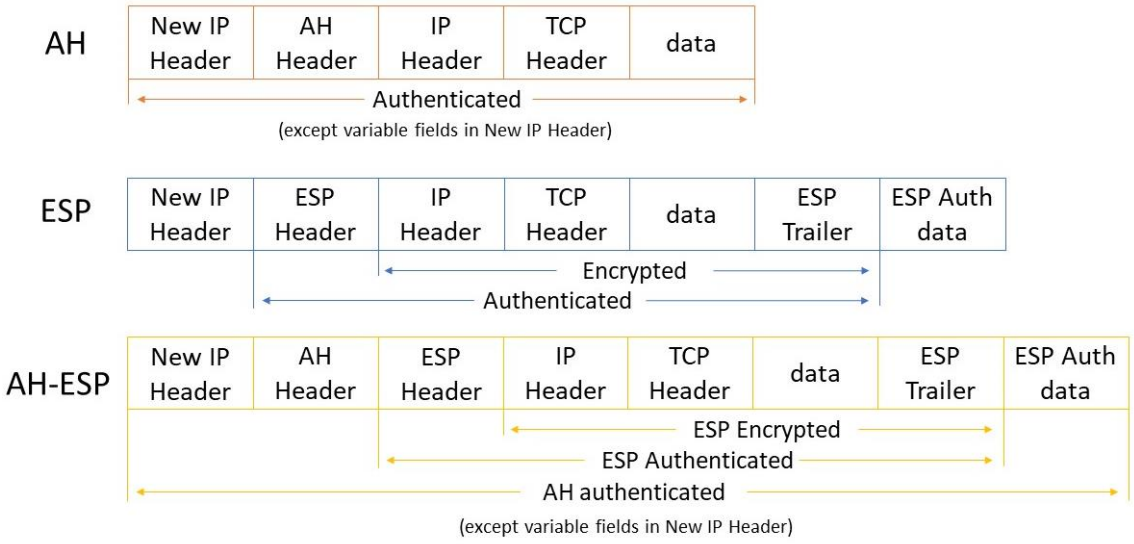


Figure 3 - Packet structure in Tunnel Mode

In tunnel mode, the original IP packet remains unchanged, and a new IP header and IPsec information are added to the original packet. In this way, the entire packet is encrypted and/or authenticated. It is then encapsulated in a new packet with a new IP header.

IPsec provides certain security functions, and some algorithms that allow for different implementations. These security functions are divided in:

- Confidentiality (encryption algorithms)

- Integrity (hashing algorithms)
- Authentication (using IKE or IKEv2)
- Secure-key exchange (using Diffie-Hellman)

Some of the possible algorithm choices are shown in Figure 4. It's possible, and mandatory, to choose one option for each field, except for Confidentiality in case AH has been chosen.

One possible implementation is:

- IPsec Protocol: ESP
- Confidentiality: AES
- Integrity: SHA
- Authentication: RSA
- Diffie-Hellman: DH5

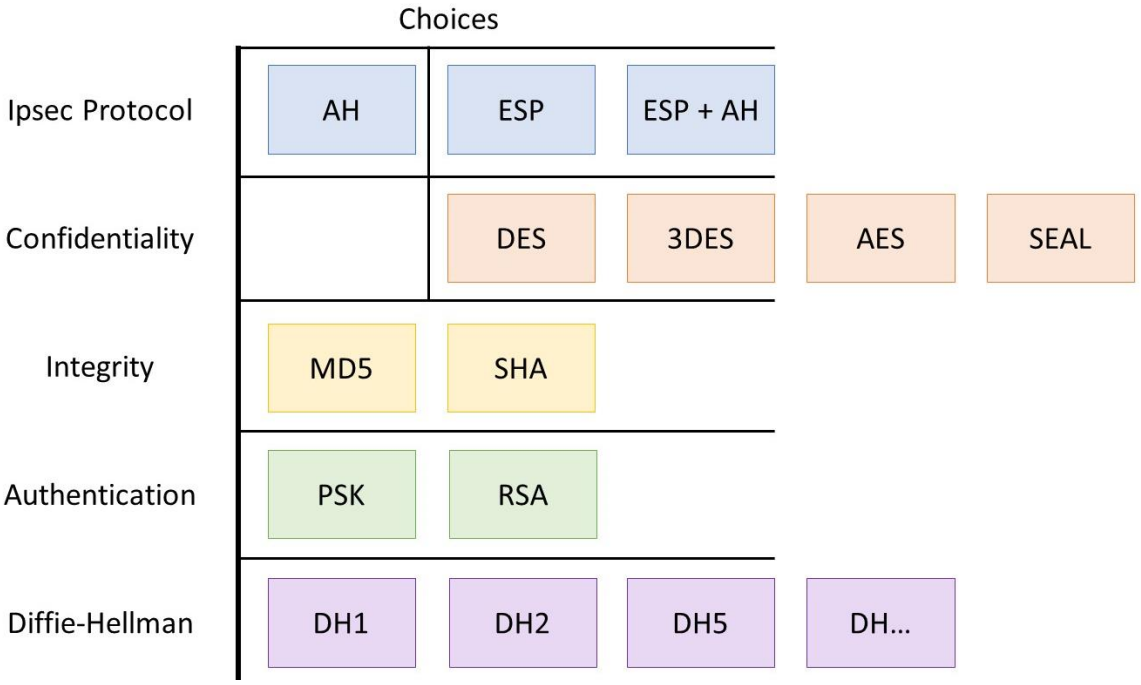


Figure.4 - IPsec framework

2.2 IKEv1 and IKEv2

IKEv2[2] (Internet Key Exchange version 2) is the successor to IKEv1, and it's a key management protocol. It handles VPN request and response actions. IKE uses the ISAKMP[3] framework to implement key exchange protocols. ISAKMP is a framework used to define processes for authenticating peers, creating, and managing security associations.

IKEv1 and IKEv2 automatically negotiate and establish an ISAKMP SA (Security Association), used to then negotiate and establish an IPsec SA securely. To do this, first, the client and the server are authenticated, and then IKE decides on which encryption methods, and what secret keys, will be used to establish the IPsec SA. This key negotiation is divided into two parts, Phase 1 and Phase 2.

The purpose of the IKE phase 1 is to establish a secure and authenticated communication channel by using the Diffie-Hellman algorithm to generate a key for encrypting further IKE communications. The initiator sends an SA proposal with the authentication type required, the encryption method to use to protect the message exchange, and the Diffie-Hellman group. This negotiation results in a single ISAKMP SA between the two peers. A pre-shared key (shared key), signature or public key encryption can be used to perform authentication. Phase 1 runs in either Main Mode or Aggressive Mode.

In Main Mode, first the peers agree on the SA proposal, then they exchange nonces, Diffie-Hellman parameters and calculate a shared secret. After this is done, they encrypt their identity and the hash of the nonce with the shared secret and send it to the peer. This way the identity of both peers is protected. The Main Mode needs to exchange 6 messages. The first four messages are unencrypted, and the last two are encrypted. This exchange can be seen in Figure 5.

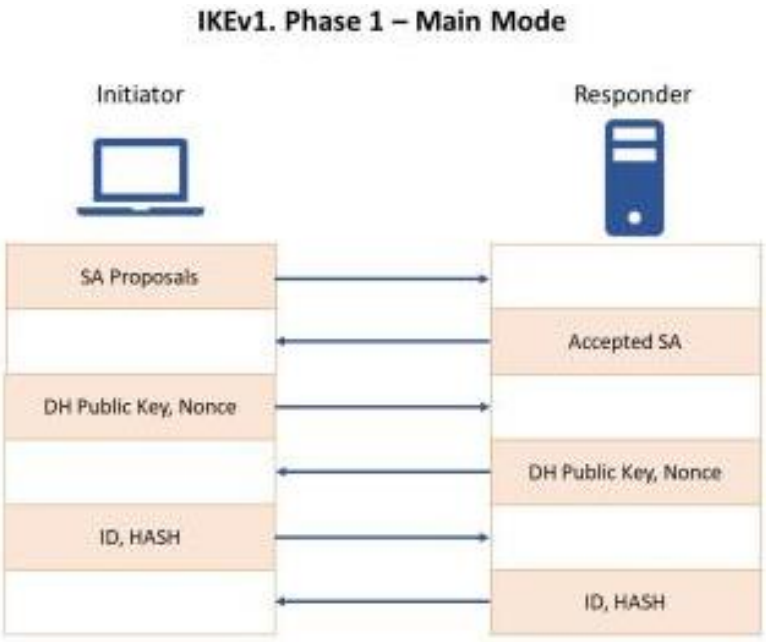


Figure 5 - IKEv1 Main Mode

In Aggressive Mode, the same information is exchanged, but as fast as possible. The SA proposal, Diffie-Hellman Key, the nonce, and the identity are sent in the same message, unencrypted.

This way, only three messages are exchanged, which means that the Aggressive mode is faster than the Main Mode. However, in Aggressive Mode the peer identities are not protected because they're sent in plain text.

This exchange can be seen in Figure 6.

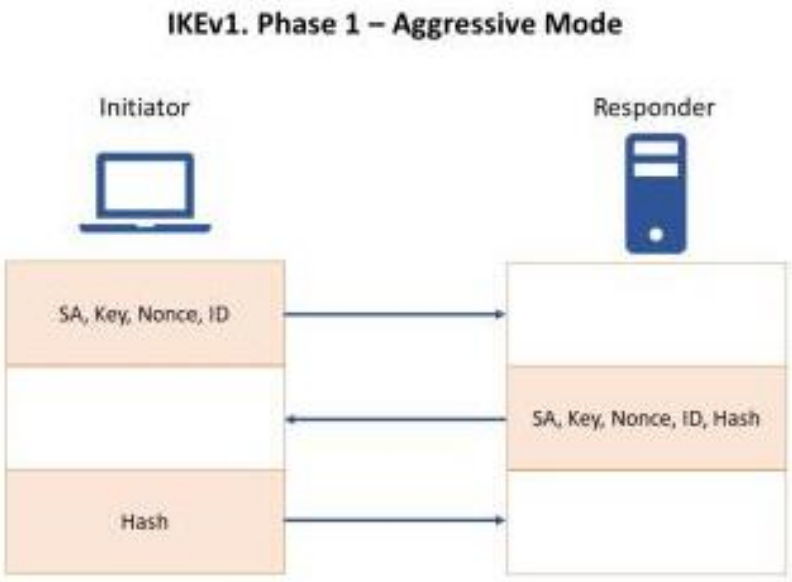


Figure 6 - IKEv1 Aggressive Mode

During IKE phase two, the IKE peers use the secure channel established in Phase 1 to negotiate Security Associations on behalf of other services like IPsec. This negotiation results in at least two one-way security associations (one inbound and one outbound). Phase 2 only runs in Quick Mode, which exchanges three messages as shown in Figure 7.

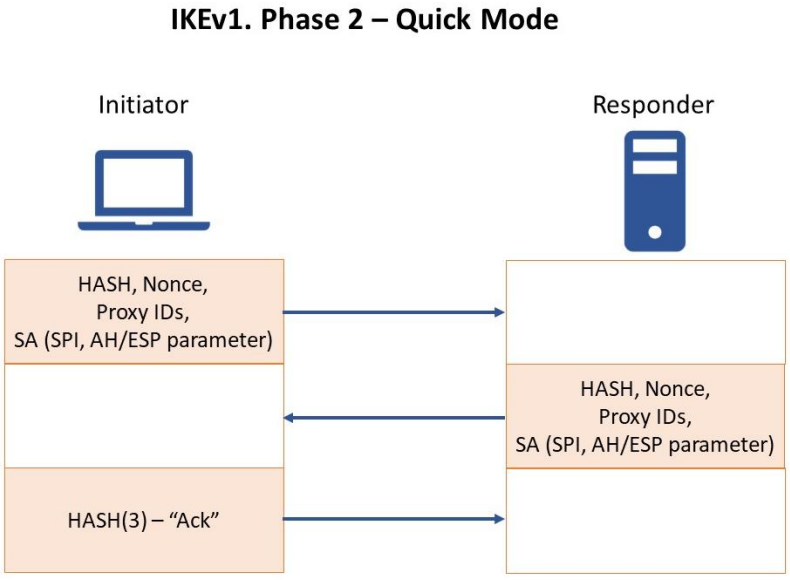


Figure 7 - IKE Quick Mode

This means that if Main Mode is used, a total of 9 messages are exchanged, and if Aggressive mode is used, a total of 6 messages are exchanged.

From IKEv1 to IKEv2, there have been some improvements. The following are some of the main differences between IKEv2 and IKEv1:

- IKEv2 needs less messages to establish a secure connection, which means that it's faster than IKEv1
- IKEv2 offers support for remote access by default, thanks to its EAP authentication.
- The IKEv2 VPN protocol can use asymmetric encryption, which means that for someone to be able to understand the IKE communication, the two keys would be needed, making it possible for IKEv2 to be more secure than IKEv1(if the encryption is correctly configured).
- Unlike IKEv1, IKEv2 can detect if a VPN tunnel is working or not. This allows to automatically re-establish a dropped connection.
- IKEv2 supports more encryption algorithms than IKEv1.
- IKEv2 offers better reliability through sequence numbers and acknowledgements.
- IKEv2 supports the Redirect notification

Figure 8 shows the message exchange in IKEv2.First the peers exchange an SA_INIT message that contains cryptographic algorithms, nonces, and Diffie-Hellman parameters. Then the IKE_AUTH message authenticates the previous messages, has the encrypted identities and certificates. This message is encrypted, which means that the identities are protected. With this message the peers establish an SA, having exchanged only four messages, compared with IKEv1's 6 (in case of Aggressive Mode) or 9 (in case of Main Mode) messages.

A redirect notification can be sent either in the SA_INIT or the IKE_AUTH. If this notification is sent, an IP address for another gateway is sent in the same message.

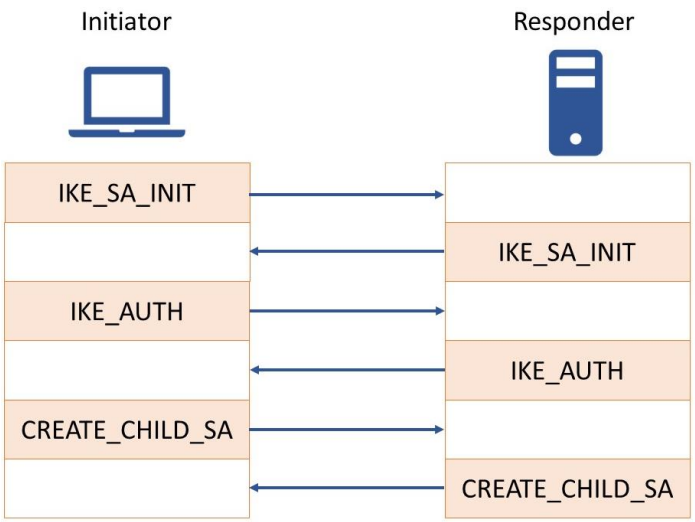


Figure 8 - IKEv2 Message Exchange

2.3 Establishing a connection

To establish a VPN connection with IPsec, the first step in the process is when the host recognizes that IPsec should be used to transmit certain data packets. This can happen by checking the IP address of the source or destination, against the policy configuration, to see if the traffic is considered "interesting" for IPsec purposes.

This traffic triggers the security policy, and the device sending the packet applies the necessary encryption and/or authentication to the packet. When it is determined that an incoming packet is "interesting", the host will verify whether the inbound packet is encrypted and/or authenticated.

The second step of the IPsec process is IKE Phase 1. It allows for the two hosts using IPsec to negotiate the policy they will use for the communication, authenticate themselves to each other, and initiate a secure channel between them. That secure channel is used to negotiate the way that IPsec will encrypt and/or authenticate the data to be sent.

The third step is IKE Phase 2, which is carried out through the secure channel created in IKE Phase 1. In this phase, the two hosts negotiate the type of cryptographic algorithms to use, as well as agreeing on secret keys to be used with those algorithms, this will assure Integrity, Authentication, and Confidentiality. There's also the exchange of nonces in this phase, randomly selected numbers used only once to provide session authentication and replay protection.

The fourth step of the IPsec connection is the actual data exchange across the newly created IPsec encrypted tunnel. From this point on, the two endpoints use the IPsec SAs setup in the previous three steps to encrypt and decrypt packets.

The last step is the termination of the IPsec tunnel, usually when the communication between the hosts is completed, the session has timed out or the pre-specified number of bytes has been passed through the tunnel. When the IPsec tunnel is terminated, the hosts will discard the keys used over that security association.

Figure 9 depicts this whole process of establishing an IPSec tunnel.

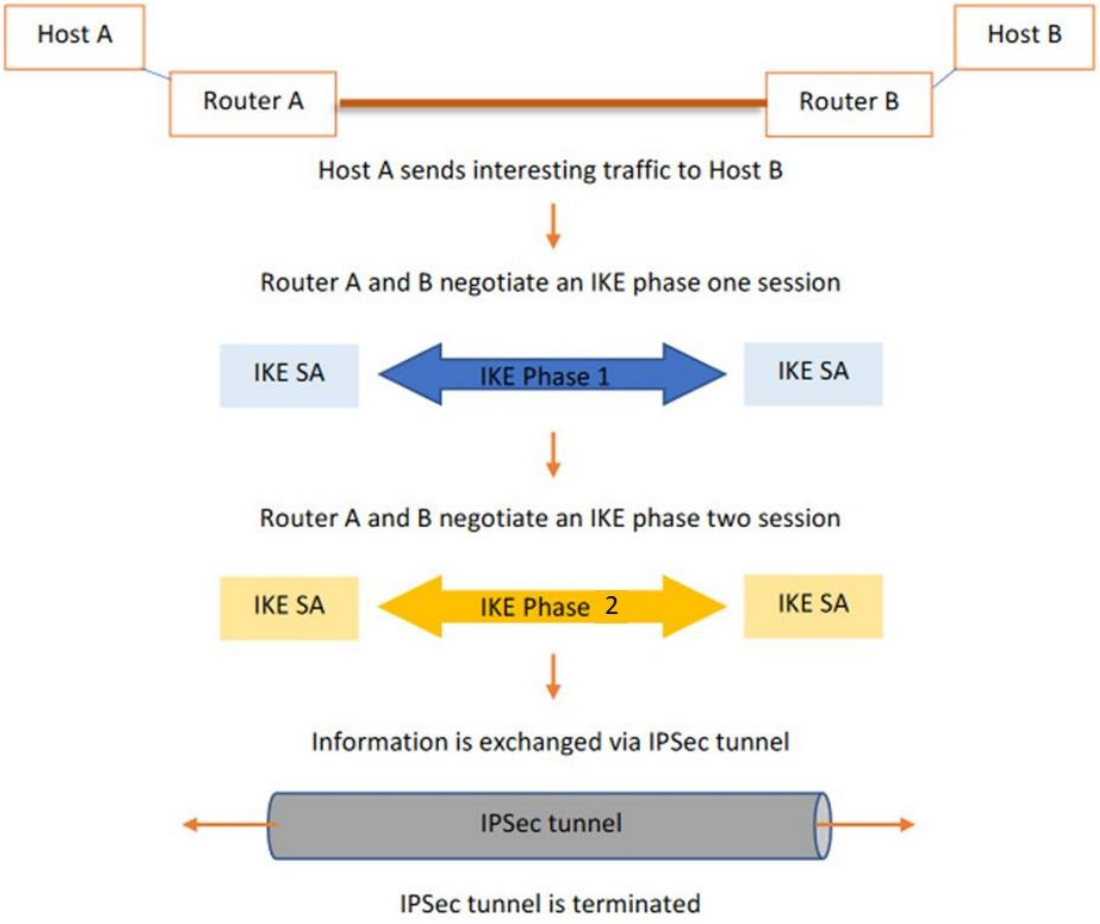


Figure 9 - Establishment of an IPSec tunnel

3. FlexVPN

FlexVPN[4] is an IKEv2-based VPN technology implementation. It makes it possible for large organizations to deploy VPNs to connect branch offices and remote users securely.

Over the years, VPN solutions have evolved a lot, each solution serving a different purpose and being developed independently, many being incompatible with one another. Each technology had different configurations, commands, and structures, only having in common the use of ISAKMP and IPSEC. FlexVPN was created to simplify the deployment of VPNs, to address the complexity of multiple solutions, and to integrate several solutions, into one technology.

FlexVPN has the following benefits:

- FlexVPN can be deployed either over a public internet or a private network.
- Designed for the configuration of both Site-to-Site and Remote-Access VPNs; one single FlexVPN deployment can accept both types of connection requests at the same time.
- It supports failover redundancy.
- It is compatible with any IKEv2-based clients by third-party VPN vendors.
- It supports Load Balancing

It allows for centralized policy control: VPN policies can be fully integrated with an authentication server and applied at a per peer basis.

3.1 Client/Server

FlexVPN Server is what is usually called the Hub, or headend, for the Remote-Access and Hub-and-Spoke VPN topologies approached in chapter 2.1. It can be configured to manage peer/client authentication, load balancing, fault tolerance support and monitoring tools. The FlexVPN Server is the responder to the VPN initiator (client).

The FlexVPN Client is a router-based VPN client, supporting FlexVPN features and deployment, it's mostly used in remote offices to provide VPN connectivity to a network. It uses a point-to-point tunnel interface with native IPsec encapsulation to connect to the FlexVPN Server. The Client can potentially connect to different VPN Servers, based on certain criteria. For example, if the main Hub is down, it can try to connect to a back-up one.

The Hub can also be configured to allow for Spoke-to-Spoke communication. Instead of all the traffic having to pass through the Hub, it can go directly from Spoke to Spoke. This is done by using the Next Hop Resolution Protocol (NHRP)[5].

NHRP is a protocol that can be used so that a device sending data to another device can learn the most direct route (the fewest number of hops) to the destination.

The process is according to the following steps:

Hub-Spoke connection

1. Spokes connect to Hub
2. Hub learns Spoke networks via routing protocol over Hub-Spoke tunnels
3. Hub advertises routes to all spokes

NHRP Redirect

4. Spoke to Spoke traffic goes through Hub
5. Hub sends NHRP Redirect indication to source spoke with destination spoke address

NHRP Resolution

6. The source Spoke initiates NHRP resolution via Hub to resolve destination Spoke
7. Hub forwards resolution request to destination Spoke
8. Destination Spoke receives resolution request, creates SA and tunnel to source Spoke
9. Destination Spoke sends resolution reply over Spoke-to-Spoke tunnel
10. Destination Spoke adds NHRP cache entry for source spoke

NHRP Shortcut

11. Source spoke receives NHRP resolution reply
12. Source spoke adds NHRP cache entry and shortcut route for destination spoke

After these steps, the FlexVPN Clients/Spokes can communicate between each other. Figure 10 shows a FlexVPN topology with a Server (Hub), many Clients (Spokes) and a Remote Access User.

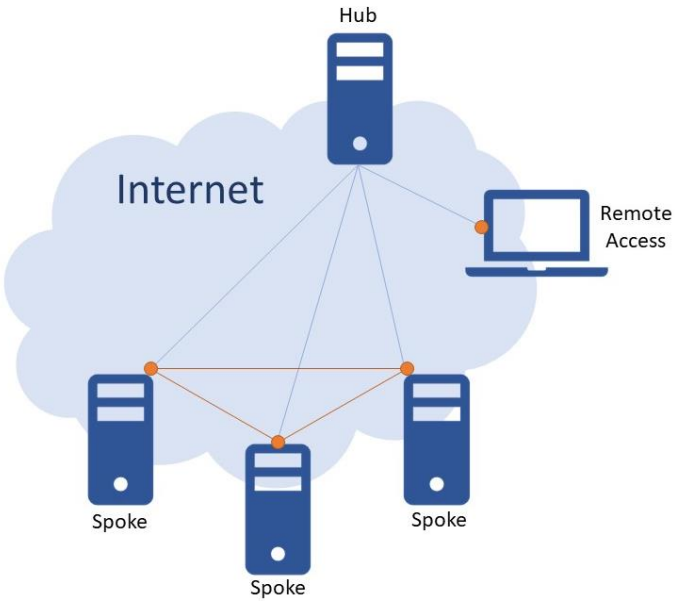


Figure 10 - A possible FlexVPN deployment

3.2 AAA

A network must be designed to control who is allowed to connect to it, when they are allowed to connect to it, and what they are allowed to do, all while being able to check everything that happened on the network. For this purpose AAA (Authentication, Authorization, and Accounting) services were created, providing the necessary framework to enable access security.

AAA security has three basic components:

- Authentication - Users must prove that they are who they claim to be. Authentication can be completed using username and password, challenge and response questions and other methods.
- Authorization - Authorization services specify which resources the user has access to and which operations the user is authorized to perform.
- Accounting - Accounting records what the user does, including what is accessed, the amount of time the resource is accessed, and any changes that were made.

Routers can be configured to use AAA to access a local username and password database. Using a local username and password database provides better security than a simple password. It is also a cost effective and easily implemented security solution for small organizations.

The local database method has some limitations. The user accounts must be configured locally. In a large enterprise environment that has multiple clients to manage, it can take time to implement and change local databases. A better solution is to have all devices refer to the same database of usernames and passwords from a central AAA server.

Most current AAA servers, used for network access, communicate with the routers using RADIUS (Remote Access Dial In User Service)[6].

The process is started by the router. It contacts the RADIUS server and transmits the request for authentication to the server, with the entered credentials. The server replies with an Access-Accepted message if the credentials are valid, otherwise it sends an Access-Rejected message to the client.

Figure 11 shows a network architecture employing an AAA server.

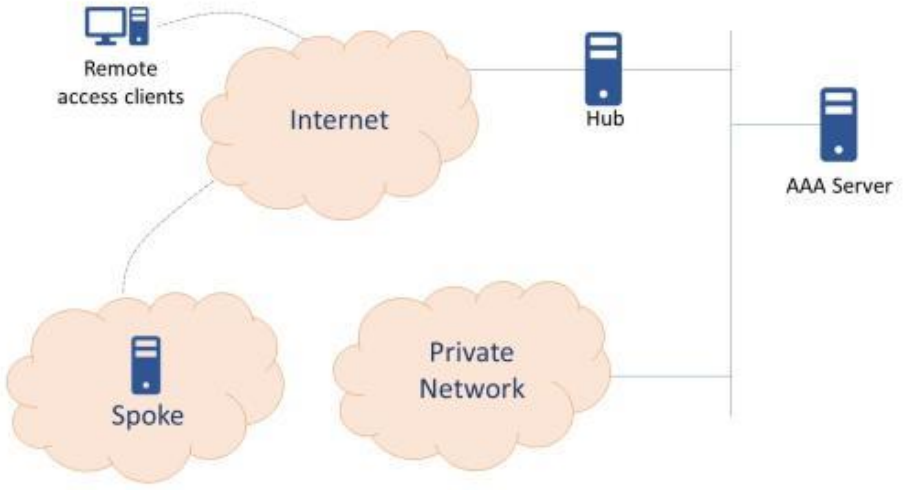


Figure 11 - AAA architecture

A very important protocol to mention when talking about AAA is EAP[7] (Extensible Authentication Protocol).

EAP is an authentication framework used mostly for wireless networks and is supported by FlexVPN. It's not a particular method for authentication, but a series of methods that make up the EAP framework.

For example, two commonly used methods are:

EAP-PSK[8] (Pre-Shared Key) – It uses a pre-shared key, or code, that's stored on both the client and the Access Point (WAP).

EAP-TLS[9] (Transport Layer Security) - This implementation only allows mutual certificated-based authentication through digital certificates.

The fact that it has a lot of methods available is a very significant advantage when choosing a protocol. Some methods might be more secure but require more infrastructure (EAP-TLS requires a full Public Key Infrastructure), while others might be less secure, but better for smaller enterprises (EAP-PSK only requires a password, or code).

EAP only defines message formats. Each protocol that uses EAP must define a way to encapsulate EAP messages within that protocol's messages. If the authentication server is a RADIUS server, then the messages are encapsulated in the RADIUS messages.

There are three basic entities to EAP authentication:

- Supplicant: The entity to be authenticated.
- Authenticator: The Access Point or Router.
- Authentication Server: An authentication database, usually a RADIUS server.

The authentication is initiated when the supplicant tries to connect to the network.

The exchange is as follows (Figure 12):

1. The authenticator (Access Point) sends an EAP-Request to authenticate the client(Supplicant).

2. The client sends an EAP-Response packet, with its identity, in reply to a valid EAP-Request.
3. The authenticator encapsulates it in a RADIUS Access-Request message, and sends it to the AAA server (Authentication server)
4. The AAA server sends a RADIUS-Access Challenge to the Authenticator, who in turn sends an EAP-Request to the client.
5. The client responds to this packet (with the password, for example).
6. The authenticator may send an additional Request packets, and the client needs to keep replying with Responses. The sequence of Requests and Responses continues as long as needed.
7. The exchange continues until the client cannot be authenticated by the authenticator (unacceptable response), or until the authenticator decides that authentication has occurred successfully. An EAP-Failure is sent in the event of failure, while an EAP-Success is sent in the case of success.

After the authentication, the Authenticator opens up a port for the Supplicant to communicate with the inside network.

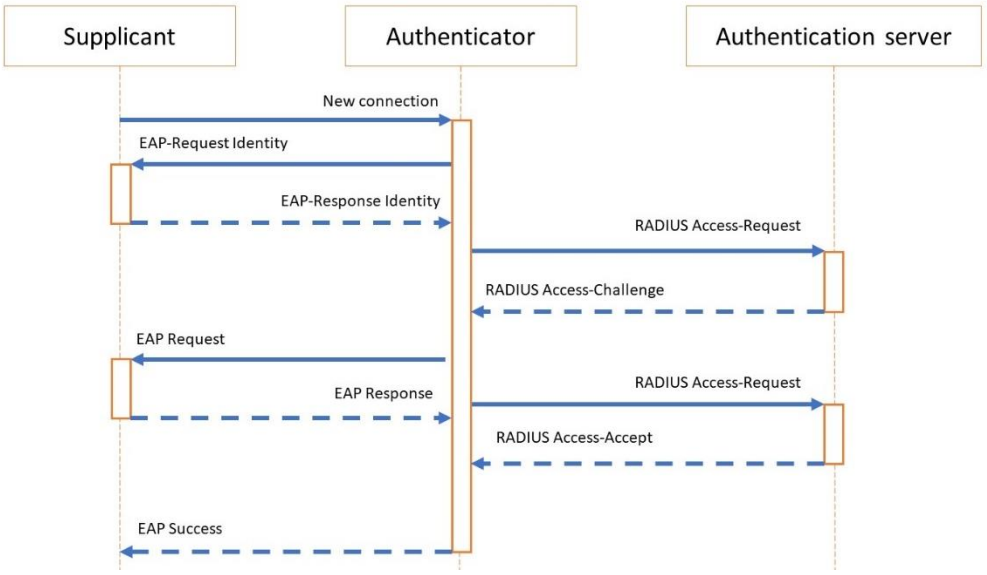


Figure 12 - EAP Message Exchange

3.3 Fault Tolerance

Fault tolerance is a property that defines how a system can continue working correctly if a failure happens. If a system can continue operating correctly despite the failure, we can say that it is fault tolerant.

There are some ways to achieve fault tolerance, the main way being redundancy. Redundancy means that we need to have more capabilities than those that would be necessary in an environment where failures don't occur. In the case of a fault those extra capabilities would be used instead of the normal resources.

In the case of Hub-and-Spoke VPNs, there needs to be an extra Hub up and ready to receive connections even when the first Hub is working. But there are three ways to implement fault tolerance this way:

- The Spoke can be configured to check the reachability of the first Hub, and if it finds out that it can't reach it, it will try an alternative Hub (Dual-Hub Single-Cloud).
- The Spoke can have two active tunnels at the same time, one with each Hub (Dual-Hub Dual-Cloud).
- The HSRP (Hot Standby Router Protocol) can be used to form a cluster of Hubs, that provides redundancy for a local subnet.

In the first method, Dual-Hub Single-Cloud topology, IP SLA/track failure detection is used on the Spoke, to keep track of the Hub. IP SLA[10] is a Cisco proprietary protocol that provides an active method of monitoring network performance. It will generate and actively monitor traffic continuously between the Hub and Spoke. If connectivity to the destination network is lost, the route state is set to Down, and if available, a different static route (which is in state Up) can be selected for routing traffic. When the Spoke detects that the Hub is down through IP SLA tracking, it will try the next configured address. The Spoke can also be configured to reconnect with the main Hub if it detects that it has returned to normal.

The second method, Dual-Hub Dual-Cloud, is to just have two active tunnel interfaces on the client, one connected to each Hub. If one of them is down, the other connection will be used seamlessly.

For the third method, HSRP allows to have a cluster of routers that are layer 2 adjacent, one is the Active Router, while the rest are Standby Routers. All the routers in a HSRP cluster have a shared Virtual IP address (VIP), and the Spokes and Remote Clients connect to this address. This means that HSRP makes a group of routers appear as a single virtual router. The Active Router is responsible for forwarding all the traffic. The other routers are not actively passing traffic but maintain HSRP state with the Active Router. If it fails, a standby router automatically takes over the active role and begins sending and receiving traffic, using the VIP.

All routers in the HSRP group must have an IP address that is separate from the VIP.

Figure 13 shows a topology where Fault tolerance could be employed. It contains two Hubs, 1 and 2, if the main Hub fails, the second one can be used.

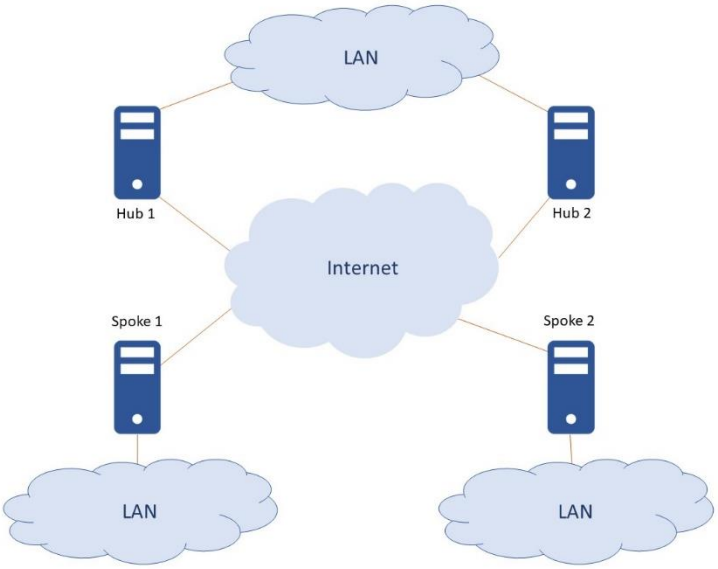


Figure 13 - Redundant topology

3.4 Load Balancing

Load Balancing is necessary if a service is to be scalable. If there are too many users trying to access a server, they should be divided among all available servers. This prevents a single device from being overloaded, and not being able to provide the service correctly.

In FlexVPN, multiple routers can be joined to create a cluster. These routers can use load balancing and distribute incoming IKEv2 requests evenly among them. The main router redirects all client requests to the least loaded gateway based on the system and cryptographic load. This allows for scalability because we can add new gateways to the cluster, or remove them, based on what is needed.

To form the cluster, the FlexVPN peers use the HSRP (Hot Standby Routing Protocol) mentioned in chapter 3.3 In addition to using HSRP, the gateways also use Cisco Load Balancing (CBL). Using

this feature, the HSRP Active Router becomes the CBL Master, while the other routers become the CBL Slaves.

These routers use the CLB Load Management Mechanism, which is a TCP-based protocol that runs between the CLB master and the CLB slaves, to exchange information. Cluster members send details of their load to the Master, namely, memory percentage used, average CPU usage for the past 5 seconds, number of IPSec SAs and number of IKEv2 SAs. This way the active router always knows which gateway is the least loaded. When receiving a connection, the Active Router will always redirect the request to the least loaded gateway. This means that CBL is used to determine the least loaded gateway by sending CPU information to the Master, and HSRP is used to get only one virtual IP address and to redirect the client requests.

Figure 14 shows a load balancing topology. There are three Hubs, using only one VIP (10.1.1.100), and the clients are distributed between these hubs.

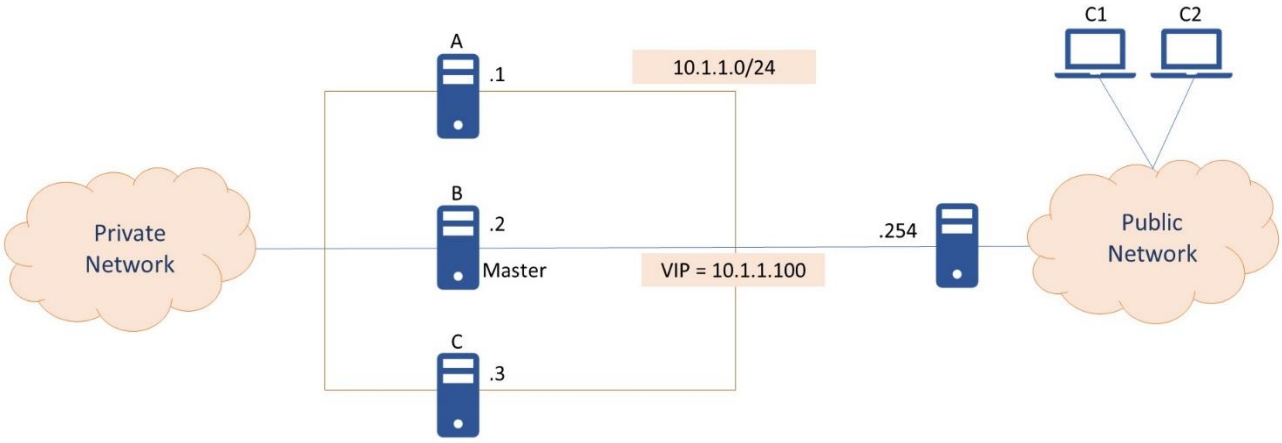


Figure 14 - Load Balancing topology

3.5 Monitoring

Monitoring is a necessary process after the establishment of a VPN connection. First, to check that the connection was established correctly, and that traffic is flowing the way it's supposed to. Second, to verify that everyone that is using the VPN is correctly authenticated and authorized to use it, and what resources they use. Third, to see the amount of resources that the VPN connections are using, and if there is any way to optimize it.

To perform this monitoring the following protocols can be used:

- Authentication, Authorization and Accounting protocols. The Accounting provides the service for collecting information. This service tracks and maintains a log of every session used to access the VPN.

- Cisco's NetFlow. This software collects and measures data allowing all routers or switches in the network to become a source of telemetry and monitoring.
- SNMP (Simple Network Management Protocol). The SNMP framework is composed by a Manager, which is the system used to control and monitor the activity of network hosts, an Agent, which is the software component within the managed device, and the Management Information Base (MIB), which is the database used to store the information.
- Syslog. The syslog protocol provides a transport mechanism to allow a machine to send event notification messages across IP network to event message collectors (syslog servers).

A structured methodology should be used when monitoring VPNs. This ensures that unintended activities are identified and corrected as soon as possible.

Table 1 exemplifies a possible methodology with some functions that should be monitored.

Table 1 – Monitoring methodology

Step	Function	Protocol	Monitoring Method
1	IP Connectivity	Transport routing protocol	SNMP, Syslog
2	VPN Tunnel Establishment	IKEv2	SNMP, Syslog
3.1	Authentication	PSK, PKI, EAP	AAA, Syslog
3.2	Authorization	AAA, RADIUS	AAA, Syslog
4	Data Encryption	IPsec	SNMP, AAA
5	Overlay Routing	Overlay routing protocol	SNMP, Syslog
6	Data Usage	IPsec	NetFlow

Related to monitoring, the ASA firewall is a security device that combines firewall, and virtual private network (VPN) capabilities.

The ASA can establish site-to-site VPNs with another ASA or router. Basically, the ASA firewall can act as a FlexVPN Hub, or Spoke. Adding its firewall capabilities, it can also inspect the traffic coming from, or to, the VPN tunnel.

3.6 Comparison with older VPN technologies

FlexVPN was preceded by many other VPN solutions, in this chapter the focus will be on comparing FlexVPN with GETVPN (Group Encrypted Transport VPN)[11], and DMVPN (Dynamic Multipoint VPN)[12].

GETVPN (Group Encrypted Transport VPN)

GETVPN was created with the idea of encrypting a large private network. Trough the creation of a trusted group, it manages to eliminate the need of point-to-point tunnels and their associated overlay routing. It has four components:

- GDOI (Group Domain of Interpretation) - Protocol to perform group key management.
- Key Servers – Routers responsible for distributing keys to the group members.
- Group Members – Routers used for encryption/decryption.
- Group Security Associations – Two IPSec SAs for the whole group.

All group members (GMs) share a common security association (SA), also known as a group SA. This enables GMs to decrypt traffic that was encrypted by any other GM. The process is the following:

1. Group Members (GM's) "register" via GDOI with the Key Server (KS)
2. KS authenticates & authorizes the GM's
3. KS returns a set of IPsec SA's for the GM's to use
4. GM's exchange encrypted traffic using the group keys

GETVPN isn't suited for use on the Internet, it should be used in a private network. It doesn't support AAA services or authentication and doesn't support Remote-Access Clients nor Hub-and-Spoke topologies.

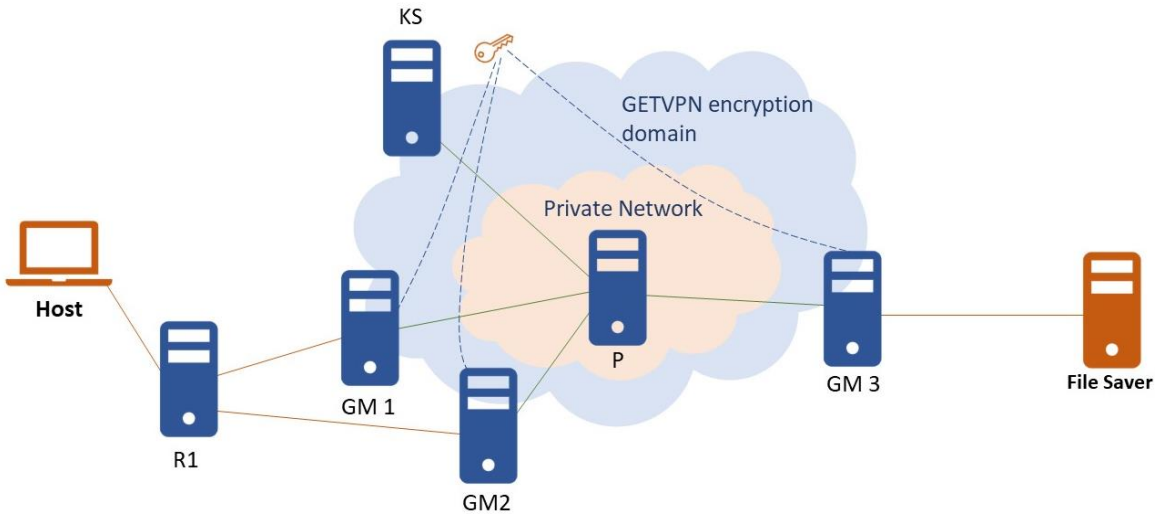


Figure 15 – GETVPN deployment

Figure 15 shows an example of a GETVPN deployment. The key server handles the key distribution among the group members and all traffic going trough the private network is encrypted.

DMVPN (Dynamic Multipoint VPN)

DMVPN, very much like FlexVPN, provided a way to configure Site-to-Site topologies, with Peer-to-Peer encryption. It also used NHRP to provide Spoke-to-Spoke communication. However, it doesn't support AAA services, and as such, doesn't support Remote-Access Clients.

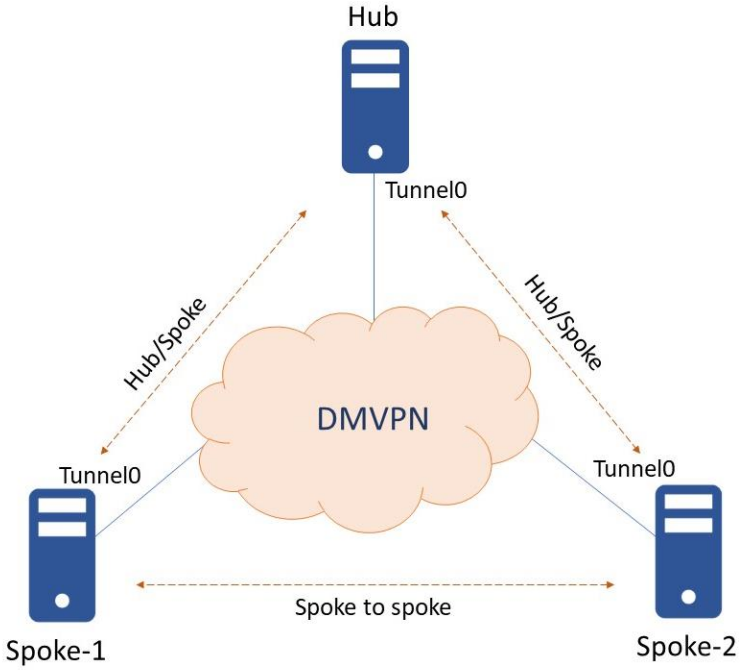


Figure 16 - DMVPN deployment

Figure 16 shows an example of a DMVPN deployment, it's main feature is Spoke-to-Spoke communication, also supported by FlexVPN

FlexVPN

FlexVPN supports many more solutions than its predecessors. Its more suited for use on the public Internet, allows for both Site-to-Site, and Remote-Access topologies, at the same time. It also supports AAA, with digital certificates, pre-shared keys, or EAP. It also supports Load Balancing, Failover redundancy, and is resistant to replay attacks.

Table 2 makes a comparation between DMVPN, GETVPN and FlexVPN.

Table 2 - FlexVPN compared with DMVPN and GETVPN

	FlexVPN	DMVPN	GET VPN

Infrastructure Network	Public Internet Transport	Public Internet Transport	Private IP Transport
Network Style	Site to Site and Remote Access	Hub-Spoke and Spoke-to-Spoke; (Site-to-Site)	Any-to-any; (Site-to-Site)
Encryption Style	Peer-to-Peer Protection	Peer-to-Peer Protection	Group Protection
AAA Suport	Local or External Server	No	No

3.7 Replay attacks against FlexVPN

Replay attacks are the main ways malicious actors target VPNs, as the traffic is normally encrypted you can't inspect it, and you can't alter the packet because there's integrity checks. But if the routers accepted duplicated packets, they could malfunction or cause a Denial of Service.

IPsec provides Anti-Replay Check Failures[13] protection against an attacker duplicating encrypted packets by giving a unique sequence number to each packet. The decryptor keeps track of which packets it has received based on these numbers. Currently, the default window size is 64 packets.

Here are is the process to handle incoming IPSec traffic on the receiving interface with anti-replay:

- When a packet is received, if the sequence number is within the window and was not previously received, the packet is accepted, and marked as received before it is sent to integrity verification.
- If the sequence number is within the window and was previously received, or if the sequence number is less than the lowest sequence in the window the packet is dropped, and the replay counter is incremented.
- If the sequence number is greater than the highest sequence number in the window, the packet is accepted, and marked as received. The sliding window is then moved to the right.

The replay counter is only used to log the amount of dropped packets, and can be consulted to check if there has been a large number of duplicated packets.

Figure 17 shows an example of a sliding window, some sequence numbers that would be accepted, and some that wouldn't.

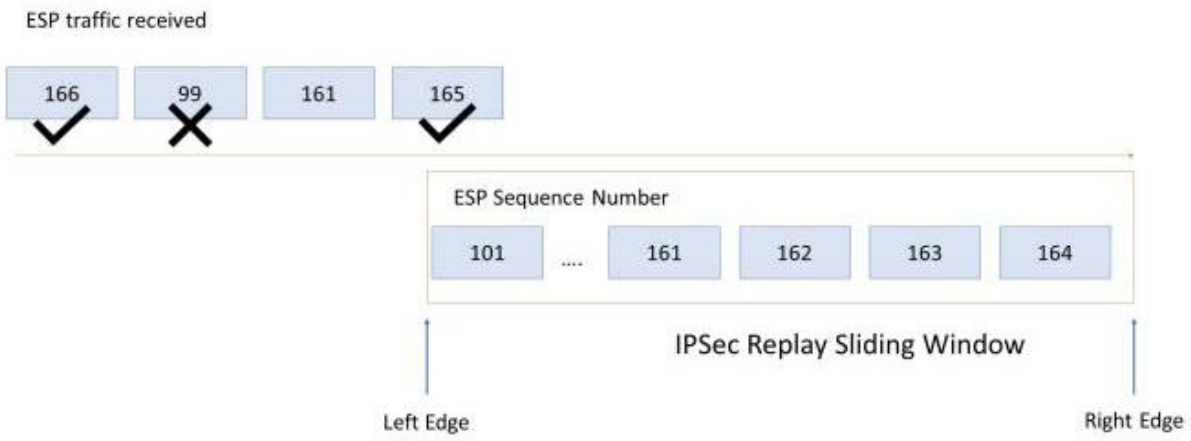


Figure 17 - IPSec Replay Sliding Window

The packet with number 99 wouldn't be accepted because it is out of the Sliding Window.

The one with number 161 would be accepted if it hadn't been received before and dropped if it had been received.

Both 165 and 166 would be accepted because they're to the right edge of the window.

4. Public Key Infrastructure (PKI)

A public key infrastructure (PKI) is a system for the employment and distribution of digital certificates, these are used to confirm that a certain public key belongs to the entity using it. The PKI has actors responsible for the creation of digital ceritificates, these map the public keys to the entities, and stores these certificates, revoking them when they are no longer valid.

A public key infrastructure is based on digital signatures, which utilizes public key cryptography. There is a private key for each entity, known only by that entity and used for signing. There is another key, the public key, which is used for validating signatures but cannot be used to sign. This public key is available to anyone and is included in the certificate.

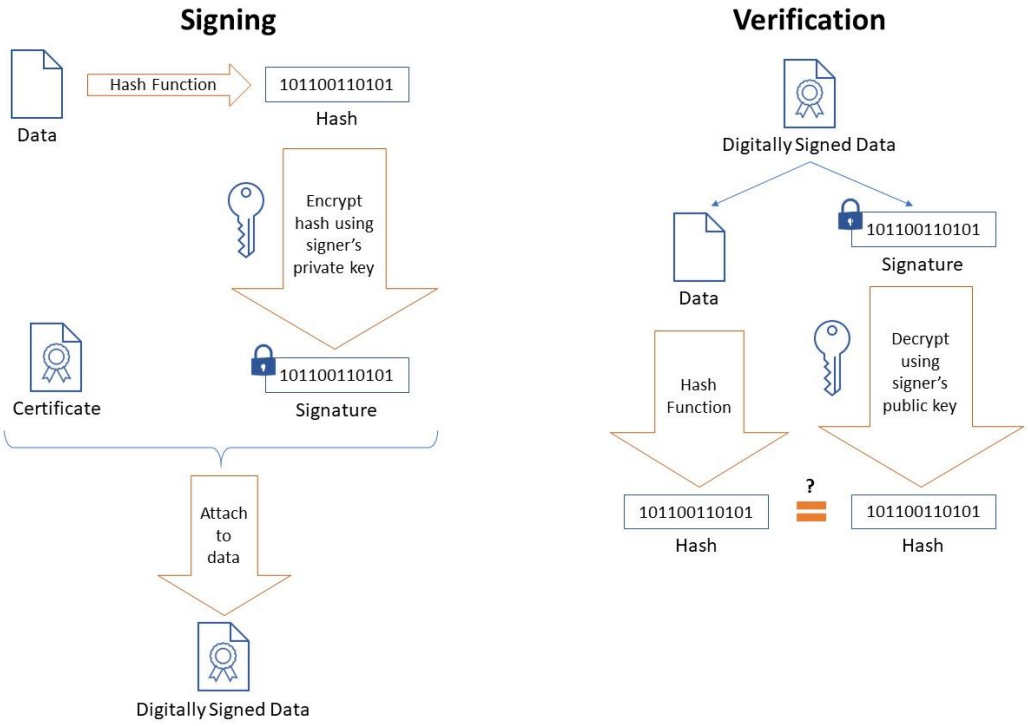


Figure 18 - Digital signature and its validation

Figure 18 represents the use of a digital signature and later its validation. First, data is hashed, and that hash is encrypted with the sender's private key. This is added to the data and sent together with the senders Certificate. When the receiver gets the Digitally Signed Data, he decrypts the signature using the Sender's public key and compares it to the hash of the data. If they're equal the message is authentic and hasn't been tampered with. If it had, the data hash would be different, and the signature validation would've failed.

A PKI consists of:

- A certificate authority (CA) that stores, issues, and signs the digital certificates.
- A registration authority (RA) which verifies the identity of entities requesting their digital certificates to be stored at the CA.
- A secure server in which keys are stored.

With a PKI we can authenticate users and devices. There needs to be trusted parties to assure that a certain key belongs to its entity, to do this, they sign that entity's certificate. The public key can then be used as an identity for the user in certain networks using PKI.

The trusted party signing the certificate that associates the key with the user is a CA. The CA also has a cryptographic key that it uses for signing these certificates.

4.1 Certificates

A certificate is a digital document, it proves the ownership of a public key. It has information about the key, who owns it, the signature of the CA that issued that certificate, and more. If the signature is valid and the entity validating the certificate trusts the CA, then the certificate is validated.

The most common format for public key certificates is defined by X.509[14].In cryptography, X.509 is a standard defining the format of public key certificates. The structure of an X.509 v3 digital certificate is as follows:

- Version Number
- Serial Number
- Signature Algorithm ID
- Issuer Name
- Validity period
- Subject name
- Subject Public Key Info
- Certificate Signature Algorithm
- Certificate Signature

4.2 Certificate Authority (CA) and Registration Authority (RA)

Certificate Authorities (CAs) are the entities that have been entrusted with the responsibility of issuing digital certificates. CAs are at the center of the entire PKI model. For the PKI to correctly function, a CA needs to validate the entities that the certificates are being given to. If this doesn't happen, PKI is impossible.

The main objective of an CA is to digitally sign the public key of a given user. This is done using the CA's own private key, that means that the CA is assuring that the key belongs to the entity, and everyone that trusts the CA will trust that certificate.

The PKI role that may be delegated by a CA to assure valid and correct registration is called a registration authority (RA). A RA accepts requests for certificates and authenticates the requesters. It approves or rejects certificate applications and is also capable of initiation certificate revocations, and can approve, or reject, renovation requests. However, RA cannot sign or issue certificates.

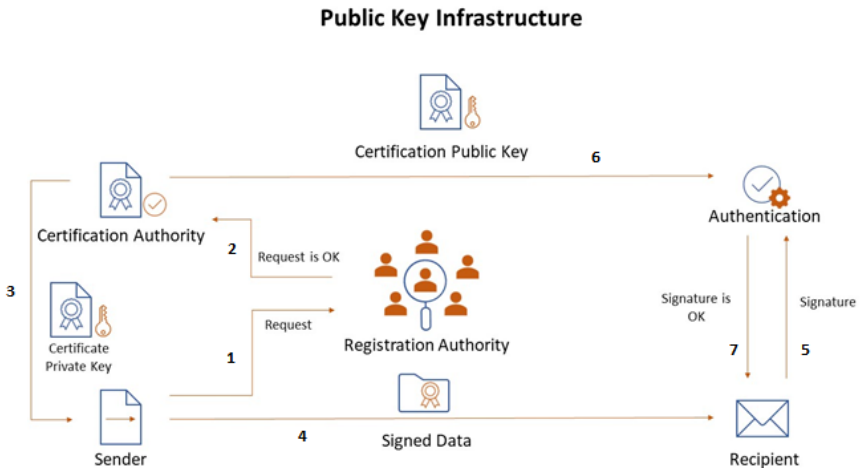


Figure 19 - Public Key Infrastructure

Figure 19 shows how a public key infrastructure works. The sender gets registered with an RA and a CA signs its certificate. He can then send signed data that a recipient can validate.

4.3 Hierarchical PKIs and Chain of Trust

Certificate hierarchy is one of the most important parts of PKI design because it affects how certificates are validated and used by PKI solutions. It will also affect their scalability. A PKI can be divided by levels of CAs, normally it's either a single CA, or a multi-layer CA infrastructure.

When there is a multi-layer CA PKI then there is a Chain of Trust. The root CA must be trusted for the PKI to work, and when the root CA issues the lower level CAs certificates, then those CAs become trustworthy because the root CA trusts them. This chain continues until the last level of the PKI. When a user trusts the root CA, then it will trust all other CAs whose certificate has been signed by the root, or by another CA with a certificate issued by the root CA. This is also called certificate chaining.

Single-Tier Hierarchy

Consists of a single CA. This CA is both the root (highest level) CA, and an issuing CA. A root CA owns the certificate that acts as the base of trust for the PKI. So a root CA public key serves as the beginning of trust paths for a domain. Any users, or devices that trust the root CA also trust any certificates issued by the PKI hierarchy. The root CA is the base for the chain of trust and it must be trusted manually. There is no higher level CA above to issue the root CA's certificate, so its own certificate is self-signed.

There are multiple risks in using only one CA: there is a single point of failure and you cannot revoke its certificate if there is a problem. Normally, root and issuing CAs are separated because it is

difficult to distribute a new root CA certificate to replace a compromised one. Because of this, a one-tier hierarchy may be sufficient for only simple implementations where there is little need for greater levels of security or flexibility.

Multiple-Tier Hierarchy

Consists of multiple CAs, arranged by levels. There is a root CA tier, issuing CAs, and (optional) middle tier/tiers placed between them. The second tier CA can be used as a policy CA. The policy CA is configured to issue certificates to the next level CA that will, in turn, provide certificates for users. Another reason to have the second tier added is that if you need to revoke a few CAs due to a compromise, you can do it at the second level, leaving other branches functional.

Security increases with the addition of a tier. This happens because compromised CAs will affect a smaller number of users, and the compromised CA can also be easily replaced as long as it isn't the root CA. On the other hand, manageability is more difficult as there are more CAs in the hierarchy to manage. The performance of certificate chain authentication on PKI clients is affected with an increase in the number of tiers because the clients need to verify the certificates of both issuing CAs, and policy CAs. Policy CAs don't issue certificates to normal users, only to lower level CAs. Figure 20 shows an example of a multi-tier CA with 3 levels. Basically, using a multiple-tier hierarchy is useful when there are a lot of users, if there's a problem with a CA then only its users will be affected instead of all users. This hierarchy is also very useful if the PKI expands internationally. Having a single CA for users all over the world is less efficient than having the user certificates issued by Subordinate CAs (SubCA) in their respective areas. Subordinate CA is any CA that isn't root. These lower-level CAs get their certificate from the root, or policy, CA and issue to normal users

The root and intermediate CAs, usually, are only online when issuing the certificates for lower level (issuing) CAs, after this is done they are put offline so they can't be compromised.

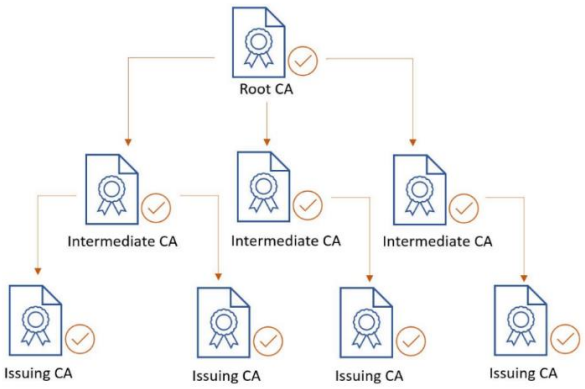


Figure 20 - Multi-tier CA Example

4.4 Enrollment

Enrollment is the process required to obtain a digital certificate. This usually follows several steps from before the certificate request until the user has a fully signed, and functional certificate. The process begins with a user generating an RSA Key pair. The user must then send a certificate request, with the public RSA key, to an issuing CA. The CA will then sign the request with its private key and generate the user certificate, which will then be sent to the user.

There are two ways to perform the certificate enrollment: manually or using SCEP-based enrollment. SCEP[15] (Simple Certificate Enrollment Protocol) is a network based approach that doesn't require the manual exportation nor importation of certificate files.

Manual enrollment requires the CA certificate to be manually exported by the CA, and imported by the user. The user must also export their certificate request and the CA must import it manually. The request must be approved before the final user certificate is manually exported and sent to the user.

SCEP-based enrollment is much easier to accomplish, and when there are a lot of network elements in need of a certificate, manual enrollment might take a long time, and a lot of configuration. SCEP enrollment requires a network connection between the user and the certificate server (CA) but the protocol will handle all steps in the enrollment, except the key generation.

SCEP runs on top of TCP and uses the HTTP POST and GET methods to exchange information with the CA. To get the CA certificate, the user sends a GetCACert message to the CA. A PKIOperation message is sent from the user to the CA to perform a certificate enrolment or renewal request. If the request is granted, a CertRep SUCCESS message is sent from the CA back to the user, and if the request is rejected, a CertRep FAILURE message is returned. If the CA is configured to manually authenticate the client, a CertRep with pkiStatus set to PENDING is sent to the user, who will then enter the polling state by periodically sending CertPoll messages to the CA until either the request is granted and the certificate is sent back or the request is rejected. These exchanges are shown in Figure 21.

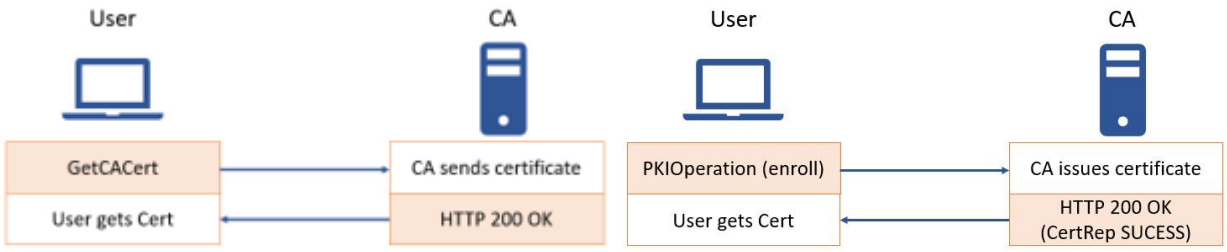


Figure 21 - SCEP Message Exchange

4.5 Certificate Revocation

All certificates have an end date when they expire. They also may be deemed invalid/compromised and need to be revoked before being replaced. If a certificate is compromised or

invalid, authenticating it might be dangerous because it the original user's private key might have been leaked. To deal with this situation its necessary to have some means to detect revoked certificates. There are three ways to verify if a certificate has been revoked prior to expiring: using Certificate Revocation Lists, where a list of revoked certificates is downloaded periodically, using OSCP[16] (Online Certificate Status Protocol) that uses network calls to a server for each certificate that it authenticates, and using AAA to restrict access to a specific certificate (the certificate attributes must be used as an AAA user).

Certificate Revocation Lists

CRLs are download periodically from a server, or a CA, and are comprised of the certificate serial number and the revocation date. The revocation list also has a lifetime after which it will expire and a new one must be downloaded. This means that CRLs don't work in real time and a revoked certificate can still be authenticated if the new CRL hasn't been downloaded yet. 0

OSCP

Using OSCP solves a problem with Certificate Revocation Lists, which is the real time checking of revoked certificates. When a device needs to check the validity of a certificate, it will query the OSCP server with the certificate serial number and the OSCP will provide the status for the certificate. However, the OSCP server still needs to check which certificates are revoked, and to do this it requires a CRL download from the CA, even if the CRL download is done in smaller time intervals. This means that there's still a small chance that a revoked certificate is authenticated by another user.

AAA

Using AAA to restrict network access to a user isn't revoking the certificate, but can also be used to a similar effect. When it is deemed that a certificate is no longer authorized to access the network, the change only needs to be done in the AAA server and network access will be restricted. However, this requires manually configuring certain certificate fields into the AAA server, for each certificate used for authentication.

5. Routing Security

The Internet is divided into many networks known as autonomous systems (AS)[17]. Each of these networks is essentially a large network run by a single organization. Routing is done, inside the ASs, and between them. Each AS also has a registered autonomous system number (ASN), which identifies it. Routing protocols set the movement of packets between host and any other host it is communicating with. All communications, be it inside ASs, or between them, are handled by these protocols.

All routing protocols are divided into two groups, IGPs (Interior Gateway Protocols) and EGPs (Exterior Gateway Protocols). IGPs are configured to handle routing inside each AS, while EGPs

handle routing between ASs. Figure 22 shows a connection between two ASs, and the routing protocols used inside them (RIP, OSPF)[18], [19], and between them (BGP)[20].

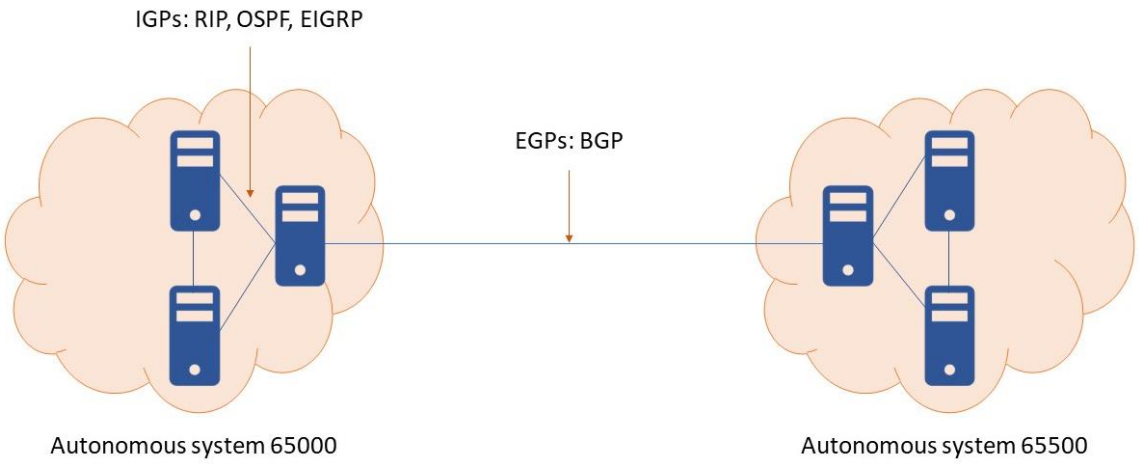


Figure 22 – Example architecture between two ASs

Most routing protocols are vulnerable to attacks, and these issues may cause problems for entire networks or redirect traffic to unintended entities. Sniffing-based attacks, man-in-the-middle, replay attacks and more are used to alter or corrupt routing tables and target the network infrastructure.

The routing protocols are also the basic infrastructure of VPNs. Without these protocols, we cannot use a VPN, which means that any attack on the routing protocol also affects the VPN. Because of this, it's very important to know the vulnerabilities of the most used protocols, like RIP, OSPF and BGP. It's even more important to know countermeasures against those vulnerabilities so that we can protect important systems.

5.1 RIPv2

RIPv2 (Routing Information Protocol Version 2) is an older protocol when compared with OSPF and BGP, but it is still used in many AS networks. It measures distance cost with hop-count, which means that the more hops, the bigger the cost and it only has two types of message: request and response. The response message is the most important, seeing as the request message is only sent right after boot. The request message asks neighboring routers to send their routing tables, while a response message sends the own routers routing table. Response messages can be sent without a request message being received.

At boot, and every 30 seconds from then on, a router configured with RIPv2 broadcasts to 224.0.0.9 a response message through every RIPv2 interface. Neighboring routers update their own routing table, with the new routes. As the requesting router receives updates from different

neighboring routers it will only update the reachable networks in its routing table if it receives information about a reachable network that isn't in its routing table, or information that a network it has in its routing table is reachable with a lower hop count.

An attacker could send fake RIPv2 response messages to insert fake routes into the routers and perform a man-in-the-middle attack. All that would be needed to do is insert a route with the victim's IP address to himself.

To prevent this kind of attacks, RIPv2 allows for password authentication, either with a plain text password or an MD5 hash, which means the routers will only accept messages from authenticated peers.

5.2 OSPFv2

OSPFv2 (Open Shortest Path First Version 2), much like RIP, handles communication inside ASs, but every node constructs a map of the connectivity to the network, in the form of a graph, showing which nodes are connected to which other nodes.

A router interface with OSPF will advertise its link cost through multicast. All routers using OSPF keep sending hello packets, a mechanism to detect if links are operational, and changes in the cost of their links become known to neighboring routers. OSPF routers advertise the information they receive from one neighboring router to all other neighboring routers. Based on this information, all routers with OSPF continuously update their databases with information about the network topology and adjust their routing tables. The database is called LSDB (link state database), which is filled with LSAs (link state advertisement). These LSAs have different types and refer to different parts of a network topology (For example, LSA Type 1 refers to the routers in the topology, while Type 2 refers the networks).

A possible countermeasure against OSPF attacks is the enabling of password authentication in OSPF. As mentioned before for RIPv2, OSPF also allows for the use of a password, either in plain-text, or an MD5 hash.

5.3 BGP

BGP (Border Gateway Protocol) is the most used protocol for communications between ASs. BGP neighbors are called peers, and every router running External BGP must have the ASN configured, which means that it must belong to an AS. Each BGP speaker exchanges routing information with its neighboring peers in the form of network prefixes, it announces these prefix announcements to its other peers. In this way, BGP allows an AS to collect all the routing information from its neighboring autonomous systems and "advertise" that information further. This means that if one BGP peer advertises networks he doesn't control, that advertisement will spread trough the Internet. This is called BGP Hijacking.

BGP hijacking is when Internet traffic is rerouted to someplace it's not supposed to go. This is accomplished by announcing network prefixes that they do not own. When an AS announces a route to IP, this announcement will be added to routing tables in BGP routers across the Internet. From then on, traffic to those IPs will be routed to that AS.

BGP uses the shortest, most specific path to the destination IP address. For the BGP hijack to be successful, the route announcement must either: offer a more specific route by announcing a smaller range of IP addresses than other ASs had previously announced (having a larger prefix), or offer a shorter route to the network prefix.

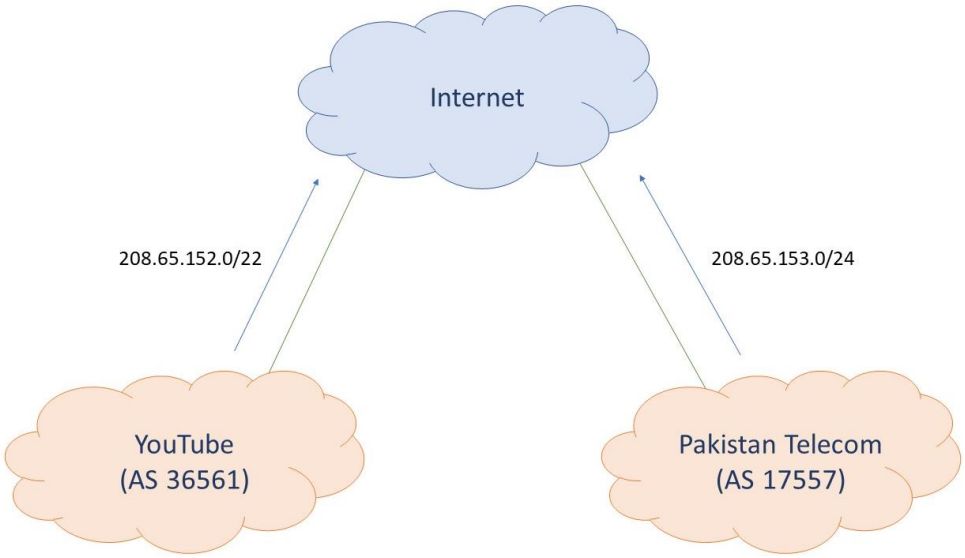


Figure 23 - Real case of BGP Hijack

Figure 23 shows a real case where a BGP Hijack happened. Pakistan Telecom wanted to block Youtube on their AS but ended up advertising an IP range controlled by Youtube, and with a larger network prefix. All of Youtube's traffic during the Hijack was directed to the AS owned by Pakistan Telecom.

These attacks can happen because BGP relies on trust, and autonomous systems trust the routes that are shared with them. There are many solutions proposed to correct these problems, but they're hard to implement because they would require every autonomous system to simultaneously update their behavior.

One of these countermeasures is RPKI (Resource Public Key Infrastructure)[21]. It allows the association of IP ranges with an ASN. Each IP range has a ROA (Route Origin Authorization), that defines which ASs are allowed to advertise it. These ROAs are stored in an external server with an RPKI Validator. Before a BGP peer accepts an advertisement, it will first check the RPKI server to validate that the advertiser is controls that IP range, and is authorized to advertise it. Only after confirming it does the router accept the advertisement it received.

6. SDWAN

The technology approached in this part of the dissertation is SDWAN[22]. It allows for the deployment of VPNs and its integration with other network functions. Using this solution, multiple sites can be connected seamlessly without the need for manual configuration, and advanced network functions can also be implemented without the use of extra equipment. This solution isn't a VPN in itself, but it can be used to create VPNs on-demand using Software-Defined Networking, in which the control plane of a network is centralized in a central server (controller).

6.1 Software-Defined Networking (SDN)

Software-Defined Networking (SDN) uses a different model from conventional networking, in which a decentralized structure is employed with dedicated network devices to route, control, and inspect traffic. SDN is an approach that uses a centralized structure, with one or more software controllers that dictate what happens to the packets on the network.

The network elements will receive their configuration from the controller, which means that virtual networks and network functions can be deployed without needing to configure several devices one by one. And seeing as everything is software-based, there is no need for specialized hardware to run these network functions.

Figure 24 shows a possible SDN topology where every device is connected to the controller, and will receive the configurations to achieve connectivity.

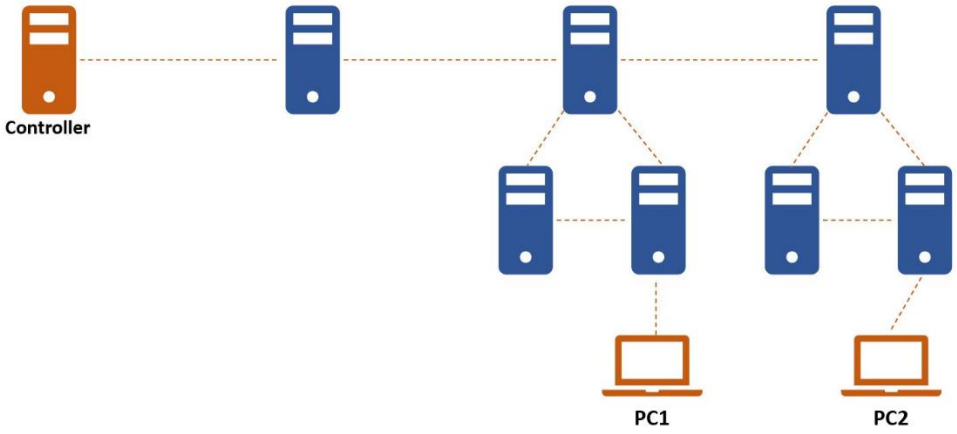


Figure 24 - Simple SDN Topology

6.2 SDWAN advantages

Software-Defined WAN is a way to configure a virtual WAN over a physical WAN or the Internet. These networks improve upon physical networks by enabling network automation and virtualization. They are more manageable, flexible, secure, and scalable than traditional networks.

Manual network configuration changes are slow and can lead to disruptions on the network. There is also an increasing number of users in most networks which makes configuring user credentials and maintaining a consistent policy very complex.

SDWAN brings the advantages of SDN to conventional WANs, with a centralized structure and a controller-based topology, the only requirement is to have a Software-Defined Gateway on site. Afterwards, the gateway can also be configured with several network functions (ACLs, Encryption, etc.)

Figure 25 illustrates how a possible SD-WAN Architecture would be deployed, with the controller being in the Main Site, and Software Defined Gateways in each site which establish connections

between each other and the Internet. A site may have normal connectivity to the Internet, or it can be configured to send all traffic to another site before being redirected outside the SDWAN.

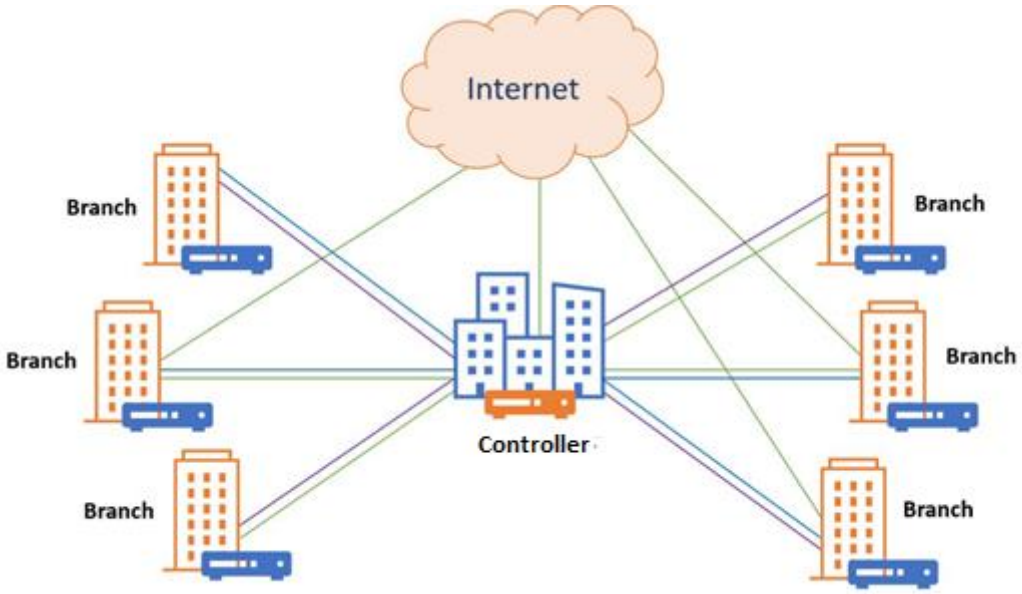


Figure 25 - SDWAN Architecture

6.3 Nuage Networks VNS

The Nuage Networks Virtual Network Services[23] (VNS) is a Nokia proprietary solution. It is based on three core elements: the Network Services Gateway (NSG), the Virtualized Services Directory (VSD) and the Virtualized Services Controller (VSC). Besides these three elements, it also needs a Key Server and a PKI to properly deploy this solution. These elements can be seen in Figure 26.

6.3.1 Architecture

Network Services Gateway (NSG)

The Network Services Gateway (NSG) constitutes the network forwarding plane for the network service. By installing an NSG at every site, this solution provides the capability to create overlay VPNs between any sites. Once the NSGs are activated, they automatically connect and are authorized by the VSD, download their configuration, and establish connectivity based on the pre-defined policies.

Virtualized Services Directory (VSD)

The VSD provides a network policy framework that enables the definition and enforcement of network policies, and the logging of activity of the SDWAN. The VSD also manages network resource assignments such as ACLs, which will be enforced by the NSGs, and IP address assignments. It is the dashboard through which the whole network can be configured.

Virtualized Services Controller (VSC)

The VSC acts as the control plane. It is a virtual machine application running all the control plane tasks. It provides a management interface to the network administrator and has an CLI as well as a toolset such as SNMP management, syslog and TACACS+/RADIUS AAA. It is also responsible for running routing protocols like BGP that allow it to have connectivity with other controllers as well as receive topology information of the underlay through Interior Routing Protocols (IGPs)

The VSC is implemented as a virtual machine. This means it can be hosted almost anywhere, only requiring IP connectivity to the other VNS components.

Virtual Routing and Switching (VRS)

The Nuage Networks also uses a component, Virtual Routing and Switching (VRS), which is an Open vSwitch (Virtual Switch) implementation that acts as the network forwarding plane. It encapsulates and de-encapsulates user traffic, enforcing policies defined in the VSD. The VRS tracks VM creation and deletion events to adjust network connectivity. This element is purely software and is present on all NSGs.

Public Key Infrastructure (PKI)

A Public Key with a certificate authority (CA) that both issues and verifies the digital certificates. The process will use these capabilities to handle certificate based NSG authentication/authorization.

Key Server (KS)

Key Server is a component for key management. This server will oversee the keys distribution for IPsec capabilities. The gateways will share a key for the IPsec tunnel and the KS will handle the re-key (Key redistribution), once the former key expires.

Figure 26 shows how the Nuage elements interact with each other. The NSG acts as the gateway for the private LAN, it gets its routing configuration from the controller, through OpenFlow, and network assignments from the VSD through HTTPS. The VSC communicates with the VSD using Extensible Messaging and Presence Protocol (XMPP). This means that the controller will install routes on the NSG, but the policy information, for example Access Control Lists or VPN information (if the NSG is part of a VPN), will be obtained from the VSD.

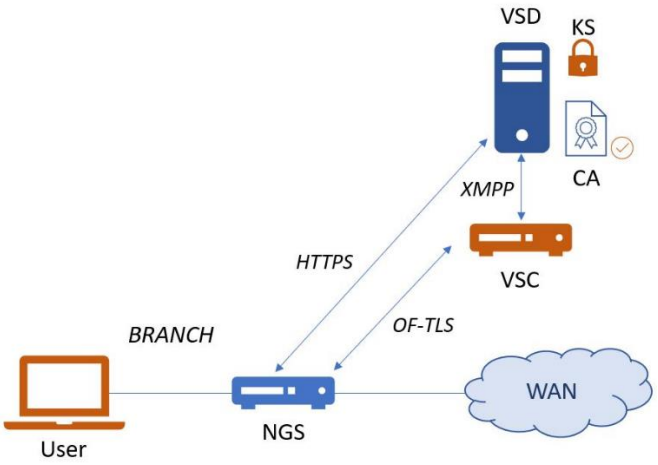


Figure 26 - Nuage Networks Architecture

6.3.2 Main Protocols

OF-TLS

OpenFlow (OF) is a network protocol used in SDN to direct and manage traffic. It allows the controllers to configure other network elements.

An OF session is established from the NSG to a VSC. The session is encrypted using Transport Layer Security (TLS), and this requires that both the NSG and VSC have signed certificates for authentication. The protocol is used for communication between the controller (VSC) and the NSG.

BGP

The controller (VSC) interacts with the underlay network, other VSCs, and NSGs, by using the BGP routing protocol. BGP is used to distribute the information for each of the services and allows multiple VSCs to form a federation, allowing services to span outside of a single controller. IGPs are also supported between the Controller and the underlay, both OSPF and IS-IS, where the main purpose is to allow the VSC to receive routing and topology information of the underlay and establish the BGP session with other VSCs and NSGs.

MP-BGP (Multiprotocol BGP)

MP-BGP is an extension of BGP that enables it to carry routing information for multiple network layers and address families. TLS-encrypted MP-BGP sessions can be used by an NSG to communicate with the VSC. The NSG uses the MP-BGP session to advertise routes learned from the downstream network to the VSC. The VSC then advertises these routes to remote NSGs. The NSG programs the path locally after learning BGP routes, and the VSC programs the path for remote NSGs.

Virtual Extensible LAN protocol (VXLAN)

The objective of VXLAN[24] is to encapsulate a Layer 2 network and to stretch it over a Layer 3 network, this way VLANs can be extended over the internet. This means that using VXLAN on two remote, internet connected sites, it would be possible for two devices to communicate with each other at Layer 2.

Each VXLAN is identified by a 24-bit VNI tag (VXLAN Network Identifier). 24 bits enable more than 16 million different VXLANs (as opposed to 4096 VLANs in conventional VLANs). A VXLAN may start or end on a VTEP (VXLAN Tunnel End Point), this VTEP encapsulates a full Layer 2 frame in/out of a UDP Layer 3 packet, it also maps a local VLAN to a VXLAN ID and remote IP VTEP. A VTEP has two logical interfaces: an uplink and a downlink. The uplink is responsible for receiving VXLAN frames and acts as a tunnel endpoint with an IP address used for routing VXLAN encapsulated frames.

Figure 27 below shows a depiction of a VTEP table, which maps the MAC addresses, and VXLAN IDs to Remote VTEP IP addresses. When a packet is sent to a L2 MAC address, it will be encapsulated in a L3 packet, and sent to the corresponding Remote VTEP address with the correct VXLAN ID, this ID is stored in a database in the VTEP. If a L2 packet was to be sent to MAC address 11:11:11:11:11:11 it would reach the VTEP, be encapsulated in an UDP packet with VXLAN ID 1001 and sent to 10.1.1.1 . The destination VTEP would receive the UDP packet, check the VXLAN ID and then would send the L2 decapsulated packet to the correct VLAN.

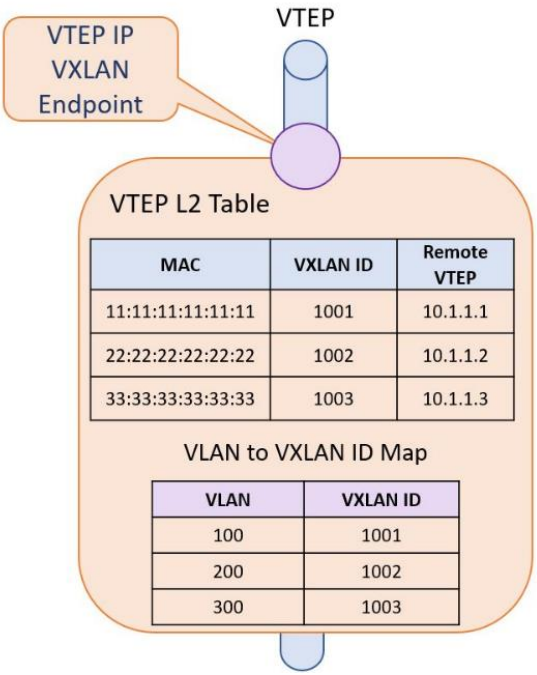


Figure 27 - VXLAN VTEP table example

A VTEP enables the mapping of two remote/routed segments on a single LAN. When a packet is sent to a remote VLAN, VXLAN frames are sent to the IP address assigned to the destination VTEP. Packets received on the uplink are mapped from the VXLAN ID to a VLAN and the Ethernet frame is sent as an 802.1Q Ethernet frame on the downlink. Packets received on the downlink are mapped to a VXLAN ID using the VLAN of the frame. A lookup is then performed within the VTEP L2 table using the VXLAN ID and destination MAC, this lookup provides the IP address of the destination VTEP.

Each VXLAN ID has an assigned IP multicast group to use for traffic flooding. When a frame is received on the downlink for an unknown destination it is encapsulated using the IP of the assigned multicast group, then sent out the uplink. Any VTEP with nodes on that VXLAN ID will have joined the multicast group and receive the frame.

6.4 Cisco SD-WAN Secure Extensible Network (SEN)

The Cisco SDWAN SEN[25] is a proprietary solution, in this solution the network runs as an overlay on hardware (vEdge routers), or virtual machines (vEdge Cloud routers), which act as the gateways between the LAN and the SDWAN.

Centralized controllers (vSmart) oversee the control plane of the SDWAN, maintenance, and security for the overlay network.

Another component the vBond Orchestrator, authenticates all edge devices when they join the overlay network.

The vManage system provides a way to configure the entire network from a single device.

Each vEdge router and vSmart Controller is assigned a system IP address, which identifies the system independently of any interface addresses. This address is similar to the router ID on a regular router.

The system IP address provides permanent network overlay addresses for vEdge routers and Cisco vSmart Controllers and allows the physical interfaces to be renumbered as needed without affecting the reachability of the Cisco vEdge device.

6.4.1 Architecture

vManage (management)

Cisco vManage is a network management system that lets you configure and manage the entire overlay network from a dashboard. It allows the configuration and management of vEdge network devices.

vManage can also be used to store certificate credentials, and to create configurations for all vEdge components. As these components come online, they request certificates and configurations from vManage. When these requests are received, the vManage pushes the certificates and configurations to the requesting network devices.

For vEdge Cloud routers, Cisco vManage can also sign certificates and generate bootstrap configurations.

vSmart (controller)

The vSmart Controller is the centralized controller of the SDWAN solution, controlling the data traffic throughout the network. It works with the vBond Orchestrator to authenticate vEdge devices as they join the network and to arrange connectivity among the vEdge routers. It functions as the control plane of the architecture, advertises routes, and handles security and policy information.

The vSmart Controller also has a route table that stores the routes learned from the edge devices, which announce their own private LANs, and other controllers, called OMP routes. Based on the configured policy, the vSmart Controller shares this information with the vEdge network devices in the network so that they can communicate with each other.

Each vSmart Controller establishes and maintains a control plane connection with each vEdge router in the overlay network.

vEdge (routers)

The vEdge routers sit at the edge of a site (such as remote sites/branches) and provide connectivity among the sites. They are either hardware devices or software. vEdge routers handle the transmission of data traffic. Basic authentication of a vEdge router is done using certificates and RSA cryptography.

Each site is identified by a unique integer, called a site ID. Each vEdge device at a site is identified by the same site ID. A branch or local site typically has a single vEdge router, but if a second one is present for redundancy, both routers are configured with the same site ID.

vBond (orchestrator)

The Cisco vBond Orchestrator coordinates the initial bringup of Cisco vSmart Controllers and vEdge routers and provides connectivity between them. During the starting processes, the vBond authenticates and validates the devices meant to join the overlay network.

The vBond Orchestrator is the only device that is in a public address space. This design allows the vBond to communicate with vSmart Controllers and vEdge routers that are located behind NAT. The vSmart and vEdge can initiate a connection with the vBond seeing as its address is in the public space.

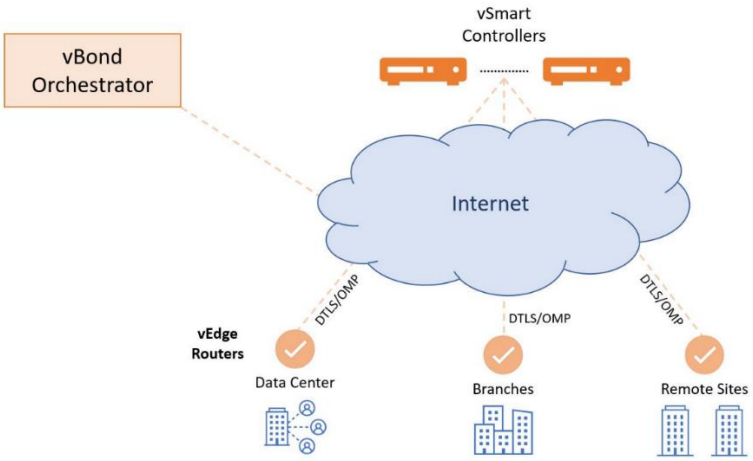


Figure 28 - Cisco SEN Architecture

Figure 28 illustrates the components of the Cisco SD-WAN SEN, The vEdges act as gateways to the remote sites. The vBond is available in the public Internet for when the Edges come online, and the smart controllers and vManage are in a distinct site from which they can also be reached.

6.4.2 Main Protocols

DTLS (Datagram Transport Layer Security)

DTLS is a protocol based on TLS, but instead of using TCP it uses UDP.

Each vEdge router has one permanent DTLS connection to each vSmart Controller it communicates with. This connection is established after device authentication, and it carries encrypted packets between the vEdge router and the Controller. These packets consist of route information necessary for the Controller to determine the network topology, and then to calculate the routes to network destinations and to distribute this route information to the vEdges.

Each vBond orchestrator has a control plane connection in the form of a DTLS tunnel with each vSmart in its domain. In addition, the vBond uses DTLS connections to communicate with vEdge routers when they come online, to authenticate them, and to redirect the router to the controller.

OMP (Overlay Management Protocol)

The OMP protocol is a proprietary routing protocol similar to BGP that manages the SD-WAN overlay network. OMP runs inside DTLS control plane connections and carries the routes, next hops, keys, and policy information needed to establish and maintain the overlay network.

This protocol runs between the vSmart Controller and the vEdge routers and carries only control plane information. The vSmart Controller processes the routes from each vEdge's LAN and advertises information learned from these routes to other vEdge routers in the overlay network.

6.5 FlexiWAN

flexiWAN[26] is composed of a software-based edge device, flexiEdge, and a management system, flexiManage. The flexiManage acts has a controller and configuration point, where the entire network can be configured, and traffic policies can be defined.

6.5.1 Architecture

flexiEdge

A flexiEdge device works as the SDN gateway on the network sites. It is a software implementation that can be installed on any normal Ubuntu machine.

The flexiEdge connects with flexiManage using a token created in flexiManage. Only one token is required for all flexiEdge devices of the organization. The Token needs to be placed in a specific file in all flexiEdge devices, when the Token is placed on the vEdge, the device will show up on the dashboard and will require manual approval to be activated. On registration, the flexiEdge device automatically gets a unique, per-device ID, and use it for its connection and authentication with flexiManage.

It has three main components:

- Router infrastructure - a modified version of FD.io Vector Packet Processor (VPP), which changes the way that packets are processed by processing several packets at the same time instead of only one.
- Routing control plane - Free Range Routing (FRR), which allows for the usage of software implementation of routing protocols.
- flexiWAN Agent - The software element that connects flexiEdge with the flexiManage.

flexiManage

flexiManage works as the controller and configuration dashboard for network management. Through it, the devices and the entire network can be configured.

Every registered flexiEdge device is shown in flexiManage together with its status. It collects information for every connected edge device. The information includes general configuration parameters, interfaces, routes and logs. Device configuration can also be changed while the device is running.

In Figure 29 we can see a flexiEdge device connected to flexiManage. flexiManage is responsible for the configuration of the flexiEdge devices and for the collection of statistics. The flexiManage, which can be accessed and configured via a Secure API, communicates with the flexiEdge via HTTPS, used to send configurations to the Edge device.

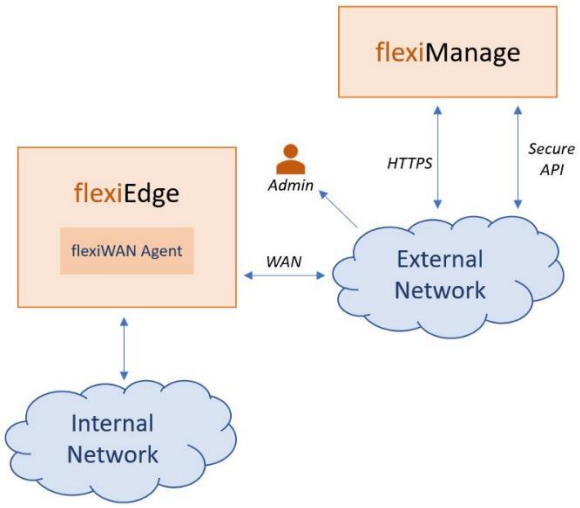


Figure 29 - FlexiWan Architecture

6.5.2 Main Features

Vector Packet Processing (VPP)

The conventional way to process network packets is one by one. VPP changes the way that packets are processed by using a vector of packets instead of one. This helps push the limits of performance when working with a large volume of network traffic.

VPP reads the largest available vector of packets as possible. A packet processing graph is then applied, node by node to the entire packet vector. Each node corresponds to a step in packet processing and instead of fully processing one packet at a time, the whole vector will be processed one step/node at a time.

An example of a VPP graph is shown in Figure 30, where a packet vector would be processed, node by node.

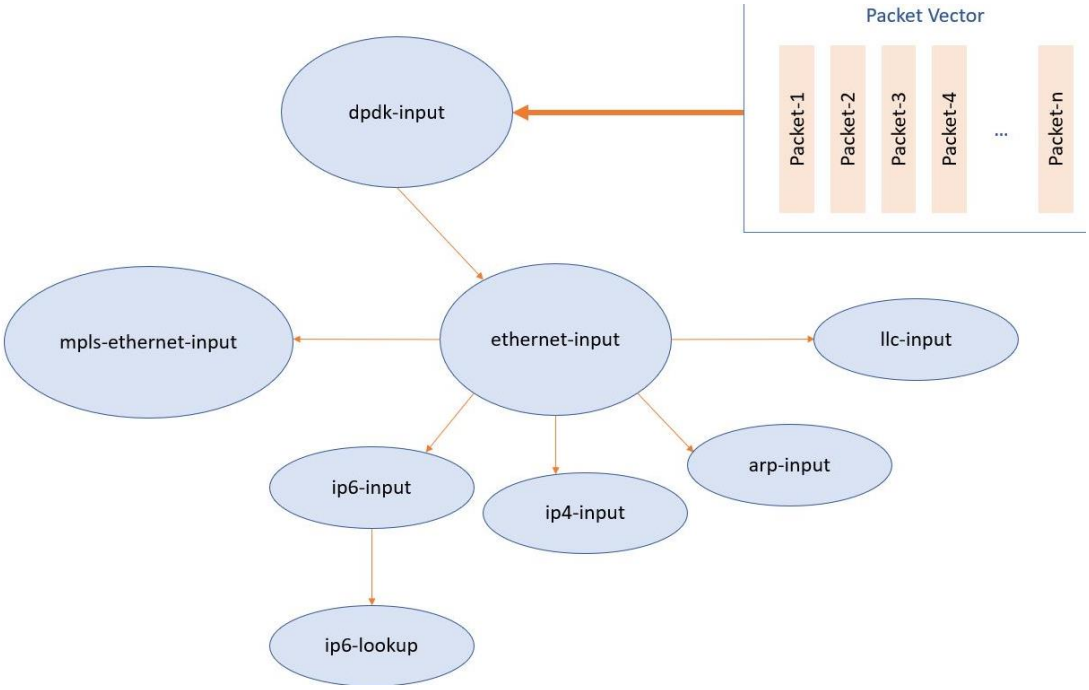


Figure 30 - VPP Graph Example

However, rather than processing the first packet through the whole graph, and then the second packet through the whole graph, VPP processes the entire vector of packets through a graph node before moving on to the next node. As a result, throughput and latency are very stable.

Free Range Routing (FRR)

FFR is an open-source suite to enable the usage of Internet routing protocols for Unix platforms. It implements BGP, OSPF, RIP, IS-IS, and other network protocols. It provides IP routing services, its role is to exchange routing information with other routers, make routing and policy decisions, and inform other layers of these decisions, much like a virtual router. In flexiWan this element is used to enable routing in the flexiEdge's which run on Linux.

Virtual Extensible LAN protocol (VXLAN)

flexiWAN uses encrypted IPSec over VxLAN tunnels between sites. The packet configuration is described in figure 31, VXLAN(UDP) runs over an IPSec Tunnel over GRE. Much like in Nuage Networks, VXLAN is used to provide separation in SDWAN networks by putting different user sites in different VXLANs. Several remote sites can belong in the same VXLAN, despite being in remote locations, but receive/send traffic only in the respective VXLAN.

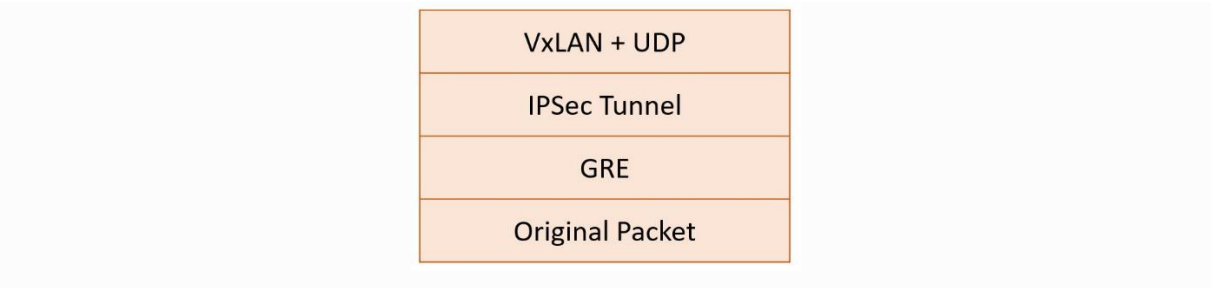


Figure 31 - FlexiWAN Tunnel Encapsulation

The tunnel configuration offers various topologies such as hub and spoke, full mesh or any other customized topology. A full mesh is created between all selected devices so that the devices are connected like point-to-point between their loopback interfaces over the secured tunnel toward the WAN. The LAN routes will be advertised across the tunnel and are able to reach each other.

flexiWAN offers the ability to create a tunnel between every two sites, offers the choice between tunnel key exchange / encryption methods (Pre-Shared Key, IKEv2 and no encryption). Tunnels can be established directly via public IP's or behind NAT and OSPF is used to provide routing between the sites LAN across the tunnel.

6.6 SDWAN solution comparison

Comparing the SDWAN solutions studied previously, the components in each SDWAN technology approached are similar between each other.

All the solutions have an Edge device, which acts as the gateway between the LAN and the SDWAN (Nuage NSG, Cisco vEdge and flexiEdge). They also have a device that acts as the controller (Nuage VSC, Cisco vSmart and flexiManage), a device that acts has a dashboard for configuring the network (Nuage VSD, vManage and flexiManage) and a component which authenticates new network elements (Nuage VSD, vBond and flexiManage).

The only difference between these architectures is how these tasks are divided, flexiWAN concentrates the authentication, controller, and dashboard in a single machine (flexiManage), while Nuage Networks divides the controller into a separate component (VSD and VSC). Cisco goes even further to separate the authentication into another component, the vBond.

The PKI CA of each of the solutions is contained in the authenticating element of the network (vBond, VSD and flexiManage), and is responsible for the key management in the SDWAN.

FlexiWAN is the only solution which employs VPP, meaning that, for large volumes of traffic, it will process packets more efficiently.

Both Nuage Networks and flexiWAN use VXLAN over IPSec by default when establishing tunnels, while Cisco SEN uses only IPSec. This means that it will be less flexible, and less VLANs will be usable (4096 instead of 2^{16}). Layer 2 services will also not be usable with Cisco SEN between sites.

In both FlexiWAN and Cisco SEN, tunnels are established between every two sites, while Nuage Networks uses group encryption between all sites that communicate between them. This means that there will be less SA establishments in Nuage Networks.

In terms of routing, in all solutions the edge devices send the downlink routes to the controller and to other edge devices in the same virtual network using routing protocols (BGP, OSPF, OMP).

Other than being different in the way that they're implemented, they all provide similar services, albeit with different protocols and small differences.

6.7 SDWAN and VPNs

VPNs are the most used solution for providing secure remote access to a network, but they have their pros and cons. As a hardware solution, every new site added to a VPN requires specialized hardware. A simple VPN only provides an encrypted connection between two endpoints, requiring additional solutions for security inspection.

SDWAN, however, is designed to move network routing and security functionality to the edge, for example, instead of having all traffic pass through a single firewall, every edge device becomes one, instead of having to inspect network traffic at the headquarters, it can be done through each Edge device. This brings several benefits. Traffic is optimally routed through the network, providing high-performance and reliable network connectivity and network functions can be deployed throughout the network.

The way that the tunnel establishment is implemented in SDWAN and VPNs is very similar (using IPSec and IKEv2 in most cases).

In SDWAN the control plane is centralized, this means that the communication between Edge devices requires no routing protocol configuration, the controller handles the route installation in these devices, only the routes from each Edges private networks are announced using routing protocols much like conventional VPNs.

The main difference between SDWAN and normal VPNs is that SDWAN is in itself a solution to deploy VPNs, in a flexible and manageable way, and to integrate them with other network functions, without needing specialized devices for these functions.

7. FlexVPN Server/Client (Hub and Spoke topology) – Experiment

To build this FlexVPN topology the GNS3 network tool was used, one FlexVPN Hub (R1) and two FlexVPN Spokes (R2 and R3) were set up, using Cisco IOSv, that supports FlexVPN, One more Cisco router (RA, C3725) was set up, that represents the public network. The objective of this experiment is to configure a FlexVPN Hub-and-Spoke topology with multiple Spokes and Spoke-to-Spoke communication. IPsec was configured using SHA as the algorithm for integrity checks, HMAC used for message authentication, and AH was used as the IPsec protocol. AH was used so that the packets weren't encrypted, and inspection was possible. The network topology is shown in Figure 32. Besides the addresses shown in the Figure, all routers have a loopback address of X.X.X.X (X being their router number) and a tunnel interface with address (10.10.10.X).

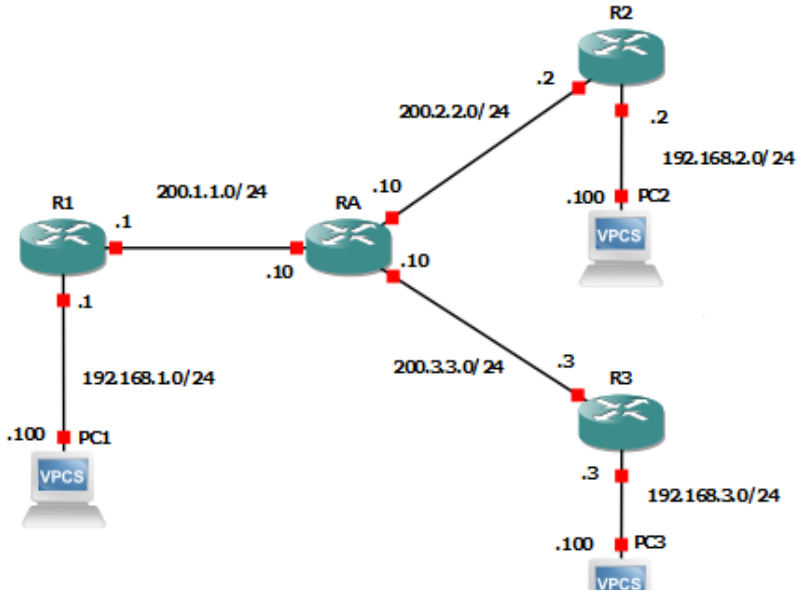


Figure 32 - FlexVPN Site-to-Site topology

After starting up all network elements in GNS3, the Spokes will try to connect to the Hub. The Wireshark capture of the message exchange (captured between the R1 and RA) is shown in Figure 33.

87	156.2005…	3.3.3.3	1.1.1.1	ISAKMP	432 IKE_SA_INIT MID=00 Initiator Request
88	156.4648…	2.2.2.2	1.1.1.1	ISAKMP	432 IKE_SA_INIT MID=00 Initiator Request
89	156.7082…	1.1.1.1	3.3.3.3	ISAKMP	432 IKE_SA_INIT MID=00 Responder Response
90	157.0036…	1.1.1.1	2.2.2.2	ISAKMP	432 IKE_SA_INIT MID=00 Responder Response
91	157.7593…	2.2.2.2	1.1.1.1	ISAKMP	586 IKE_AUTH MID=01 Initiator Request
92	157.9651…	3.3.3.3	1.1.1.1	ISAKMP	586 IKE_AUTH MID=01 Initiator Request
93	158.9028…	1.1.1.1	2.2.2.2	ISAKMP	298 IKE_AUTH MID=01 Responder Response
94	159.0075…	1.1.1.1	3.3.3.3	ISAKMP	298 IKE_AUTH MID=01 Responder Response

```
> Frame 87: 432 bytes on wire (3456 bits), 432 bytes captured (3456 bits) on interface 0
> Ethernet II, Src: c2:01:0c:be:00:00 (c2:01:0c:be:00:00), Dst: 0c:94:1b:8d:75:00 (0c:94:1b:8d:75:00)
> Internet Protocol Version 4, Src: 3.3.3.3, Dst: 1.1.1.1
> User Datagram Protocol, Src Port: 500, Dst Port: 500
v Internet Security Association and Key Management Protocol
```


Figure 33 - IKEv2 Message Exchange

The Initiators (Spokes) send an IKE_SA_INIT request message to the Hub, which responds with IKE_SA_INIT repsonse message. These first messages are unencrypted, as shown. Afterwards, the Initiators send an IKE_AUTH Request to the Hub, which responds with an IKE_AUTH Response, these messages are encrypted. As shown in Figure 34, the payload is Encrypted and Authenticated.

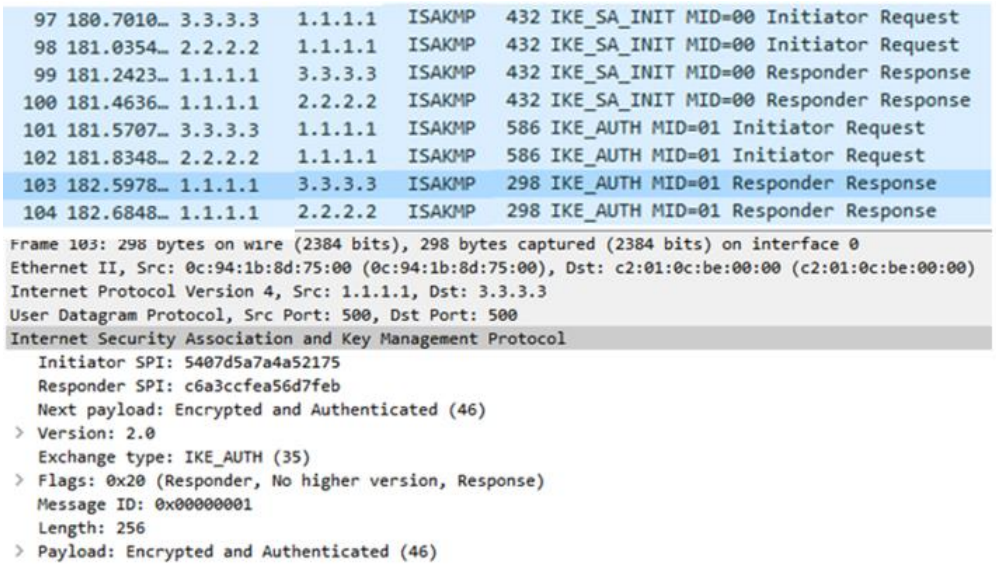


Figure 34 - IKE_AUTH Packet

After this message exchange, R1 will have two SAs established (one for each Spoke), and each Spoke will one SA established (with the Hub). These SAs can be seen in the Hub using the following command: sh crypto ikev2 sa.

Figure 35 shows the SAs in the Hub.

Figure 35 - SAs shown in Hub

For routing to be possible across all the private networks, routing information (in this case, RIP packets), needs to be sent across the VPN tunnels. Figure 36 shows a packet capture between R1 and RA, where the RIP packets sent between the Hub and Spokes can be seen.

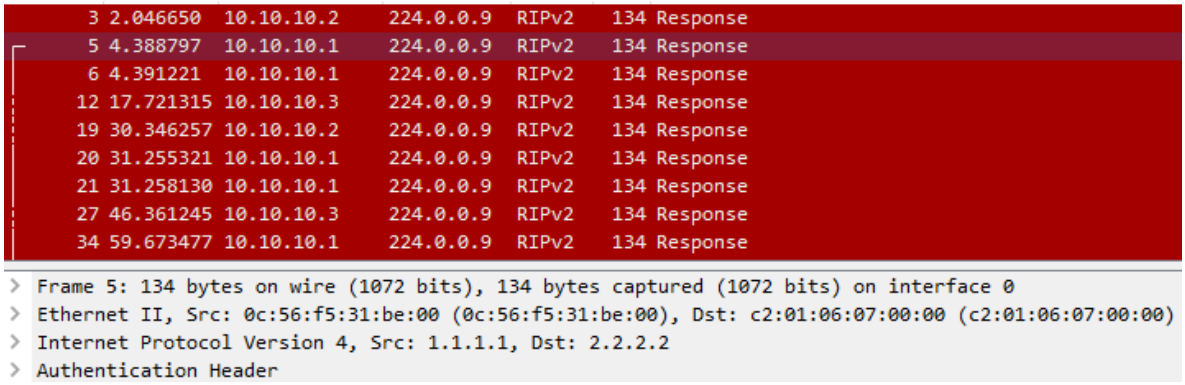


Figure 36 - RIP traffic between Hub and Public Network

The next step is to verify that Spoke-to-Spoke communication is possible. This means that traffic needs to be sent from one spoke to another. In this experiment traffic will go from R2 to R3. To better understand this process, the routing table of R2 should be checked beforehand.

```
R*    0.0.0.0/0 [120/1] via 10.10.10.1, 00:00:10, Tunnel0
      1.0.0.0/32 is subnetted, 1 subnets
O        1.1.1.1 [110/12] via 200.2.2.10, 00:04:56, GigabitEthernet0/0
      2.0.0.0/32 is subnetted, 1 subnets
C        2.2.2.2 is directly connected, Loopback0
      3.0.0.0/32 is subnetted, 1 subnets
O        3.3.3.3 [110/3] via 200.2.2.10, 00:04:56, GigabitEthernet0/0
      10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C        10.10.10.0/24 is directly connected, Tunnel0
L        10.10.10.2/32 is directly connected, Tunnel0
      192.168.2.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.2.0/24 is directly connected, GigabitEthernet0/1
L        192.168.2.2/32 is directly connected, GigabitEthernet0/1
O     200.1.1.0/24 [110/11] via 200.2.2.10, 00:05:06, GigabitEthernet0/0
      200.2.2.0/24 is variably subnetted, 2 subnets, 2 masks
C        200.2.2.0/24 is directly connected, GigabitEthernet0/0
L        200.2.2.2/32 is directly connected, GigabitEthernet0/0
O     200.3.3.0/24 [110/2] via 200.2.2.10, 00:05:06, GigabitEthernet0/0
Router#
```

Figure 37 - R2 Routing table before Spoke-to-Spoke communication

In the routing table of R2, shown in figure 37, we can see that it only has routes to the Hub, and the public addresses of R3 (public interface and loopback). There's a default route (0.0.0.0) to the Hub, via the Hub's tunnel address. To trigger Spoke-to-Spoke communication, a ping is performed from R2 to R3.

Figure 38 shows the network capture, beginning with the traffic indication that the communication is Spoke-to-Spoke, and afterwards, the NHRP resolution request.

87	150.1695…	1.1.1.1	2.2.2.2	NHRP	166 NHRP Traffic Indication
90	150.2099…	3.3.3.3	1.1.1.1	NHRP	154 NHRP Resolution Request, ID=2
91	150.2140…	1.1.1.1	3.3.3.3	NHRP	166 NHRP Traffic Indication
92	150.2201…	2.2.2.2	1.1.1.1	NHRP	154 NHRP Resolution Request, ID=2
94	150.2416…	1.1.1.1	2.2.2.2	NHRP	174 NHRP Resolution Request, ID=2

Figure 38 - NHRP Exchange

After the NHRP Exchange, both Spokes establish a connection between each other. We can see the IKE exchange in Figure 39.

74.419260 2.2.2.2	3.3.3.3	ISAKMP	432 IKE_SA_INIT MID=00 Initiator Request
74.551218 3.3.3.3	2.2.2.2	ISAKMP	432 IKE_SA_INIT MID=00 Responder Response
74.816352 2.2.2.2	3.3.3.3	ISAKMP	586 IKE_AUTH MID=01 Initiator Request
75.087105 3.3.3.3	2.2.2.2	ISAKMP	298 IKE_AUTH MID=01 Responder Response

Figure 39 - Spoke-to-Spoke IKEv2 Exchange

After all of this, the routing table of R2 now has routers to R3's network. This routing table is shown in Figure 40. Two NHRP routes were added, one to 10.10.10.3 and 192.168.3.0 (R3's tunnel interface and private network).

```
R*    0.0.0.0/0 [120/1] via 10.10.10.1, 00:00:33, Tunnel0
      1.0.0.0/32 is subnetted, 1 subnets
O        1.1.1.1 [110/12] via 200.2.2.10, 00:08:30, GigabitEthernet0/0
      2.0.0.0/32 is subnetted, 1 subnets
C        2.2.2.2 is directly connected, Loopback0
      3.0.0.0/32 is subnetted, 1 subnets
O        3.3.3.3 [110/3] via 200.2.2.10, 00:08:30, GigabitEthernet0/0
      10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
C        10.10.10.0/24 is directly connected, Tunnel0
L        10.10.10.2/32 is directly connected, Tunnel0
H        10.10.10.3/32 is directly connected, 00:02:14, Tunnel0
      192.168.2.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.2.0/24 is directly connected, GigabitEthernet0/1
L        192.168.2.2/32 is directly connected, GigabitEthernet0/1
H     192.168.3.0/24 [250/255] via 10.10.10.3, 00:02:14, Tunnel0
O     200.1.1.0/24 [110/11] via 200.2.2.10, 00:08:40, GigabitEthernet0/0
      200.2.2.0/24 is variably subnetted, 2 subnets, 2 masks
C        200.2.2.0/24 is directly connected, GigabitEthernet0/0
L        200.2.2.2/32 is directly connected, GigabitEthernet0/0
O     200.3.3.0/24 [110/2] via 200.2.2.10, 00:08:40, GigabitEthernet0/0
Router#
```

Figure 40 - R2 Routing table after Spoke-to-Spoke communication

Checking the IKEv2 SAs in R2, in figure 41 we can see that it now has 2 SAs, one for each peer. And every communication going from R2 to R3, or vice-versa, will no longer pass through the Hub, but go directly to the peer.

```
Router#sh crypto ikev2 sa
 IPv4 Crypto IKEv2  SA

Tunnel-id Local                 Remote                fvrf/ivrf            Status
1         2.2.2.2/500           1.1.1.1/500           none/none            READY
      Encr: AES-CBC, keysize: 192, PRF: SHA256, Hash: SHA256, DH Grp:2, Auth sign: PSK, Auth verify: PSK
      Life/Active Time: 86400/411 sec

Tunnel-id Local                 Remote                fvrf/ivrf            Status
2         2.2.2.2/500           3.3.3.3/500           none/none            READY
      Encr: AES-CBC, keysize: 192, PRF: SHA256, Hash: SHA256, DH Grp:2, Auth sign: PSK, Auth verify: PSK
      Life/Active Time: 86400/5 sec

 IPv6 Crypto IKEv2  SA
```

Figure 41 - IKEv2 SAs seen in R2 after Spoke-to-Spoke communication

8. Load Balancing - Experiment

To build this FlexVPN topology the GNS3 network tool was used, two FlexVPN Servers (Hub and Slave Cisco IOSv) and two FlexVPN Clients (Spokes 2 and 3 Cisco IOSv), were set up using Cisco IOSv, that supports FlexVPN. One more Cisco router (C3725) was added, it represents the public network.

The objective of this experiment is to configure Load Balancing between the Hub and Slave (routers 1 and 4), and when the Spokes (routers 2 and 3) connect to the VPN, they'll be redirected to the least loaded gateway.

RIP was configured between the FlexVPN Routers and hosts, and OSPF was configured between the FlexVPN routers and R1. This topology is shown in Figure 42.

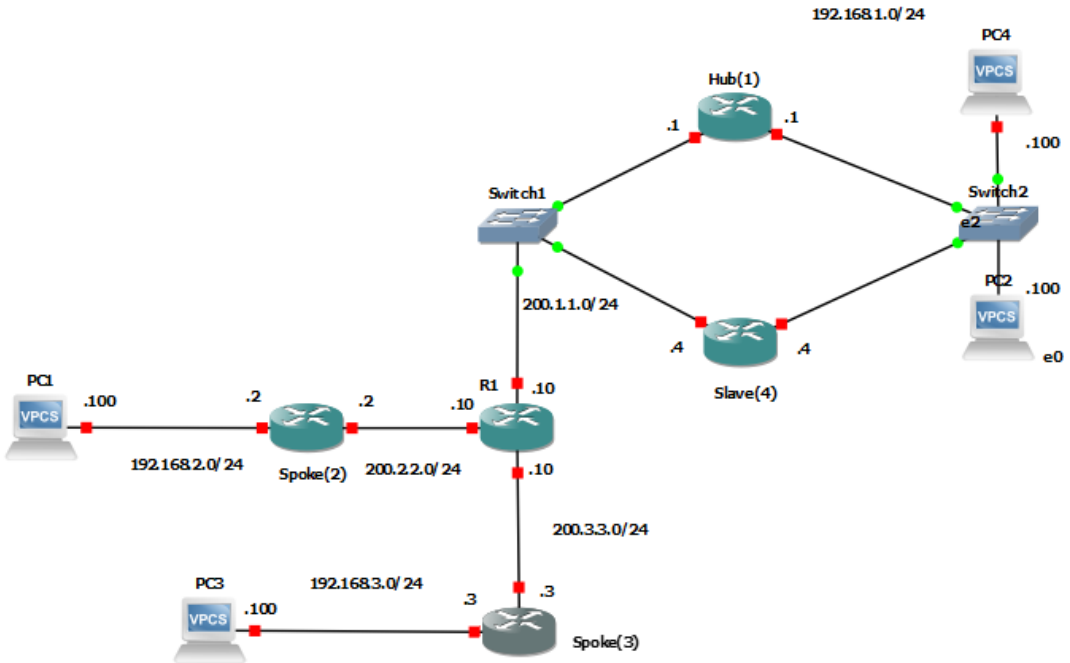


Figure 42 - Load balancing topology

At start-up, both the Hub and the Slave will start sending HSRP packets with the Virtual IP address, to synchronize with each other. The Wireshark capture of these messages is shown in Figure 43.

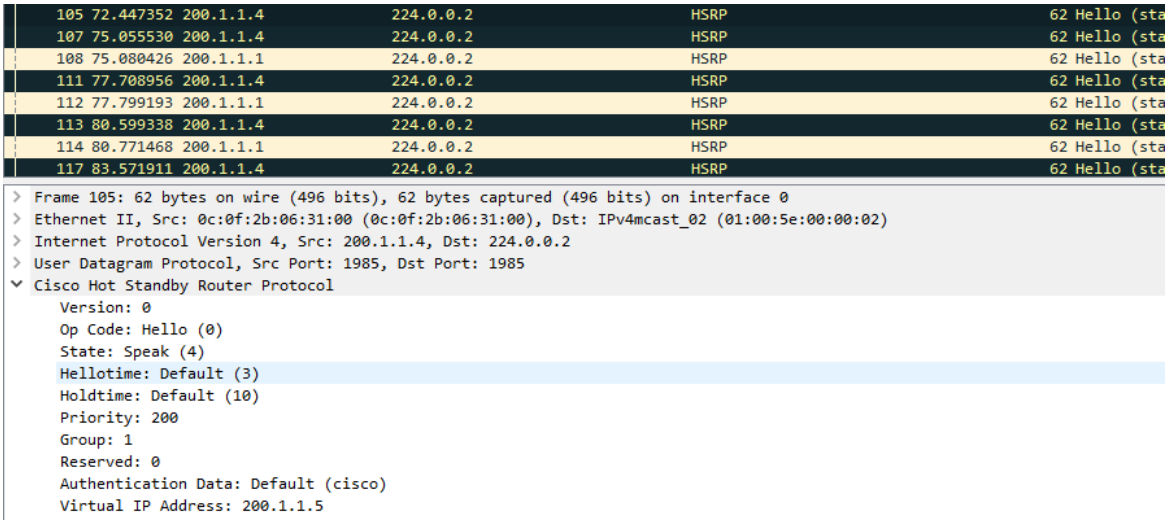


Figure 43 - Wireshark capture of HSRP messages

These HSRP messages are sent by both routers to a broadcast address of 224.0.0.2. We can see the Virtual IP address in the last field of the packet, in this case 200.1.1.5.

After the Active Router is elected, the CLB (Cisco Load Balancing) protocol is then used by the Slave to send the load information to the Virtual IP address, which the Active Router will receive. This exchange can be seen in Figure 44.

Figure 44 - CBL Exchange

After this, the load of each router in the cluster can be checked on the Hub, by using the command: show crypto ikev2 cluster.

Before any clients connect to the VPN, the load in both routers is very similar, as is shown in Figure 45.

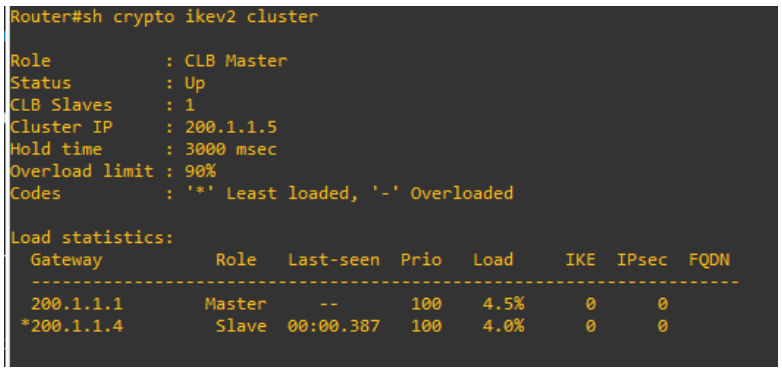


Figure 45 - Cluster Load seen in Hub

When a Spoke tries to connect to the VPN, it will send an IKE_INIT message to the Virtual IP address. The Active Router (Hub) will respond with an IKE_INIT message, but with a redirect notification (mentioned in 2.1.2), as is shown in the Wireshark capture in Figure 46.

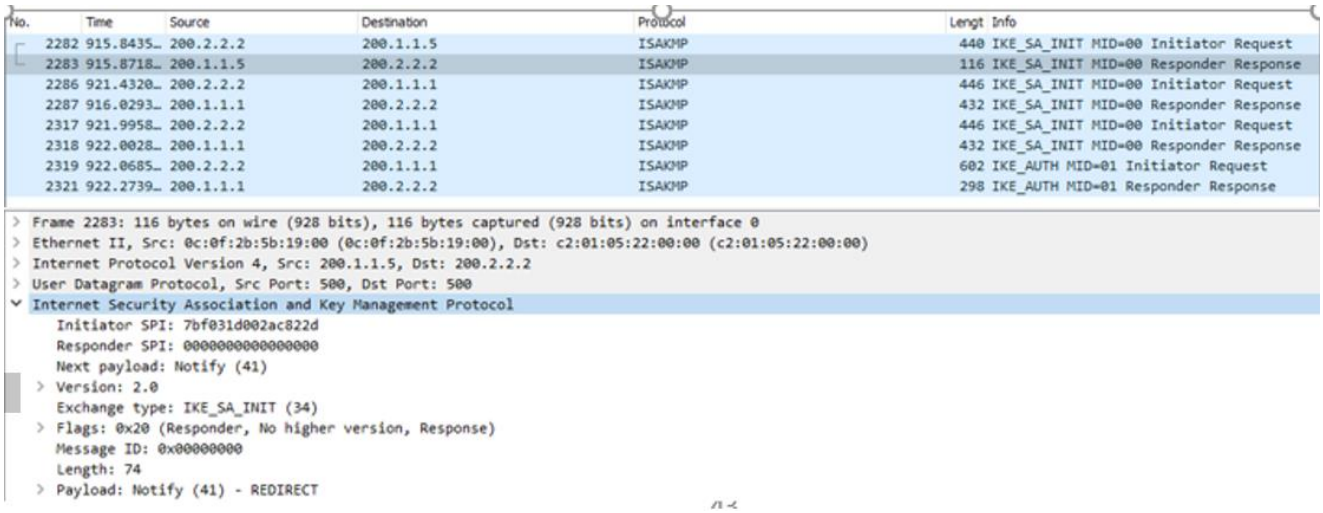


Figure 46 - Wireshark capture IKE exchange Load Balancing

In this case, the Spoke is the first client connecting. Because of this the client is redirected to the Active Router itself, the Hub. After the first client connects, the load on the Hub increases as shown in Figure 47.

```
Router#sh crypto ikev2 cluster

Role           : CLB Master
Status         : Up
CLB Slaves     : 1
Cluster IP     : 200.1.1.5
Hold time      : 3000 msec
Overload limit : 90%
Codes          : '*' Least loaded, '-' Overloaded

Load statistics:
  Gateway           Role   Last-seen  Prio   Load     IKE  IPsec  FQDN
  ------------------------------------------------------------------------
  200.1.1.1        Master     --       100   19.0%      1      2
 *200.1.1.4         Slave  00:00.619   100    3.5%      0      0
```

Figure 47 - Cluster load after client connects

Since the Hub now has a much greater load than the Slave, when we boot-up the next client, it will be redirected to the Slave. This can be seen in Figure 48.

```
Router#sh crypto ikev2 cluster

Role           : CLB Master
Status         : Up
CLB Slaves     : 1
Cluster IP     : 200.1.1.5
Hold time      : 3000 msec
Overload limit : 90%
Codes          : '*' Least loaded, '-' Overloaded

Load statistics:
  Gateway          Role   Last-seen  Prio   Load     IKE  IPsec  FQDN
  ------------------------------------------------------------------------
 *200.1.1.1        Master     --       100   18.7%      1      2
  200.1.1.4         Slave  00:00.211   100   18.7%      1      2
```

Figure 48 - Cluster load after both clients connect

Every client that connects to the VPN will be redirected to the router in the cluster that has the least load, this assures that one of the routers won't be overwhelmed, unless both routers get loaded to full capacity.

9. RIPv2 Attack - Experiment

To build this topology the GNS3 network tool was used. We set up three Cisco routers (C3725), using RIPv2 as the routing protocol, we also set up two VPCs (PC1 and PC2), and an Attacker, that is represented by the Kali Linux system. RIP was configuring as the routing protocol in this topology.

The objective of this experiment is to perform an attack on RIPv2, and direct traffic destined for PC1, to the attacker (Kali). After this is done, authentication in RIP is to be enabled. This topology is shown in Figure 49.

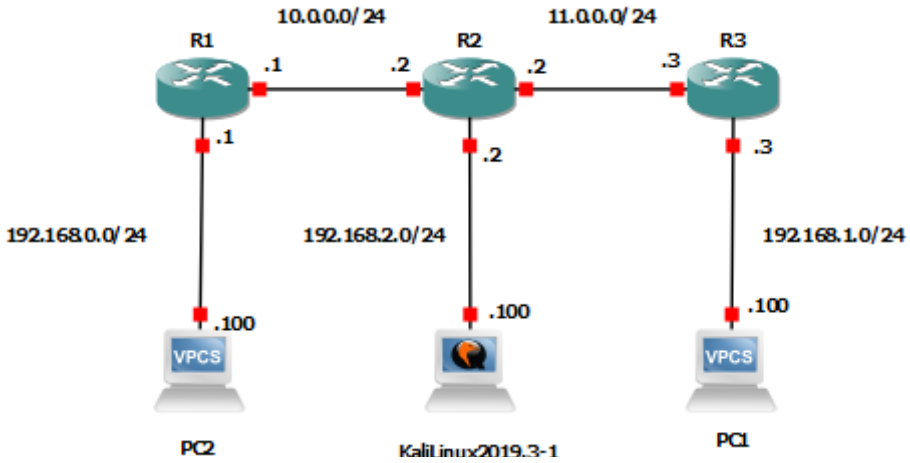


Figure 49 - RIPv2 Exploit topology

9.1 Details

This RIPv2 attack will involve building RIPv2 Response messages in Scapy and sending them from an attacker (Kali Linux), to R2, indicating that the network 192.168.1.0 is the network of the attacker. This will change the routing tables on R2, and any traffic passing through R2 going to that network will go to the attacker's network instead.

Having configured the network to use RIPv2, we can check the routing table on R2 to verify that the routing is correct. The routing table is shown in Figure 50.

```
C    10.0.0.0/8 is directly connected, FastEthernet0/0
C    11.0.0.0/8 is directly connected, FastEthernet0/1
R    192.168.0.0/24 [120/1] via 10.0.0.1, 00:00:01, FastEthernet0/0
R    192.168.1.0/24 [120/1] via 11.0.0.3, 00:00:04, FastEthernet0/1
C    192.0.0.0/8 is directly connected, FastEthernet1/0
R2#
```

Figure 50 – Routing table before the attack

The routing table is correct, R2 can reach the network 192.168.0.0 through R1, and 192.168.1.0 through R3.

We developed a python script using Scapy to perform the attack. The script sends periodic RIP messages with a given subnet address and was installed in the Kali Linux. We can do that by using the following code in Figure 51.

```
54  pkt_evil = Ether()/IP(dst='224.0.0.9')/
55  UDP(sport=520, dport=520)/
56  RIP(cmd=2, version=2)/
57  RIPEntry( AF="IP",RouteTag=0,addr='192.168.1.0',mask="255.255.255.0",nextHop="0.0.0.0", metric=1)
58
59  sendp(pkt_evil, loop=1, inter=2, iface='eth0')
```

Figure 51 - Scapy attack code

The first four lines of the code create the packet. This code will create packets, with destination address of 224.0.0.9 (RIP broadcast), with source and destination port UDP 520 (which is used by RIP) advertising the 192.168.1.0 network. The last line of the code sends the malicious packet, with loop=1, which means that it will keep sending the packet on a loop, with a 2 second interval between messages, through the eth0 interface. The RIP messages can be seen in the Wireshark capture in Figure 52.

114	665.1419…	192.168.2.100	224.0.0.9	RIPv2	66 Response
115	667.1459…	192.168.2.100	224.0.0.9	RIPv2	66 Response
116	669.1495…	192.168.2.100	224.0.0.9	RIPv2	66 Response
118	671.1528…	192.168.2.100	224.0.0.9	RIPv2	66 Response
119	673.1565…	192.168.2.100	224.0.0.9	RIPv2	66 Response
120	675.1604…	192.168.2.100	224.0.0.9	RIPv2	66 Response
121	677.1640…	192.168.2.100	224.0.0.9	RIPv2	66 Response
122	679.1688…	192.168.2.100	224.0.0.9	RIPv2	66 Response
124	681.1725…	192.168.2.100	224.0.0.9	RIPv2	66 Response

Figure 52 – Malicious RIPv2 Packets

When R2 receives these messages, it updates its routing table with a route to 192.168.1.0 through 192.168.2.100 (the attacker). As shown in Figure 53.

```
Gateway of last resort is not set

C    10.0.0.0/8 is directly connected, FastEthernet0/0
C    11.0.0.0/8 is directly connected, FastEthernet0/1
R    192.168.0.0/24 [120/1] via 10.0.0.1, 00:00:10, FastEthernet0/0
R    192.168.1.0/24 [120/1] via 192.168.2.100, 00:00:00, FastEthernet1/0
                    [120/1] via 11.0.0.3, 00:00:08, FastEthernet0/1
C    192.0.0.0/8 is directly connected, FastEthernet1/0
R2#
```

Figure 53 - R2 routing table after exploit

Afterwards, if we try to ping from PC2 to PC1, all traffic will be sent to the attacker.

Figure 54 shows a Wireshark capture on the Kali Linux OS after a ping from PC2 to PC1. This means that the attacker would be able to see the victim's traffic inside his device.

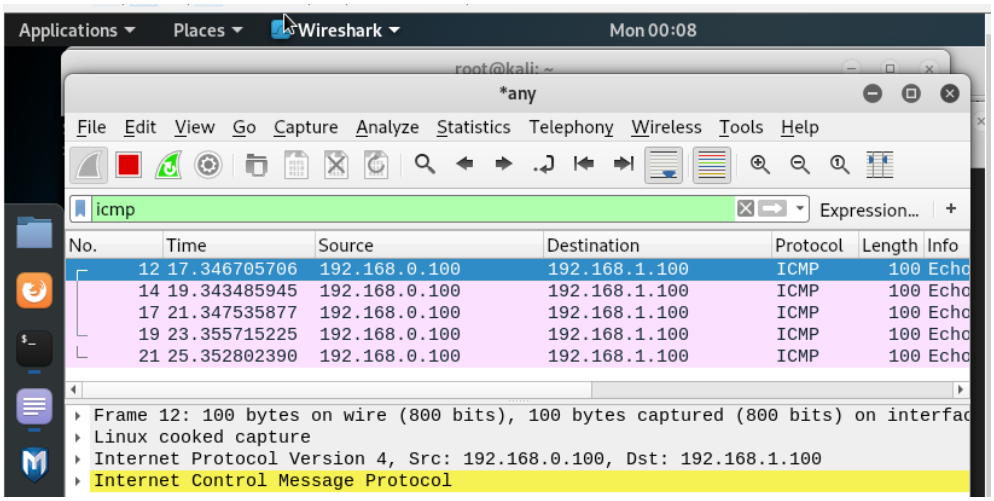


Figure 54 - Wireshark capture on Kali

The source IP address on the ICMP packets is the PC2 address, and the destination is the PC1 address, but the attacker has access to all the traffic meant for PC1.

9.2 Countermeasures

To counter this kind of attack, RIPv2 offers an Authentication method. It's possible to add a password authentication to the RIP packets, in clear-text, or with an MD5 hash. If a router receives RIP messages without this option, with the correct password or hash, it won't accept the packet, and its routes will remain unchanged. To do this all that is necessary is to add the following lines to each router's configuration.

- key chain rip

- key 1
- key-string password
- int f0/0
- ip rip authentication key-chain rip
- ip rip authentication mode md5

Figure 55 shows a RIP packet with MD5 authentication enabled.

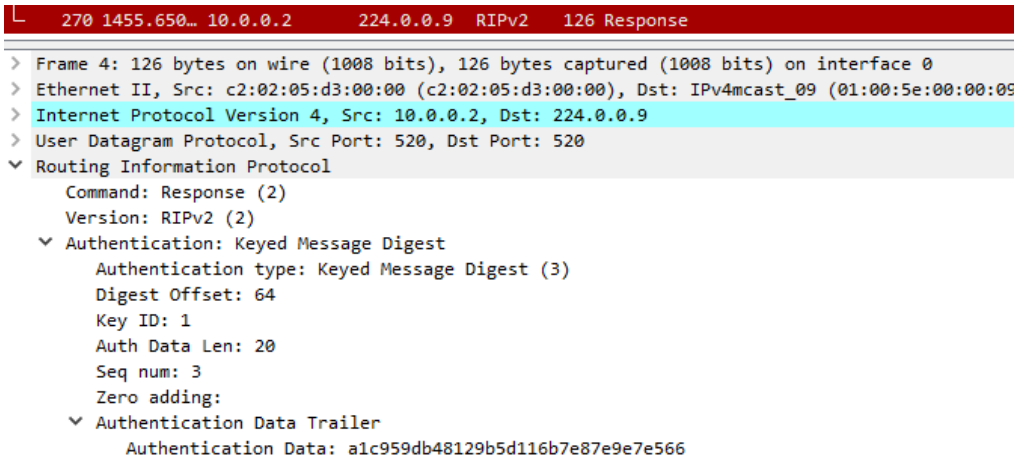


Figure 55 - RIPv2 Packet with authentication enabled

10. SDWAN – Experiments

10.1 Nuage Networks

The objective of this experiment is to connect a client/branch to the SDWAN network by activating a Network Gateway Service (NSG). This NSG will be the gateway between the network where it Is installed, and the controller/manage dashboard. The solution requires the connection to a controller where configurations can be set for the NSG and other devices, not needing to configure each device individually but being able to configure them all at the controller.

For this purpose a Virtualbox image of the NSG was used to connect to the SDWAN, and a Virtualbox Ubuntu image was used as a client (connected only to the NSG and acting like a user in the local network), the remaining infrastructure was made available by Nokia, and follows the topology in Figure 56. The topology shows the VSD and VSC used in the experiment (VSD1 and VSC Internet), it also shows the MPLS underlay provided by Nokia. Two NSGs used where across the Internet from this lab, NSG_NOK and NSG_CLI, both communicate with VSD1 and VSC Internet.

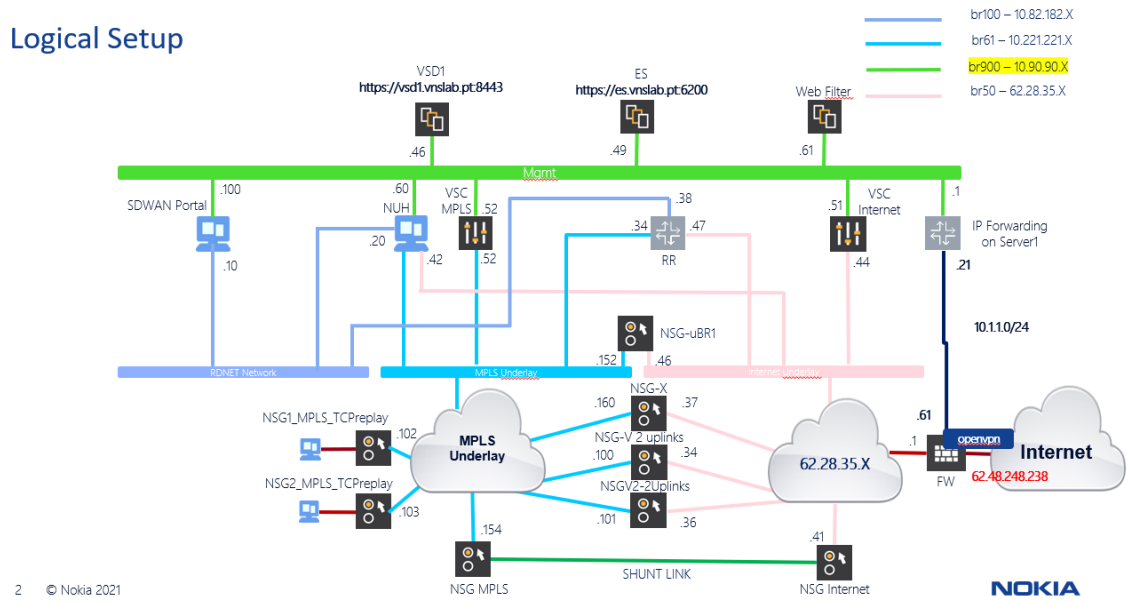


Figure 56 - Nuage Topology

First, OpenVPN had to be used to connect to this environment, which means that the client and NSG used in this experience are not present locally, but across the Internet. From the remaining topology, the elements used in this experiment are the VSD1, the VSC Internet and other NSGs only to check connectivity.

After connecting to the VPN, there's access to the SDWAN portal used to perform the configurations, this corresponds to the VSD device in the topology, which is accessed through https://vsd1.vnslab.pt:8443.

The first step after getting access is to create a Network Service Gateway on the portal, shown in the figure 57, to create the NSG we must give it a name, and choose a template.

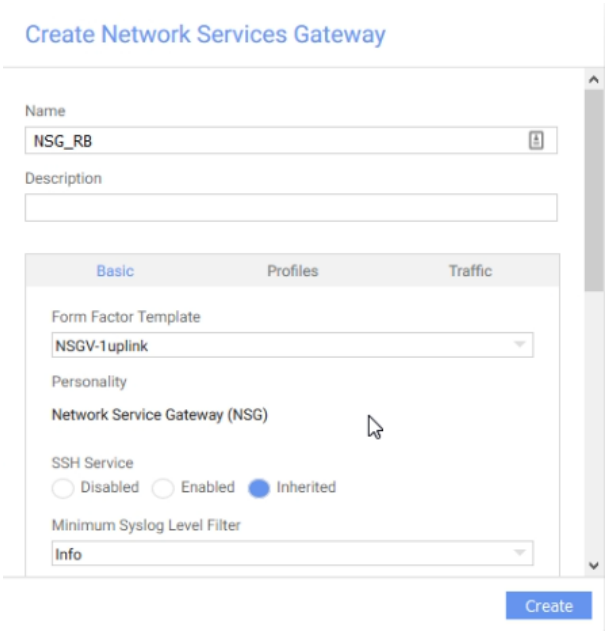


Figure 57 - Create NSG options on the VSD

The created NSG is called NSG_RB, it can then be set up to be activated manually, or through an email sent to the person installing the NSG.

For the NSG to have network access to both the LAN and the SDWAN, 3 ports are created, 2 are for connections to the LAN and 1 is for Network access to the SDWAN. Figure 58 shows the ports created.

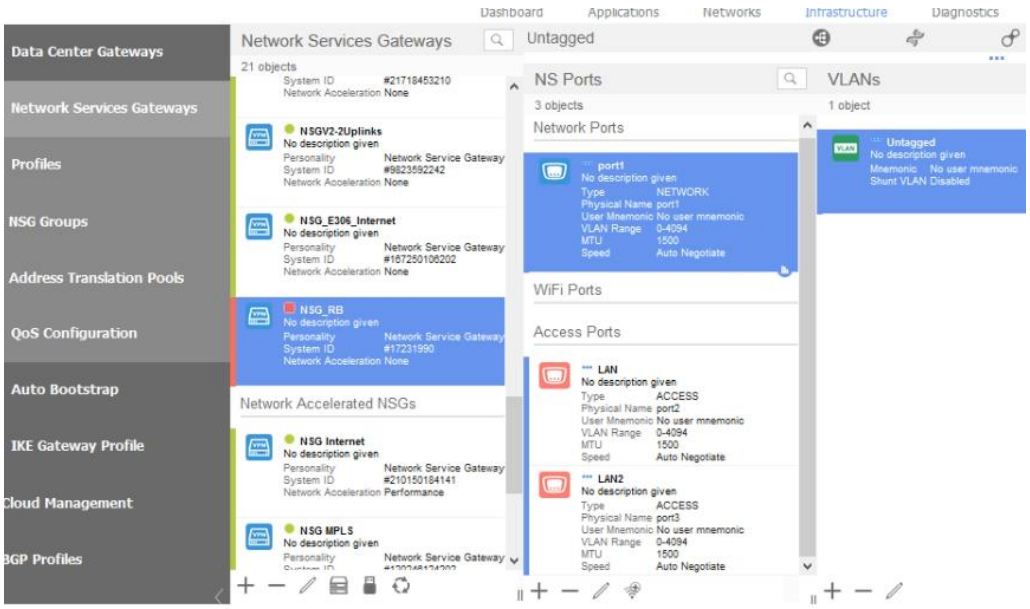


Figure 58 - NSG Port Information on the VSD

Even without being activated, the NSG allows for a normal Internet connection, which means that the NSG can be directly activated from the client (Ubuntu). In the client OS (Ubuntu in virtual box connected to the NSG) we access the activation email and follow the activation link to activate the NSG, as shown in Figure 59.

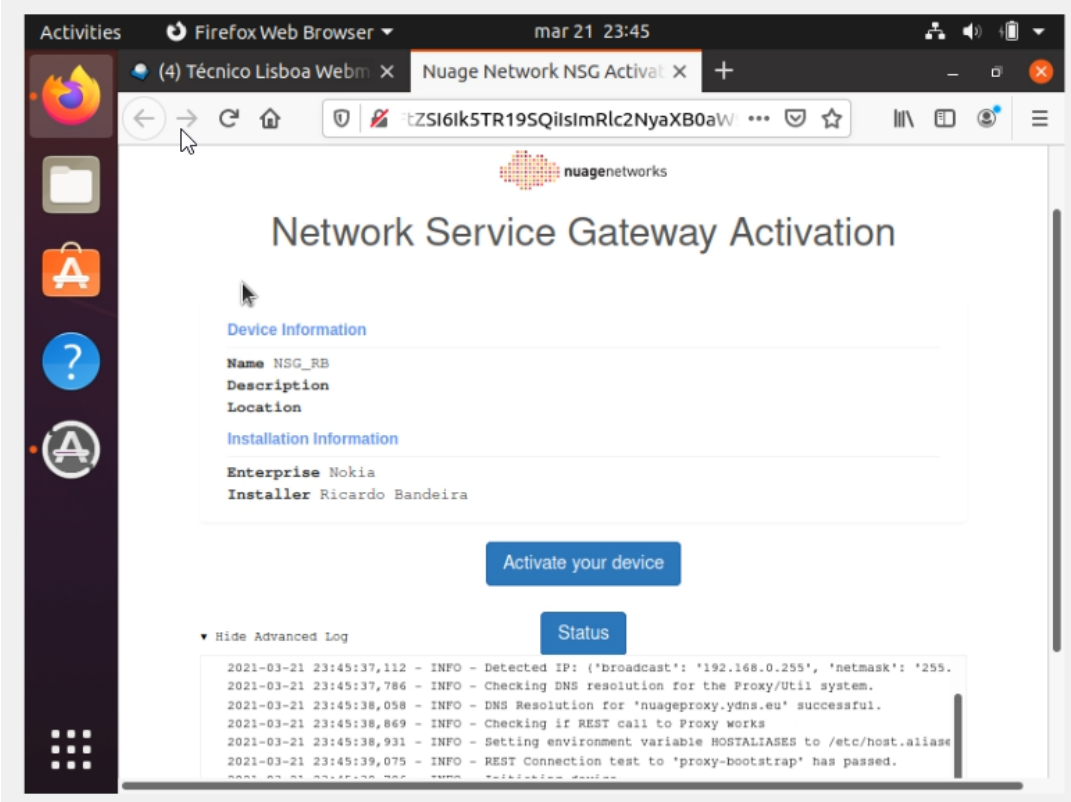


Figure 59 - NSG email activation on the Ubuntu device

Afterwards, the NSG appears as active in the SDWAN portal. This means that the device configurations can now be set through the VSD. Figure 60 shows the NSG device information (the device is active).

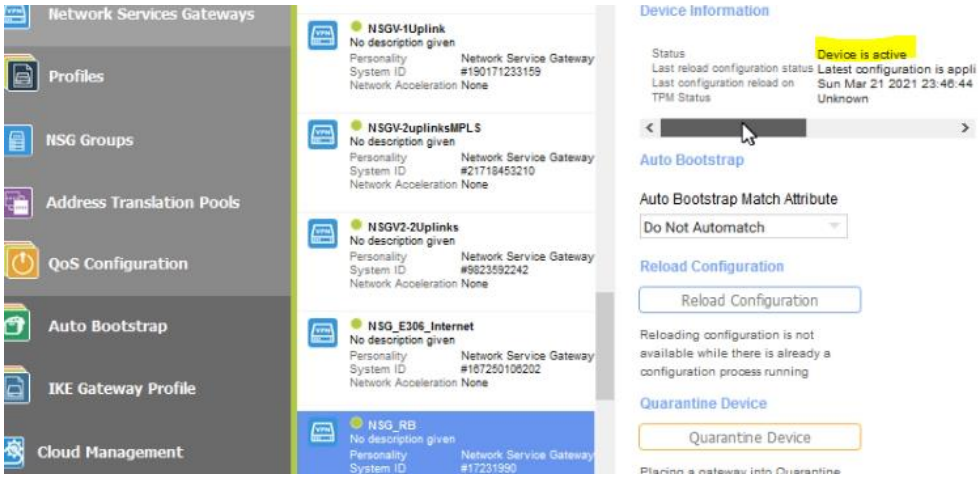


Figure 60 - NSG Device information on the VSD

To deploy a VPN between sites, a subnet needs to be created, along with a virtual port, this port will be used by the NSG to connect to the subnet. Every NSG added to this subnet would be connected to a virtual network with the other NSGs in the subnet, like adding a new site in a Site-to-site VPN. In this case, we'll create the subnet 192.168.23.0/24, and create the vPort on the NSG. Figure 61 shows these configurations.

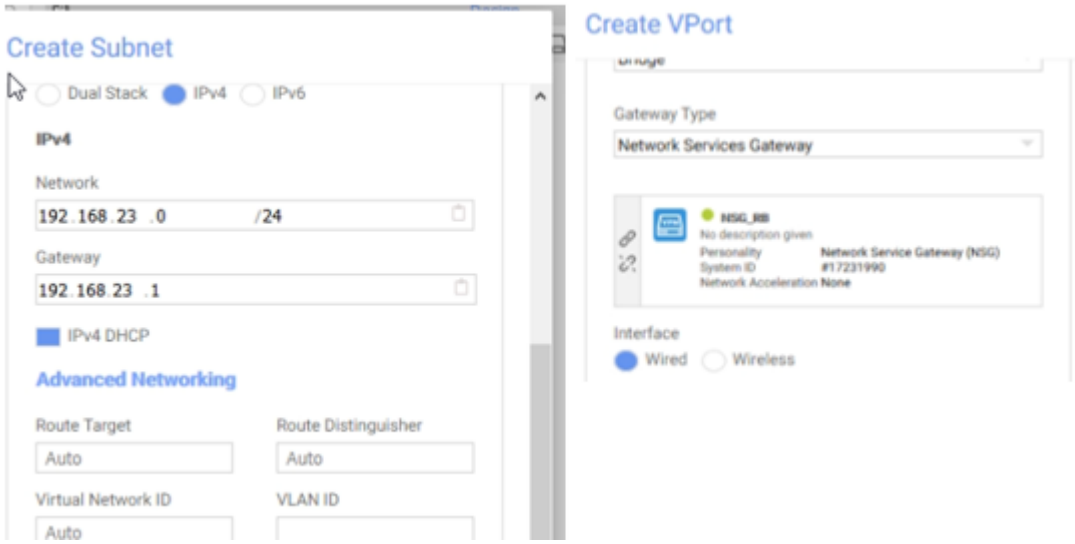


Figure 61 - Subnet and vPort configuration on the VSD

In that subnet we can then enable DHCP to set the clients IP's and DNS settings, for example the DNS server will be 8.8.8.8, and the DHCP pool will be 192.168.23.100-254. Figure 62 shows the DNS and DHCP configuration.

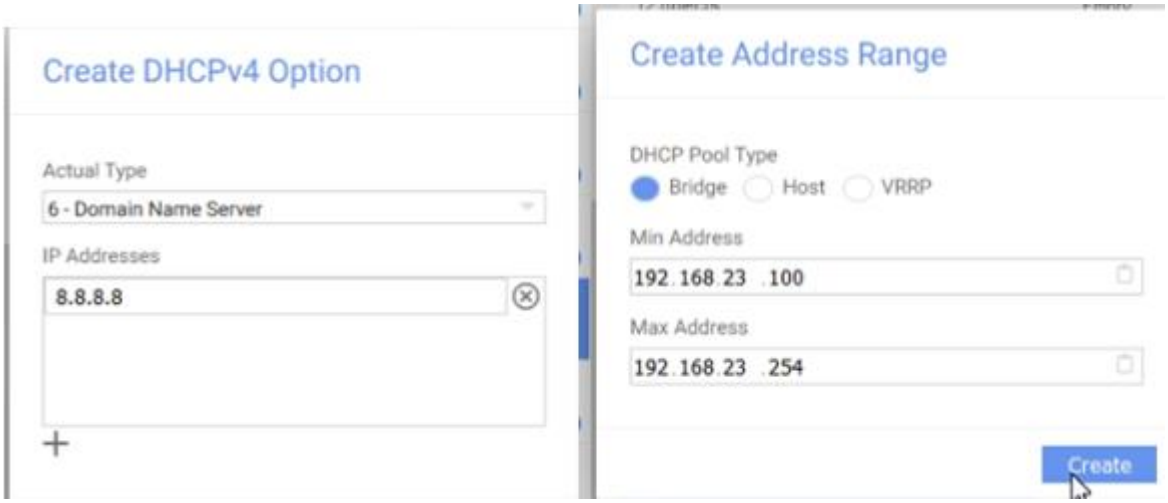


Figure 62 - DNS and DHCP configuration on the VSD

Afterwards, the client will be attributed an IP address belonging to the created subnet. As the image below shows, the client got the IP 192.168.23.251 when before it had no IP address. Figure 63 shows the output of the command "ip a" which shows the device's interfaces.

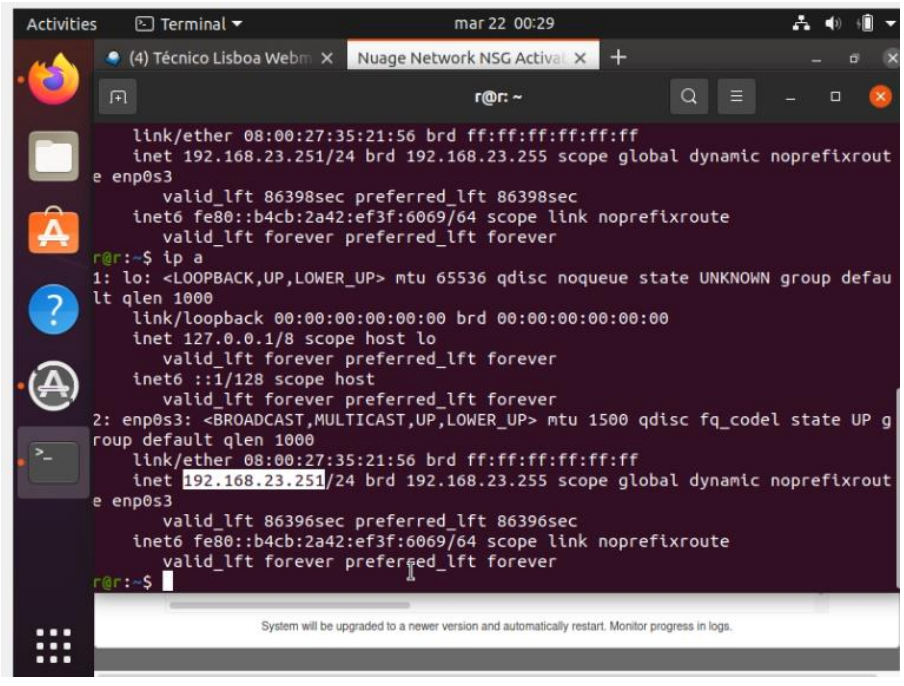


Figure 63 - Ubuntu client interfaces

When traffic is sent from the client to another subnet, we can inspect the packets passing by the NSG, in this case, we'll ping an NSG in the network 10.44.127.0/24, the IP 10.44.127.3 belongs to an NSG setup by Nokia for this experiment, as shown below in Figure 64. these addresses belong to the subnet "Subnet-NSGV-2Uplinks".

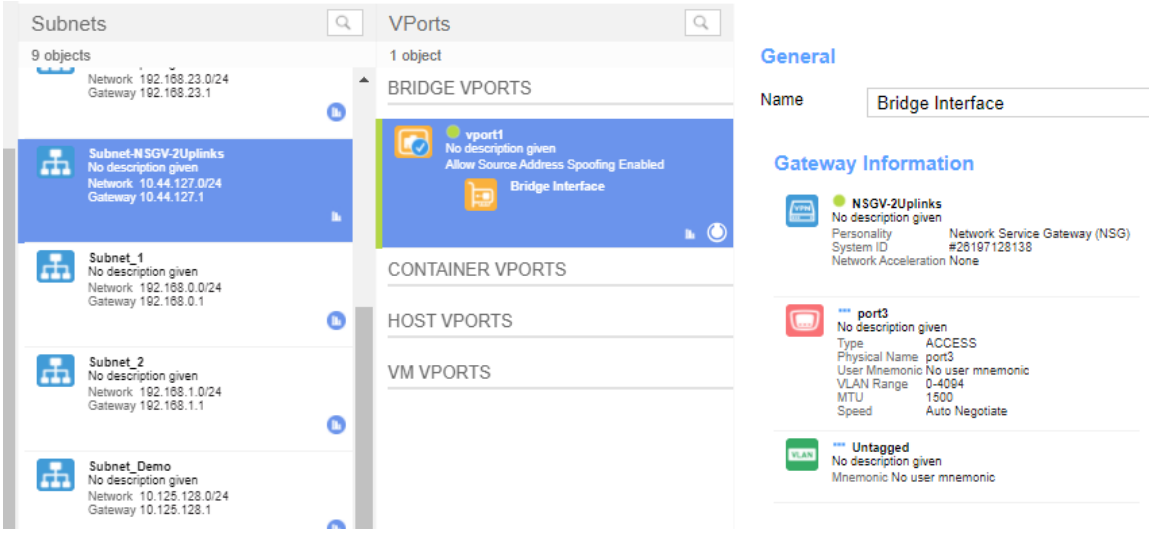


Figure 64 - Destination Subnet and NSG on the VSD

The Figure 65 below shows a Wireshark capture of a ping from the client (Ubuntu OS). With these packets we can check the connectivity to the SDWAN.

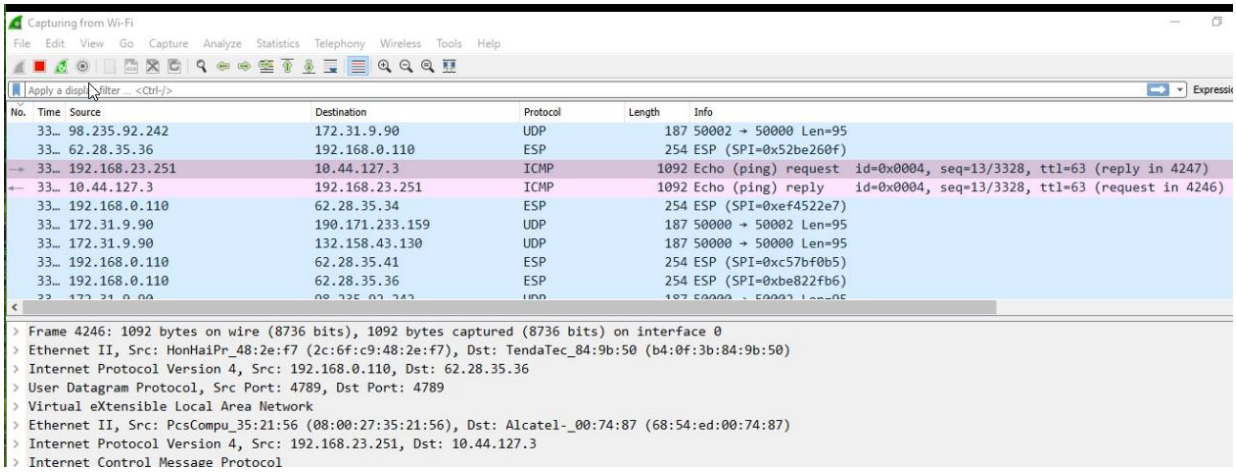


Figure 65 - ICMP Wireshark capture

We can also analyze the packets sent and verify the VXLAN encapsulation, where the VNI (VXLAN Network Identifier) is sent, in this case the VNI is 1232922, as shown in Figure 66.

```
Virtual eXtensible Local Area Network
  Flags: 0x0800, VXLAN Network ID (VNI)
  Group Policy ID: 0
  VXLAN Network Identifier (VNI): 1232922
  Reserved: 0
```

Figure 66 - VXLAN header

For here on, the NSG connects our LAN to the SDWAN and we can configure our NSG to have connectivity to any other branch we decide to setup, or that we have already set up before. All traffic sent to the internet breaks out locally and doesn't go through the tunnels established by the SDWAN, unless it is specifically configured for that to happen, this means that only traffic directed to other sites will pass through these tunnels. In regards to Hub-and-Spoke topologies, the traffic can be configured to always pass through the Hub, but it's not configured that way by default.

One thing that this SDWAN solution also allows for, is the enabling/disabling of IPSec encryption on already running systems. At any time, the encryption can be enabled and the NSG will handle the encryption when it routes network traffic. This is shown in figure 67.

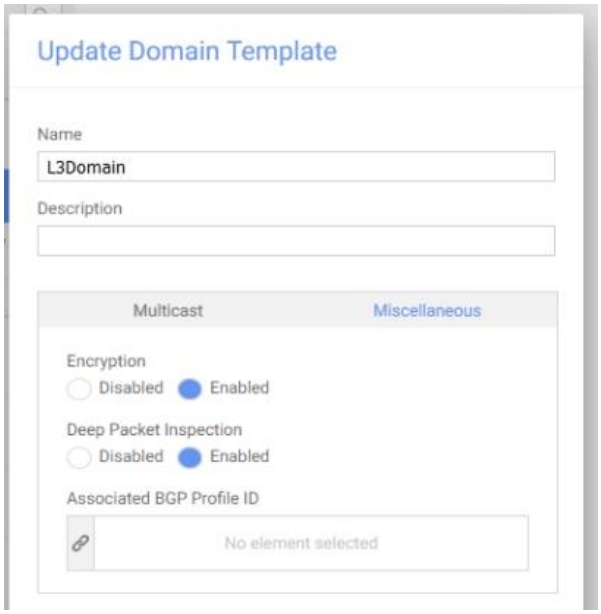


Figure 67 - Encryption configuration on the VSD

After enabling the encryption, all traffic sent by the NSG is encrypted using ESP. The image below shows a ping between the client and the same endpoint used in the last ping. The size was set to 1000 so the ICMP packets can be found even when encrypted. This capture is shown in Figure 68. The protocol for the ICMP packets is ESP and they are encrypted, the only reason that we know it's an ICMP packet is because of the length of the packet (1166, which is the 1000 for the ICMP payload, and 116 for the headers).

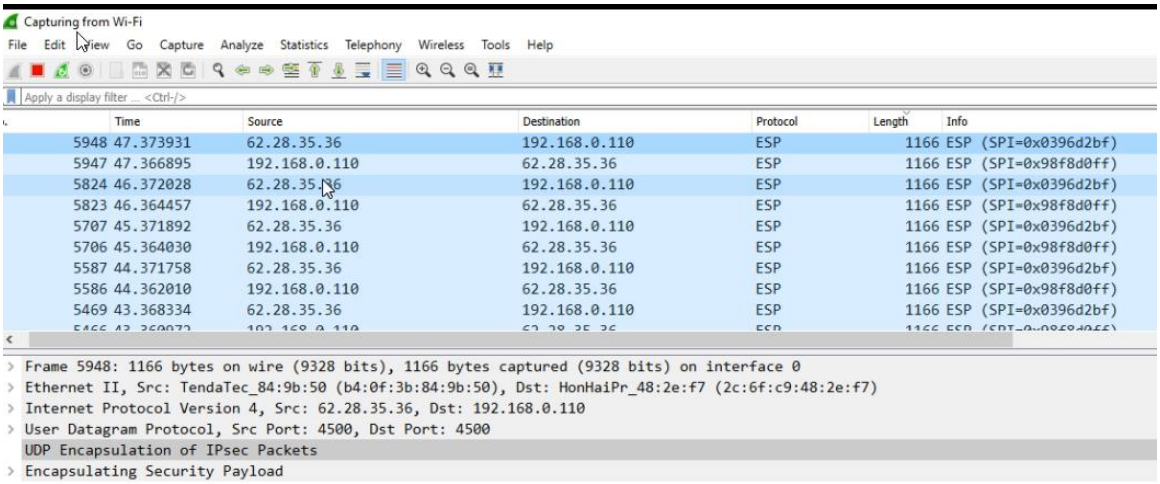


Figure 68 - Encrypted Wireshark capture

The client can now has encrypted communications with other sites across the internet connected to the SDWAN. This encryption is group based, which means that there's a shared key between all elements of a subnet, the key distribution is handled by the Key Server, which also preforms the rekey (key redistribution) when the lifetime of the key expires.

Using SSH to connect to the VNS we can see input some commands to learn information. The command "show vswitch-controller vswitches" shows us the active NSGs (the NSGs are an altered version of the vSwitch software used in SDN), the one that was created in this report is NSG-172.31.9.90, which has an uptime of 32 minutes at this time. This is shown in Figure 69.

```
*A:vsc2.vnslab.pt# show vswitch-controller vswitches

===============================================================================
Virtual Switch Table
===============================================================================

-------------------------------------------------------------------------------
Legend: * -> Primary Controller ! -> NSG in Graceful Restart
-------------------------------------------------------------------------------
VSwitch Name            System-Id         Uptime        Num VM/Host/Bridge/Cont
                        Mesh-Group-ID                   Num Resolved
-------------------------------------------------------------------------------
*NSG-26.197.128.138     26.197.128.138    0d 19:45:48   0/0/1/0
                        0                               0/0/1/0
*UBR-96.135.3.101       96.135.3.101      17d 04:08:07  0/0/0/0
                        0                               0/0/0/0
*NSG-98.235.92.242      98.235.92.242     17d 04:08:06  0/0/0/0
                        0                               0/0/0/0
*NSG-132.158.43.130     132.158.43.130    17d 04:13:54  0/0/0/0
                        0                               0/0/0/0
*NSG-172.31.9.90        172.31.9.90       0d 00:32:56   0/0/1/0
                        0                               0/0/1/0
-------------------------------------------------------------------------------
Number of Virtual Switches: 5
-------------------------------------------------------------------------------
===============================================================================
*A:vsc2.vnslab.pt#
```

Figure 69 - vSwitch (NSG) Table

Using the command ‘show vswitch-controller vswitches vs-name “NSG-172.31.9.90” detail’ allows the verification the Virtual Switch Table. This is shown in figure 70.

```
*A:vsc2.vnslab.pt# show vswitch-controller vswitches vs-name "NSG-172.31.9.90" detail

===============================================================================
Virtual Switch Table
===============================================================================
Vswitch Name                : NSG-172.31.9.90
Datapath-Id                 : 172.31.9.90
Mesh-Group-Id               : 0
Uptime                      : 0d 00:34:30       VM Count              : 0
Graceful Restart Support    : Disabled
Num of hostIf               : 0                 Num of bridgeIf       : 1
Num of multiVMs             : 0
Num of Containers           : 0
Cntrl. role                 : primary           Cntrl. Conn. type     : tls
Num cntrl role changes      : 0
Last cntrl role change at   : -
Cntrl. crl lookup           : false
Cntrl. Conn. mode           : secure
Cntrl. Conn. state          : ready
Cntrl. client verification  : true
Cntrl. client IP verification : false
Peer IP for resiliency      : -
Gateway Topic               : nuage_gateway_id_172.31.9.90
Gateway Retry/Audit Time    : 577
XMPP error code             : 0
XMPP error text             : (Not Specified)
Trusted                     : Trusted
Quarantine Reason           : Trusted
Static Peer                 : False
XMPP Tls Profile            : n/a
OF Tls Profile              : vsc-tls-profile
Ovsdb Tls Profile           : n/a
Ovsdb Conn Type             : none
Num Plcy Req Sent           : 11
Num Plcy Resp Rcvd          : 10
```

Figure 70 - NSG details

Figure 70 shows the NSG name and ID, uptime, control connection type (TLS), if it’s trusted, the OpenFlow profile and the number of Policy requests sent, and Policy responses received. There’s some more information but these are the most important.

The NSG connections can be monitored with the command:

- show vswitch-controller vswitches vs-name “NSG-172.31.9.90” detail connections

This shows the NSG IP addresses, and a number of informations on IPSec, VXLAN, NAT and network information. The VSwitch Connections are shown in Figure 71.

```
*A:vsc2.vnslab.pt# show vswitch-controller vswitches vs-name "NSG-172.31.9.90" detail connections

===============================================================================
Virtual Switch Connections
===============================================================================
VSwitch Name            : NSG-172.31.9.90
Uplink IP               : 85.243.175.182 Uplink System ID      : 70.148.144.156
Private IP              : 10.0.2.15
Underlay ID             : 14606          Uplink ID             : 15255
OpenFlow Conn State     : ready          OpenFlow Uptime       : 0d 00:35:40
OpenFlow Conn. Port     : 6633           OpenFlow Conn Type    : tls
OpenFlow Version        : 1              OpenFlow Nego. Version: 1
Behind NAT              : yes            PAT Uplink            : yes
AUX Uplink              : no             AUX Uplink active     : no
Hold Time(sec)          : 15             Echo Time(sec)        : 5
JSON Conn. State        : Up             JSON Session Uptime   : 0d 00:35:36
VXLAN DTLS Sess. Status: Up              VXLAN DTLS Public Port: 61597
VXLAN DTLS Sess. Uptime: 0d 00:35:39
IPsec DTLS Sess. Status: Up              IPSec DTLS Public Port: 49497
IPsec DTLS Sess. Uptime: 0d 00:35:40
RPC Last Failed Request: n/a
RPC Last Failed Respons: n/a
Uplink Order            : 1
Uplink Preference       : Primary        Uplink Underlay Name  :
No. Uplink Connection   : 1              Uplink IP Addr        : 85.243.175.182
GR Enabled              : False          GR Restart Int.       : 0d 00:00:00
GR Key Expired          : False          Site ID               : 0
Cntrl Local Addr        : 62.28.35.44    Controllerless Timer  : 0

Uplink information
Uplink Order            : 1
Uplink IP Addr          : 85.243.175.182 Uplink Preference     : Primary
Behind NAT              : yes            PAT Uplink            : yes
Underlay Id             : 14606          Numa Id               : 0
AUX Uplink              : no
Uplink Order            : 0
Press any key to continue (Q to quit)
```

Figure 71 - NSG Connection details

Figure 71 shows the uplink IP, which is the public IP address used by the NSG (85.243.175.182), private IP address (10.0.2.15), OpenFlow connection port and type, if the NSG is behind NAT; we can also verify if the VXLAN and IPSec sessions are online, and through which ports they communicate.

ACCESS CONTROL

Access control lists can also be deployed almost instantly using SDWAN. A rule can be created to deny ICMP packets from the configured NSG network to the referred branch site. This is shown in Figure 72. To create a rule, first a name is assigned, and then match criteria are defined. In this case any IPv4 ICMP packets will be dropped.

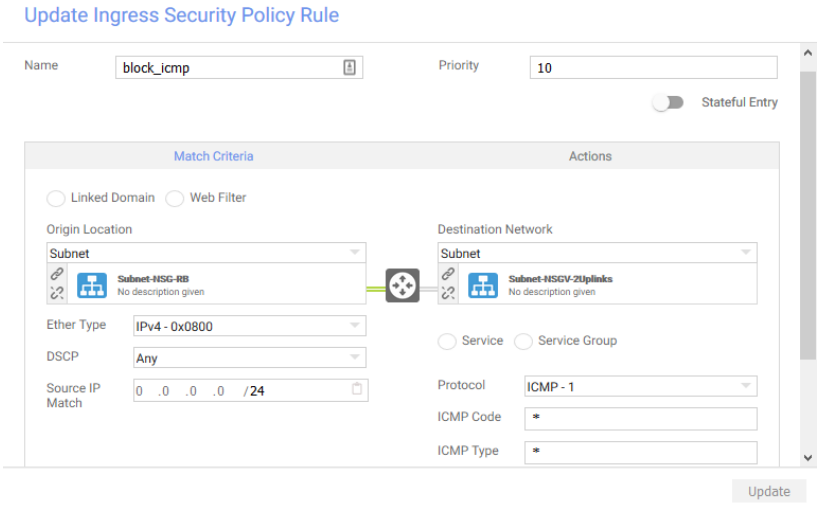


Figure 72 - Policy Rule Configuration on the VSD

10.2 CISCO SEN

The objective of this experiment is to create a Hub-and-Spoke VPN in the Cisco SDWAN network between 4 Cisco Edge devices using IPSec. For this purpose a demo SandBox, available online on Cisco's website was used (https://developer.cisco.com/sdwan/sandbox/). It allows the usage of 4 vEdge devices, and vManage, vBond and vSmart but the topology cannot be changed.

The topology used follows Figure 73, there's the vManage, vSmart and vBond, and also an Edge device that will act as a Hub, while the other vEdges are the gateway to their own local network. Between these elements is the Public Internet.

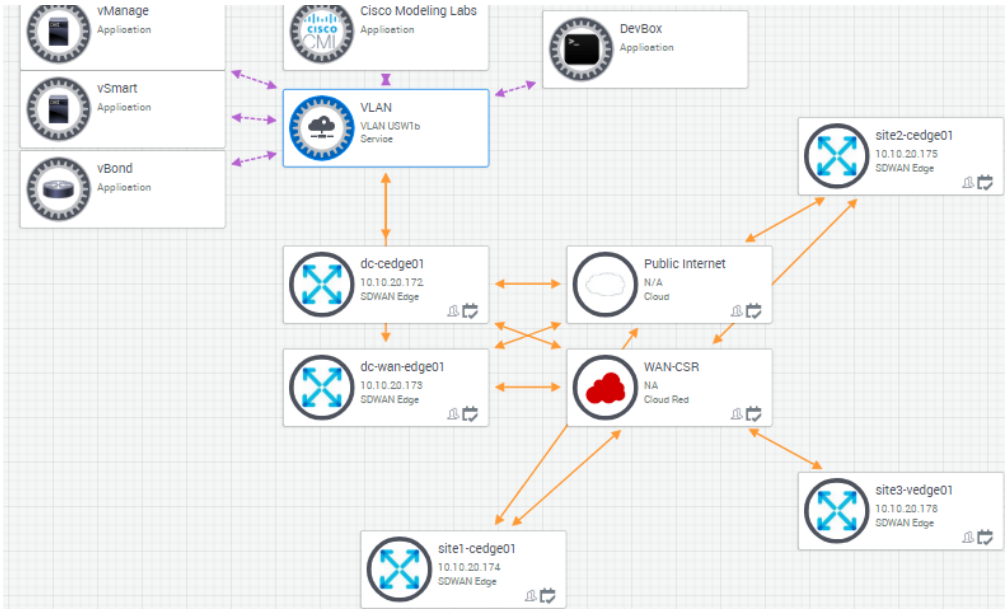


Figure 73 - Cisco SEN Topology

The Cisco AnyConnect VPN client needs to be used to access this environment. All the devices are already created in the sandbox, they need only to be configured. The vManage IP is 10.10.20.90. It can be accessed via HTTPS or SSH while all other elements can only be accessed via SSH.

Each vEdge Device has its own Site-ID, which is used in the configurations, the site IDs for the edge devices are in Figure 74, also shown is the System IP, Device Model, and Host-name.

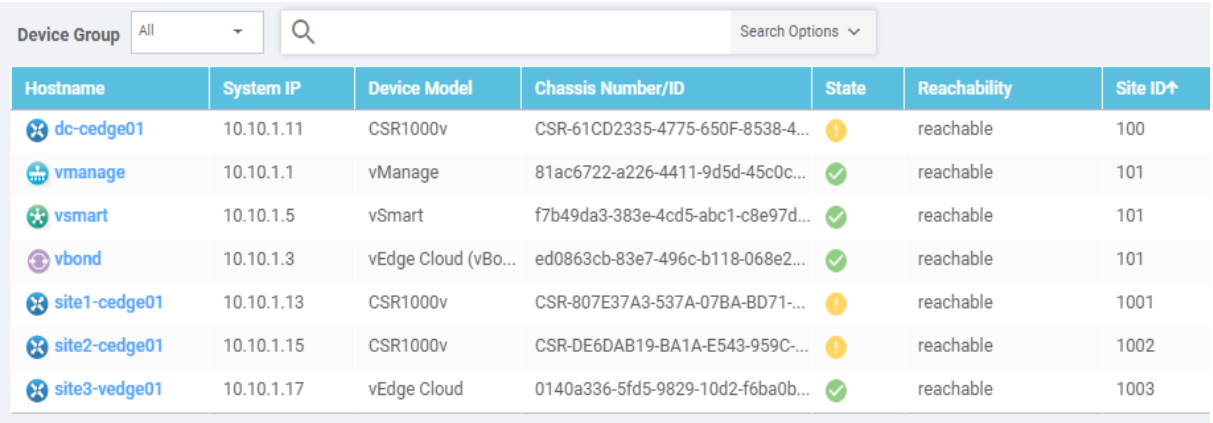


Figure 74 - Network elements information on the vManage

In order to establish a VPN in this solution, groups of interest need to be established and Site Lists need to be created, as shown in figure 74, two site lists are created: a Hub list with dcEdge1, and a Spoke list, with all other vEdges. A VPN list, named Corp-VPN-List, can now be configured using these devices.

Figure 75 - Groups of Interest configuration on the vManage

After the groups of interest are configured, a topology can be chosen from the options in figure 75 (Hub-and-Spoke, Mesh, or Custom Control.

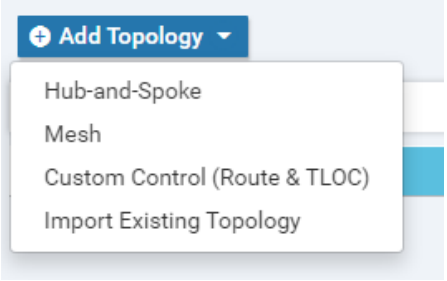


Figure 76 - Topology options on the vManage

Afterwards is the definition of which roles the groups of interest have to perform, in this case which sites are Hubs, and which sites are Spokes, as can be seen in Figure 77.

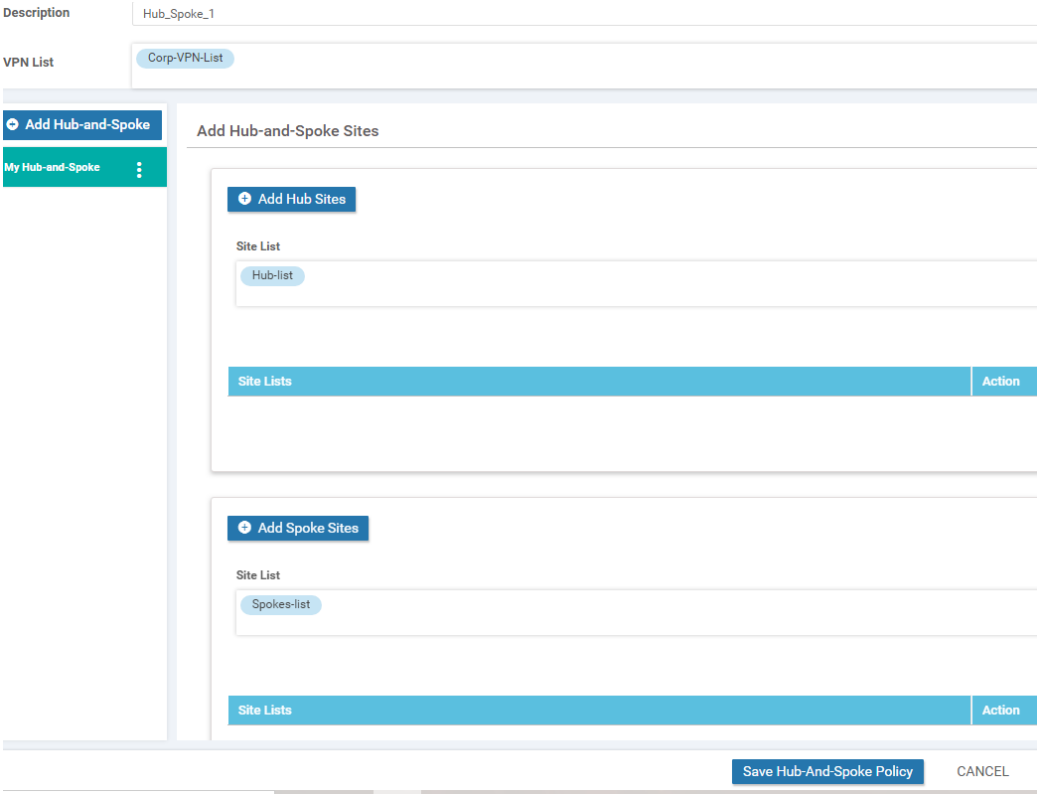


Figure 77 - Hub and Spoke topo configuration on the vManage

Then the policy just needs to be saved and the devices updated for the VPN to start working.

Connecting to the vSmart with IP 10.10.20.70, via SSH, and using the command “show omp peers” prints the devices with which the vSmart has OMP sessions. Figure 78 shows that the vSmart communicates via OMP with all vEdges. The Site-ID is shown per peer, we can see that IDs 100, 1001, 1002 and 1003, which belong to the peers, are the Site-IDs shown for the vEdges in Figure 74, also shown is the System IP under the “PEER” tab.

```
vsmart# show omp peers
R -> routes received
I -> routes installed
S -> routes sent

                        DOMAIN    OVERLAY   SITE
PEER            TYPE    ID        ID        ID        STATE    UPTIME           R/I/S
-----------------------------------------------------------------------------------
10.10.1.11      vedge   1         1         100       up       0:00:32:28       4/0/6
10.10.1.13      vedge   1         1         1001      up       0:00:33:11       2/0/8
10.10.1.15      vedge   1         1         1002      up       0:00:32:55       2/0/8
10.10.1.17      vedge   1         1         1003      up       0:00:36:14       2/0/8
```

Figure 78 - OMP Peers on the vSmart

Using the command “show omp routes” prints all routes learned through OMP. Figure 79 shows the usage of the command on the vSmart. The vSmart has received routes from all peers, related to each private network. It has a default route (prefix 0.0.0.0/0) through peer 10.10.1.11 which is the Hub device. The encapsulation on all these routes is IPSec.

```
vsmart# show omp routes
Code:
C   -> chosen
I   -> installed
Red -> redistributed
Rej -> rejected
L   -> looped
R   -> resolved
S   -> stale
Ext -> extranet
Inv -> invalid
Stg -> staged
U   -> TLOC unresolved

                                    PATH                ATTRIBUTE
VPN    PREFIX           FROM PEER   ID    LABEL  STATUS TYPE       TLOC IP     COLOR           ENCAP  PREFERENCE
-------------------------------------------------------------------------------------------------------------------
1      0.0.0.0/0        10.10.1.11  66    1001   C,R    installed  10.10.1.11  mpls            ipsec  -
                        10.10.1.11  69    1001   C,R    installed  10.10.1.11  public-internet ipsec  -
1      10.10.20.0/24    10.10.1.11  66    1001   C,R    installed  10.10.1.11  mpls            ipsec  -
                        10.10.1.11  69    1001   C,R    installed  10.10.1.11  public-internet ipsec  -
1      10.10.21.0/24    10.10.1.13  66    1001   C,R    installed  10.10.1.13  mpls            ipsec  -
                        10.10.1.13  69    1001   C,R    installed  10.10.1.13  public-internet ipsec  -
1      10.10.22.0/24    10.10.1.15  66    1001   C,R    installed  10.10.1.15  mpls            ipsec  -
                        10.10.1.15  69    1001   C,R    installed  10.10.1.15  public-internet ipsec  -
1      10.10.24.0/24    10.10.1.17  66    1003   C,R    installed  10.10.1.17  mpls            ipsec  -
                        10.10.1.17  69    1003   C,R    installed  10.10.1.17  public-internet ipsec  -

vsmart#
```

Figure 79 - OMP Routes on the vSmart

The Hub device can be accessed in 10.10.20.172, using the command “show ip interface brief”, as shown in Figure 80, shows that the device has two Tunnels (Tunnel2 and Tunnel4), used for redundancy and fault tolerance.

```
dc-cedge01#sh ip int br
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       10.10.20.172    YES other  up                    up
GigabitEthernet2       10.10.23.6      YES other  up                    up
GigabitEthernet3       10.10.20.182    YES other  up                    up
GigabitEthernet4       10.10.23.38     YES other  up                    up
Tunnel2                10.10.23.6      YES TFTP   up                    up
Tunnel4                10.10.23.38     YES TFTP   up                    up
```

Figure 80 - Hub Interfaces

Figure 81 shows the interfaces for vEdge device 3. Interfaces ge0/0 and ge0/2 are Tunnel interfaces. Seeing the address 10.10.23.18, the address of one of the tunnel interfaces, we can now use this address to check the SA between this Spoke and the Hub.

```
site3-vedge01# show interface

                                IF      IF      IF
           AF                   ADMIN   OPER    TRACKER  ENCAP
 INTERFACE  TYPE  IP ADDRESS    STATUS  STATUS  STATUS   TYPE   PORT TYPE  MTU   HWADDR
-------------------------------------------------------------------------------------------
 ge0/0      ipv4  10.10.23.18/30   Up      Up      NA       null   transport  1500  52:54:00:00:a3:af
 ge0/2      ipv4  10.10.23.50/30   Up      Up      NA       null   transport  1500  52:54:00:0d:30:7c
 system     ipv4  10.10.1.17/32    Up      Up      NA       null   loopback   1500  00:00:00:00:00:00
 ge0/1      ipv4  10.10.24.1/24    Up      Up      NA       null   service    1500  52:54:00:0a:0d:e7
 eth0       ipv4  10.10.20.178/24  Up      Up      NA       null   service    1500  52:54:00:0a:f6:04
```

Figure 81 - vEdge3 interfaces

The SA per peer can be viewed using the command “sh crypto ipsec sa peer <ip address>”. Figure 82 shows the SAs between the Hub and vEdge 3. There is a ESP SA for Tunnel2 with SPI 0x102.

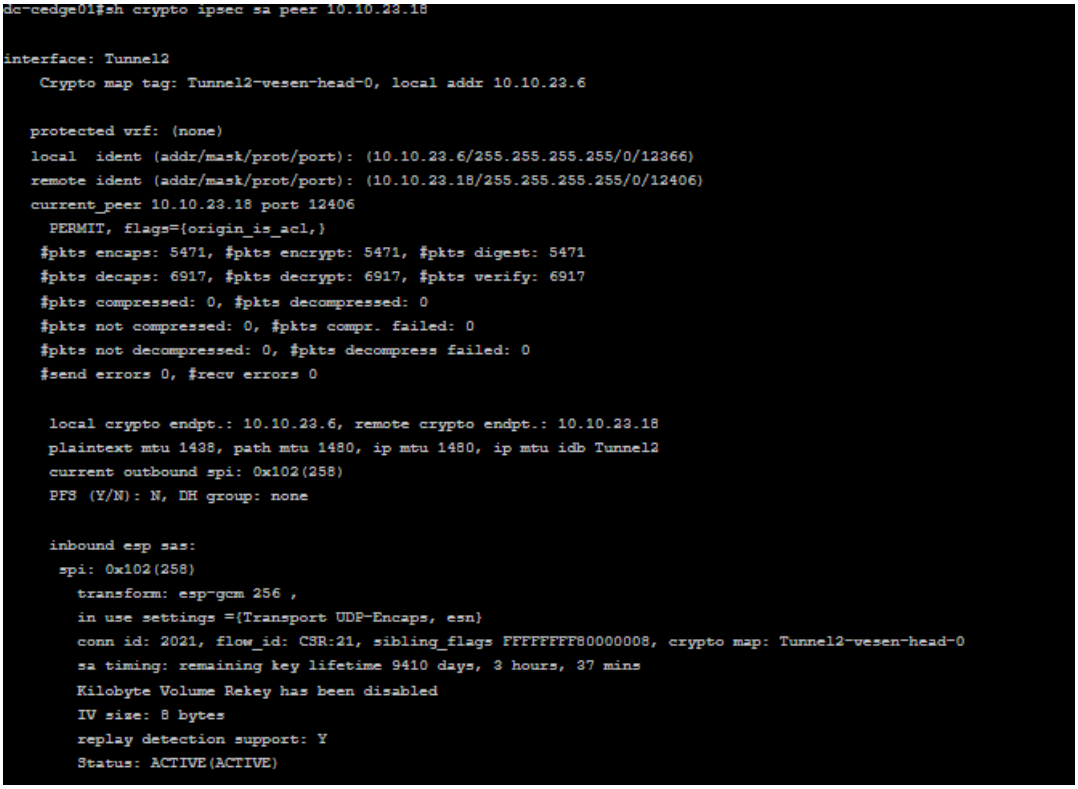


Figure 82 - SA between Hub and vEdge3

The Figure 83 below shows the SA used by Tunnel4, with the same SPI 0x102, and same remote identity 10.10.23.18. This means that each device has two connections to every other device in the VPN, this offers redundancy between VPN elements. The addresses 10.10.23.6 and 10.10.23.38 belong to the Hub, while 10.10.23.18 belongs to vEdge3. Because each vEdge has two tunnel interfaces, each one will have an established connection with the 10.10.23.18 interface on vEdge3.

```
interface: Tunnel4
    Crypto map tag: Tunnel4-vesen-head-0, local addr 10.10.23.38

  protected vrf: (none)
  local  ident (addr/mask/prot/port): (10.10.23.38/255.255.255.255/0/12366)
  remote ident (addr/mask/prot/port): (10.10.23.18/255.255.255.255/0/12406)
  current_peer 10.10.23.18 port 12406
    PERMIT, flags={origin_is_acl,}
   #pkts decaps: 5414, #pkts decrypt: 5414, #pkts verify: 5414
   #pkts compressed: 0, #pkts decompressed: 0
   #pkts not compressed: 0, #pkts compr. failed: 0
   #pkts not decompressed: 0, #pkts decompress failed: 0
   #send errors 0, #recv errors 0

    local crypto endpt.: 10.10.23.38, remote crypto endpt.: 10.10.23.18
    plaintext mtu 1438, path mtu 1480, ip mtu 1480, ip mtu idb Tunnel4
    current outbound spi: 0x102(258)
    PFS (Y/N): N, DH group: none

    inbound esp sas:
     spi: 0x100(256)
       transform: esp-gcm 256 ,
       in use settings ={Transport UDP-Encaps, esn}
       conn id: 2029, flow_id: CSR:29, sibling_flags FFFFFFFF80000008, crypto map: Tunnel4-vesen-head-0
       sa timing: remaining key lifetime 9410 days, 3 hours, 37 mins
       Kilobyte Volume Rekey has been disabled
       IV size: 8 bytes
       replay detection support: Y
       Status: ACTIVE(ACTIVE)
```

Figure 83 - Second SA between Hub and vEdge3

ACCESS CONTROL

Deploying ACLs in Cisco SEN is very easy. Creating a ACL is a matter of filling the Name, and adding a rule, where the conditions for matching are specified, and defining an action for matching packets (Drop or Forward). Afterwards, the policy needs to be applied to all devices that will use it.

The Figure 84 below shows an ACL where ICMP echo packets sent from the network 10.10.23.0/24 are dropped.

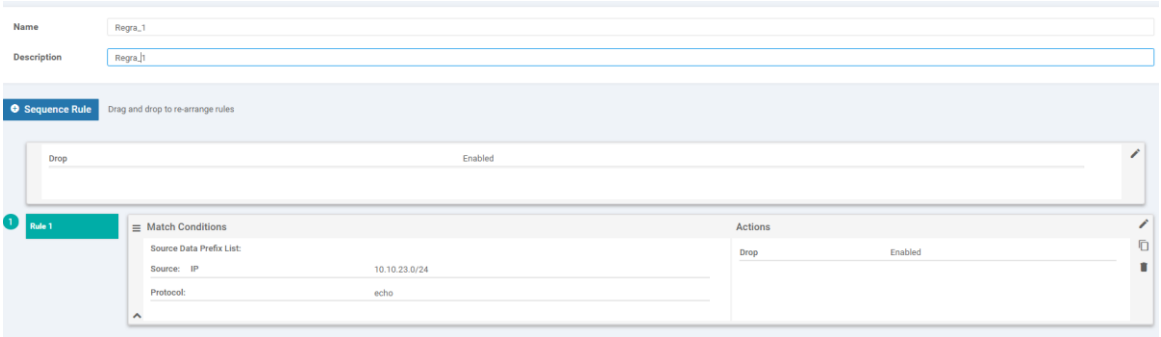


Figure 84 - ACL configuration on the vManage

After the device is updated successfully, the ACL is deployed, as shown in Figure 85.

Figure 85 - vEdge Update on the vManage

10.3 FlexiWAN

The objective of this experiment is to create a VPN between two sites using the Flexiwan SDWAN solution.

For this purpose the FlexiManage portal was used provided by Flexiwan (https://manage.flexiwan.com/home). It allows the use of up to 3 Edge devices for free. Seeing as it is an opensource solution, its possible for someone to setup their own FlexiManage to allow more devices. Two Ubuntu VMs were also used to act as Edge devices. The topology used is a simple site to site, shown below in Figure 86. The FlexiManage element of the topology was provided by FlexiWan and was in their own servers. The two flexiEdges are two ubuntu clients. Every element of this topology was separated by the internet.

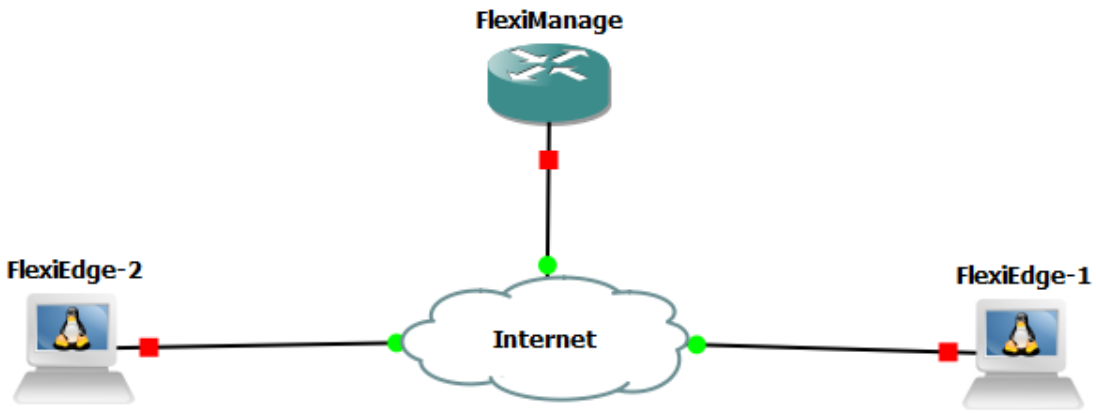


Figure 86 - Flexiwan topology

First, the following commands are used to install the flexiwan router software on the Edge Devices:

```
curl -sL https://deb.flexiwan.com/setup | bash -
apt-get install flexiwan-router
```

Afterwards, a token needs to be created in the FlexiManage portal. This token needs to be present in all the devices using the SDWAN. The token used is shown in Figure 87.

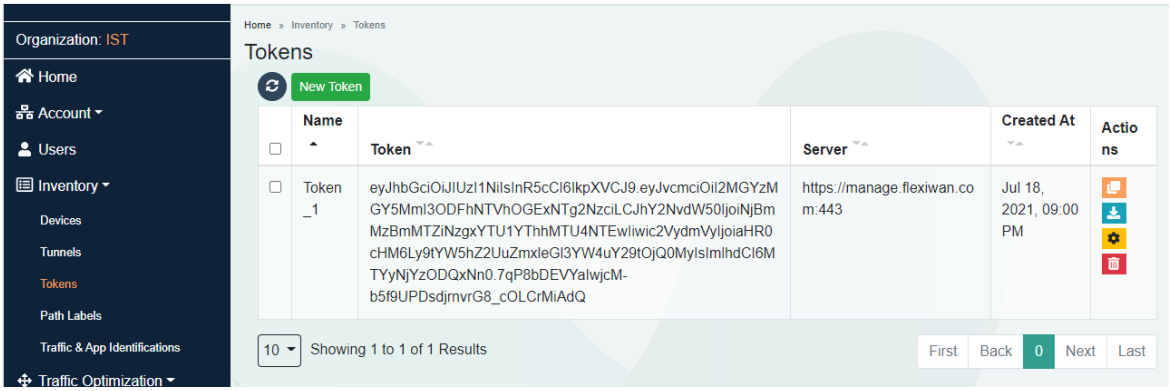


Figure 87 - Flexiwan Token

After copying the token to /etc/flexiwan/agent/ on the Ubuntu VM, the device shows up in the FlexiManage as an Unknown device. As show in Figure 88, the hostname is rsb2-VirtualBox.

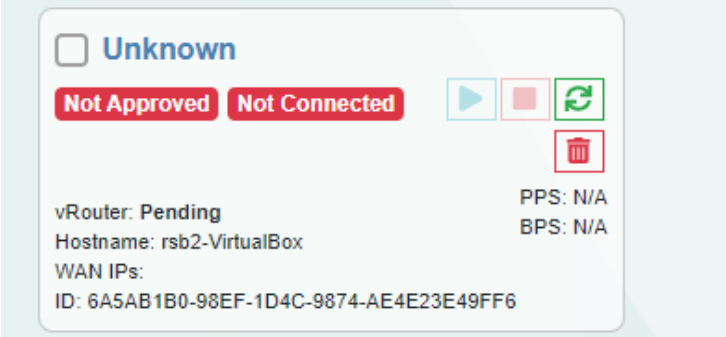


Figure 88 - Unknown device

In the device configuration, we can see several tabs. More importantly, the General and Interfaces tabs, which need to be configured for the device to be usable. First, the device needs to be approved manually by sliding the Approved slider in the General tab. This is show in Figure 89

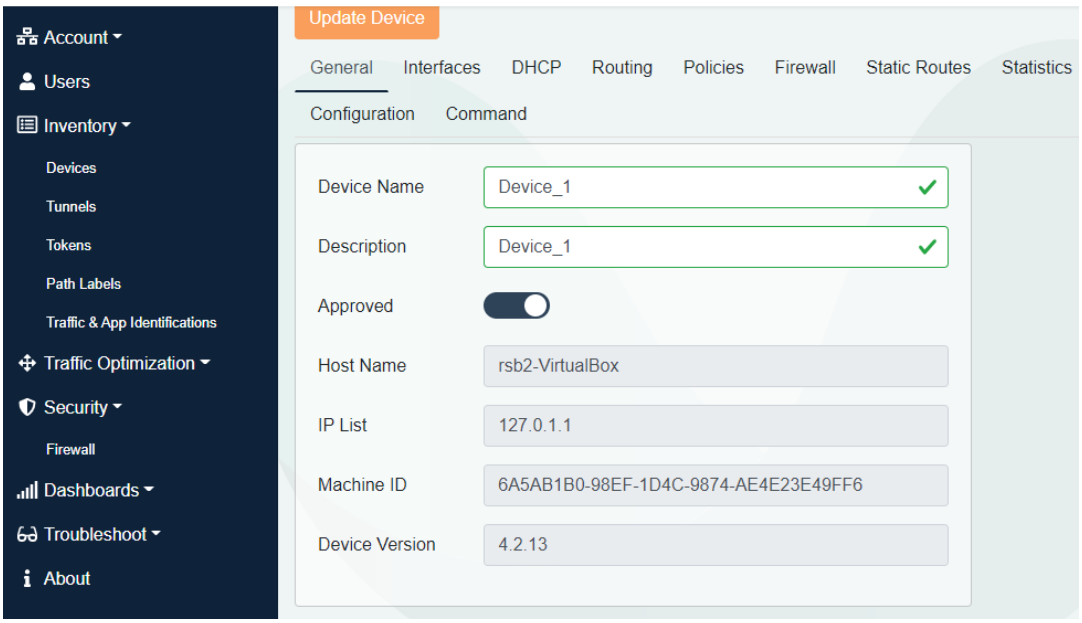


Figure 89 - Device configuration

In the Interfaces tab, there needs to be an assignment of which interface is to the LAN, and which is to the WAN, this is shown for both devices in Figure 90.

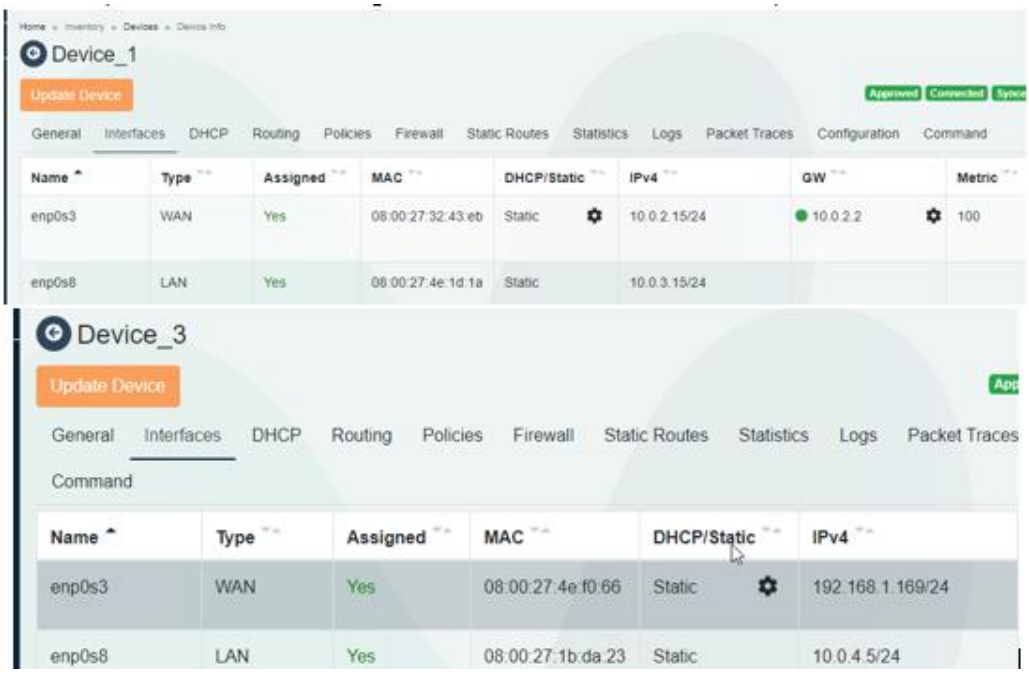


Figure 90 - Devices Interface configuration

Before establishing a Tunnel between the two devices, the key exchange method must be defined. It can be PSK, IKEv2 or no encryption. No encryption was used for it to be possible to perform packet

analysis, but when encryption was used, IKEv2 was selected as the Exchange Method. These methods are shown in Figure 91.

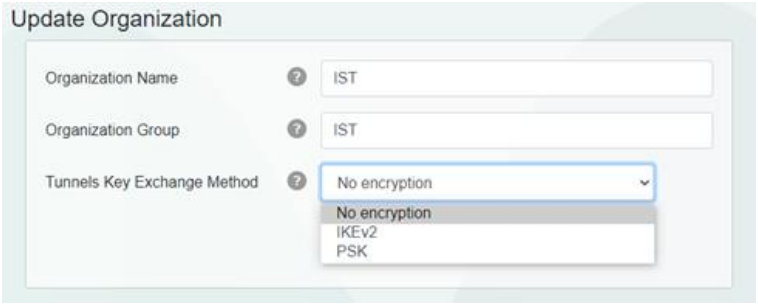


Figure 91 - Flexiwan Key Exchange Methods

In order to create the tunnels, the devices need to be selected, and the create tunnels option must be used as shown in Figure 92.

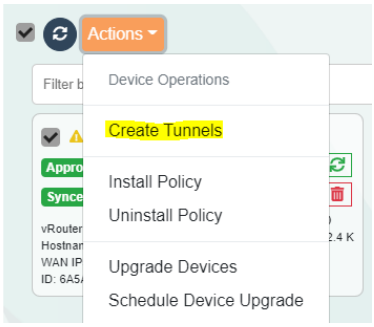


Figure 92 - Flexiwan Tunnel Creation

After this step, the tunnel is created between two loopback interfaces (each in one device), created specifically for the tunnel establishment. The connected tunnel is shown in Figure 93.

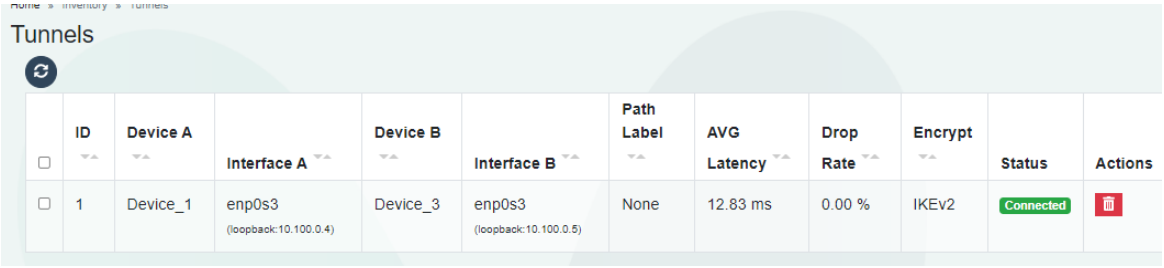


Figure 93 - Flexiwan Tunnel

The device interface configurations are the following.

Device_1:

- Hostname “rsb-VirtualBox”
- Int enp0s3 - 10.0.2.15

- Int enp0s8 - 10.0.3.15

Device_3:

- Hostname “rsb2-VirtualBox” enp0s3:
- Int enp0s3 - 192.168.1.169
- Int enp0s8 - 10.0.4.5

Afterwards, connectivity is achieved between the two sites, as shown by the ping in figure 94.

```
rsb@rsb-VirtualBox:~$ ping 10.0.3.15
PING 10.0.3.15 (10.0.3.15) 56(84) bytes of data.
64 bytes from 10.0.3.15: icmp_seq=1 ttl=64 time=13.7 ms
64 bytes from 10.0.3.15: icmp_seq=2 ttl=64 time=10.4 ms
64 bytes from 10.0.3.15: icmp_seq=3 ttl=64 time=5.30 ms
64 bytes from 10.0.3.15: icmp_seq=4 ttl=64 time=12.3 ms
64 bytes from 10.0.3.15: icmp_seq=5 ttl=64 time=10.3 ms
64 bytes from 10.0.3.15: icmp_seq=6 ttl=64 time=10.5 ms
64 bytes from 10.0.3.15: icmp_seq=7 ttl=64 time=12.3 ms
^C
--- 10.0.3.15 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6007ms
rtt min/avg/max/mdev = 5.300/10.744/13.780/2.515 ms
```

Figure 94 - Ping between both devices

FlexiWan uses Vector Packet Processing, which means that Wireshark captures show the packet for each node that it passed through. This allows for the viewing of real time packet construction. The packets show the VPP Dispatch Trace, NodeName and Buffer information. Figure 95 shows the ICMP packet when before it was processed by the vxlan4-encap node. Before it was processed, it showed only the ICMP payload, IPv4 header, and L2 header, along with the VPP information.

```
  82 0.989… 10.100.0.5 10.0.3.15   ICMP   3426 Echo (ping) request  id=0x139c, seq=4/1024, ttl=64
  83 0.989… 10.100.0.5 10.0.3.15   ICMP   3485 Echo (ping) request  id=0x139c, seq=4/1024, ttl=64
  84 0.989… 10.100.0.5 10.0.3.15   ICMP   3528 Echo (ping) request  id=0x139c, seq=4/1024, ttl=64
  85 0.989… 10.100.0.5 10.0.3.15   ICMP   3533 Echo (ping) request  id=0x139c, seq=4/1024, ttl=64
<
> Frame 82: 3426 bytes on wire (27408 bits), 3426 bytes captured (27408 bits)
> VPP Dispatch Trace
  NodeName: vxlan4-encap
> VPP Buffer Metadata
> VPP Buffer Opaque
> VPP Buffer Opaque2
> Ethernet II, Src: 02:00:27:fd:00:05 (02:00:27:fd:00:05), Dst: 02:00:27:fd:00:04 (02:00:27:fd:00:0
> Internet Protocol Version 4, Src: 10.100.0.5, Dst: 10.0.3.15
> Internet Control Message Protocol
```

Figure 95 - ICMP Packet before VXLAN node

Figure 96 shows the packet after it was processed, now having an extra IPv4 and UDP headers, and VXLAN information. We can see that the VNI (VXLAN Network ID) is 2.

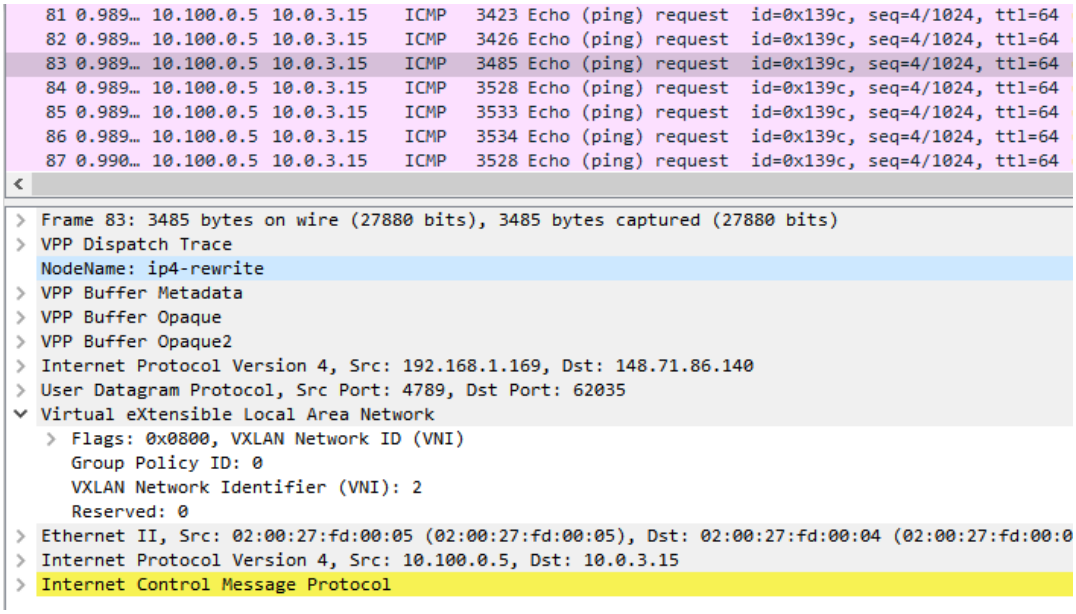


Figure 96 - ICMP Packet after VXLAN node processing

Figure 97 shows the packet fully constructed before being processed by the last node and sent to its destination. It shows an extra L2 header relative to Figure 96. VPP only impacts the rate at which packets are processed, meaning that it affects only the speed of processing.

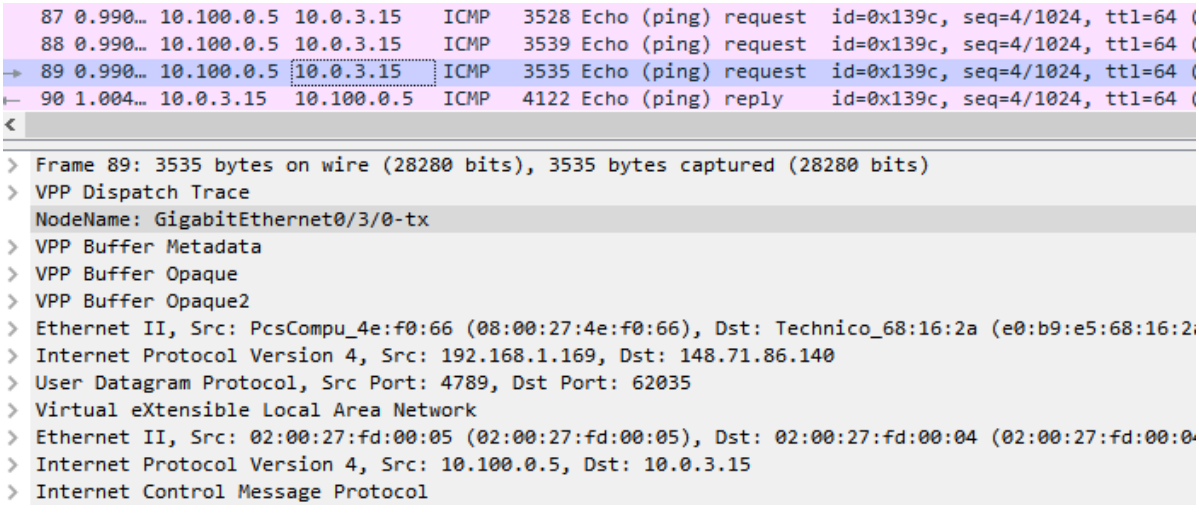


Figure 97 - ICMP packet before last node

ACCESS CONTROL

To deploy ACLs in Flexiwan a rule must be configured with the Match Conditions and the corresponding action (Deny, Allow). Afterwards, the policy needs to be applied to all devices that will use it. Figure 98 shows an ACL where all ICMP echo packets sent are dropped.

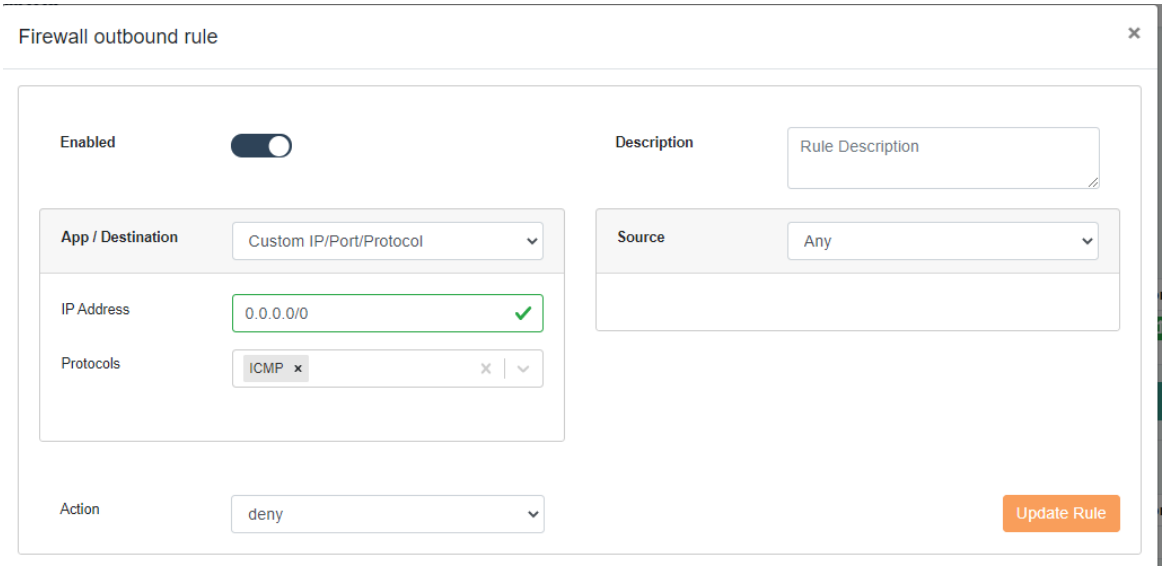


Figure 98 - Flexiwan ACL configuration

The rule is created, and the policy is saved, all that's left is to apply the ACL to all devices, as shown in Figure 99.

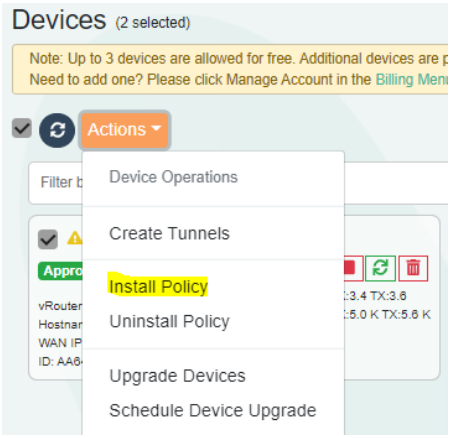


Figure 99 - Flexiwan Install Policy

11. Public Key Infrastructure (Single and Hierarchical) - Experiments

11.1 Single

The objective of this experiment is to perform key distribution using Digital Certificates between the PKI Client and Server. Figure 100 shows the topology. OSPF has been configured between the routers to establish connectivity. To build this topology the GNS3 network tool was used. One PKI CA (Server) and one PKI Client/User were set up, using Cisco IOSv. One more Cisco router (RA, C3725) was set up to represent the public network.

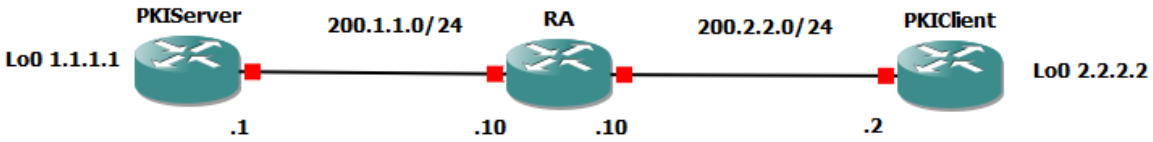


Figure 100 - Single PKI Topology

The process begins by generating an RSA Key pair in both the PKIServer and the PKIClient. Then the PKIClient sends a certificate request (enrollment), with its public RSA key, to the PKIServer (CA). The PKIServer will then sign the request with its private key and generate the user certificate, which will then be sent back to the PKIClient.

The starting CA configuration involves creating an HTTP server on the CA, and configuring the server with its issuer name (CN= Common Name, O= Organization), the granting mode used is auto (the other mode would be manual). To configure the CA, the following commands were used.

CA Configuration

```
ip http server
crypto pki server CA
issuer-name CN=CA, O=teste.com
grant auto
shutdown
no shutdown
```

The above configuration will also generate the RSA Key pair in the server. We can verify the keys through the command “show crypto key mypubkey rsa”, as shown in Figure 101. The command prints the Key name, type and data.

```
Router#sh crypto key mypubkey rsa
% Key pair was generated at: 13:24:32 UTC Aug 28 2021
Key name: CA
Key type: RSA KEYS
 Storage Device: not specified
 Usage: General Purpose Key
 Key is not exportable.
 Key Data:
  30819F30 0D06092A 864886F7 0D010101 05000381 8D003081 89028181 00D53684
  E30C9867 58018E7E B32EFAF0 C3C4276E 29124461 7A4DA765 D084626D 27D87FE7
  1DA3F3E7 AF0787A6 A5FE365D 56301AE3 E11646A1 B08502F0 FD8B600C 4BE9A7C9
  43707495 14F956C0 2FF50B2C 00C8C900 6403EF5E 071AC8AE 53903299 5C80E172
  8A8D2652 2BF812E4 94EED7BD 8E234E71 FDDCE24D EF83AED6 356BDF68 C3020301
  0001
% Key pair was generated at: 13:24:34 UTC Aug 28 2021
Key name: CA.server
Key type: RSA KEYS
Temporary key
 Usage: Encryption Key
 Key is not exportable.
 Key Data:
  307C300D 06092A86 4886F70D 01010105 00036B00 30680261 00A25CAA BA46FF8D
  D4134665 294B7E02 9CF13AD3 55B12386 6357739E F86C114B 1448952F E2474F47
  5C2D0C87 68DEB888 40BD08BB 0B1CB8BB 1970C40A 1622E5AD 8CA06745 DD7F63AF
  7A0CB01C DCECBD1B DED62705 D720DBF5 F46445FF 283BEBF2 A9020301 0001
Router#
```

Figure 101 - PKI CA Keys on the PKIServer

Using the command “sh crypto pki server” on the PKIServer, the certificate server configuration can be viewed. The command is shown in Figure 102. It includes the server status, issuer name (in this case Common Name=CA and Organization=teste.com), the certificate fingerprint, the granting mode (auto means that all requests will be granted; there is also the option to grant manually) and the CA certificate expiration timer.

```
Router#sh crypto pki server
Certificate Server CA:
    Status: enabled
    State: enabled
    Server's configuration is locked  (enter "shut" to unlock it)
    Issuer name: CN=CA, O=teste.com
    CA cert fingerprint: F8AE7790 E85A01C5 206C59CB A0526846
    Granting mode is: auto
    Last certificate issued serial number (hex): 1
    CA certificate expiration timer: 13:24:33 UTC Aug 27 2024
    CRL NextUpdate timer: 19:24:33 UTC Aug 28 2021
    Current primary storage dir: nvram:
    Database Level: Minimum - no cert data written to storage
Router#
```

Figure 102 - PKI Server Configuration

After the certificate server is online, the client also needs to be configured. This means generating the RSA key pair, setting the enrollment url, and creating a trustpoint on the router. This is done with the following commands:

Client Configuration

crypto key generate rsa modulus 2048 label c1.teste.com

crypto pki trustpoint Trusted-CA

!Enrollment ur is http://<CA_Loopback_Address>

enrollment url http://1.1.1.1

rsakeypair c1.teste.com

!Set FQDN which acts as the Subject Hostame on the certificate

fqdn c1.teste.com

subject-name CN=c1, O=teste.com

revocation-check none

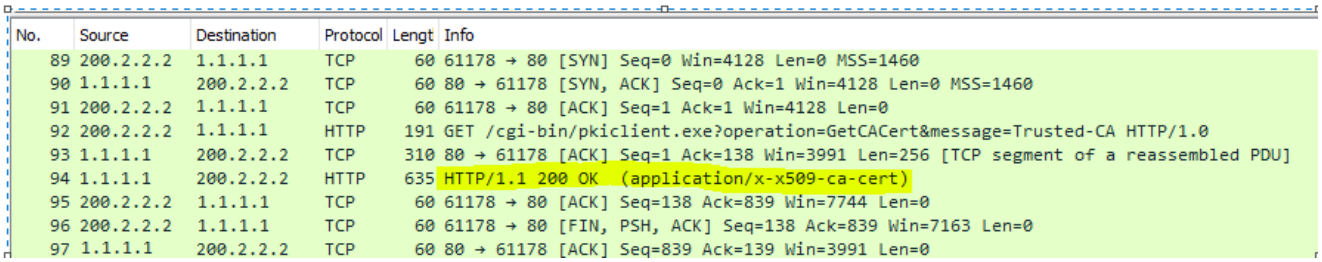
The client then needs to authenticate the CA manually, to obtain its certificate and mark it as a trusted entity. Figure 103 shows this, and the trustpoint configuration.

```
Router(config)#crypto pki trustpoint Trusted-CA
Router(ca-trustpoint)#enrollment url http://1.1.1.1
Router(ca-trustpoint)#rsakeypair c1.teste.com
Router(ca-trustpoint)#fqdn c1.teste.com
Router(ca-trustpoint)#subject-name CN=c1, O=teste.com
Router(ca-trustpoint)#revocation-check none
Router(ca-trustpoint)#exit
Router(config)#crypto pki authenticate Trusted-CA
Certificate has the following attributes:
       Fingerprint MD5: F8AE7790 E85A01C5 206C59CB A0526846
      Fingerprint SHA1: C046F6A1 0AD43977 DD6A9F18 A0F4E1FA 1244422B

% Do you accept this certificate? [yes/no]: yes
Trustpoint CA certificate accepted.
Router(config)#
```

Figure 103 – Authentication of the CA on the PKIClient

With a Wireshark capture it's possible to see the SCEP exchange between the PKIClient and PKIServer (CA). Figure 104 shows the first part of the exchange, which is the authentication. Packet 92 is the user (200.2.2.2) request to the CA (1.1.1.1); this request is done using a GetCACertMessage HTTP GET request. Packet 94 is the CA response with its own certificate; this is done using a 200 OK HTTP message.



No.	Source	Destination	Protocol	Lengt	Info
89	200.2.2.2	1.1.1.1	TCP	60	61178 → 80 [SYN] Seq=0 Win=4128 Len=0 MSS=1460
90	1.1.1.1	200.2.2.2	TCP	60	80 → 61178 [SYN, ACK] Seq=0 Ack=1 Win=4128 Len=0 MSS=1460
91	200.2.2.2	1.1.1.1	TCP	60	61178 → 80 [ACK] Seq=1 Ack=1 Win=4128 Len=0
92	200.2.2.2	1.1.1.1	HTTP	191	GET /cgi-bin/pkiclient.exe?operation=GetCACert&message=Trusted-CA HTTP/1.0
93	1.1.1.1	200.2.2.2	TCP	310	80 → 61178 [ACK] Seq=1 Ack=138 Win=3991 Len=256 [TCP segment of a reassembled PDU]
94	1.1.1.1	200.2.2.2	HTTP	635	HTTP/1.1 200 OK (application/x-x509-ca-cert)
95	200.2.2.2	1.1.1.1	TCP	60	61178 → 80 [ACK] Seq=138 Ack=839 Win=7744 Len=0
96	200.2.2.2	1.1.1.1	TCP	60	61178 → 80 [FIN, PSH, ACK] Seq=138 Ack=839 Win=7163 Len=0
97	1.1.1.1	200.2.2.2	TCP	60	80 → 61178 [ACK] Seq=839 Ack=139 Win=3991 Len=0

Figure 104 - SCEP 1st part Wireshark capture

Figure 105 shows the HTTP payload of packet 92.

```
Hypertext Transfer Protocol
  GET /cgi-bin/pkiclient.exe?operation=GetCACert&message=Trusted-CA HTTP/1.0\r\n
  User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Cisco PKI)\r\n
  \r\n
  [HTTP request 1/1]
  [Response in frame: 94]
```

Figure 105 - HTTP payload of SCEP GetCACert message

Figure 106 shows the HTTP payload of packet 94. The content type is application/x-x509-ca-cert which means that the Media Type carries a x509 certificate. The bytes from the Media Type can be extracted to a file, with extension .crt, to get the certificate.

```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
  Date: Thu, 07 Oct 2021 20:53:50 GMT\r\n
  Server: cisco-IOS\r\n
  Content-Type: application/x-x509-ca-cert\r\n
  Expires: Thu, 07 Oct 2021 20:53:50 GMT\r\n
  Last-Modified: Thu, 07 Oct 2021 20:53:50 GMT\r\n
  Cache-Control: no-store, no-cache, must-revalidate\r\n
  Pragma: no-cache\r\n
  Accept-Ranges: none\r\n
  \r\n
  [HTTP response 1/1]
  [Time since request: 0.024096000 seconds]
  [Request in frame: 92]
  [Request URI: /cgi-bin/pkiclient.exe?operation=GetCACert&message=Trusted-CA]
  File Data: 543 bytes
Media Type
  Media type: application/x-x509-ca-cert (543 bytes)
```

Figure 106 - HTTP payload of SCEP OK mesage

Figure 107 shows the certificate file when opened in windows. The issuer is CA, from domain teste.com, which is the CA of this experiment.

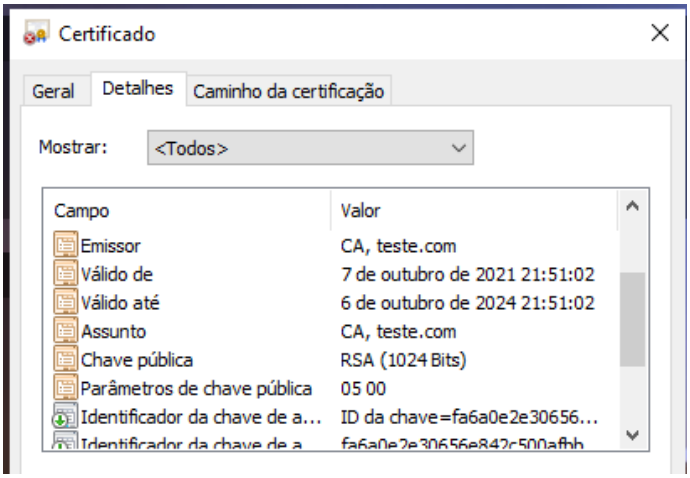


Figure 107 - CA certificate extracted from SCEP exchange

After the client authenticates the server, it can request a certificate for itself with the command "crypto pki enroll Trusted-CA". Figure 108 shows this command and also shows the confirmation that the request was sent, and that the certificate was received.

```
Router(config)#crypto pki enroll Trusted-CA
%
% Start certificate enrollment ..
% Create a challenge password. You will need to verbally provide this
   password to the CA Administrator in order to revoke your certificate.
   For security reasons your password will not be saved in the configuration.
   Please make a note of it.

Password:
Re-enter password:

% The subject name in the certificate will include: CN=c1, O=teste.com
% The subject name in the certificate will include: c1.teste.com
% Include the router serial number in the subject name? [yes/no]: n
% Include an IP address in the subject name? [no]: n
Request certificate from CA? [yes/no]: y
% Certificate request sent to Certificate Authority
% The 'show crypto pki certificate verbose Trusted-CA' commandwill show the fingerprint.

Router(config)#
*Aug 28 13:30:51.503: CRYPTO_PKI:  Certificate Request Fingerprint MD5: C4AD66D7 8E3D372B B121BDBB 646824A3
*Aug 28 13:30:51.510: CRYPTO_PKI:  Certificate Request Fingerprint SHA1: 46B20F60 0390294F 6FC7FB9D DD0134E7 DB06CDAE
Router(config)#
*Aug 28 13:30:53.827: %PKI-6-CERTRET: Certificate received from Certificate Authority
Router(config)#
```

Figure 108 - Client Certificate Request

The second part of the SCEP exchange can also be captured using Wireshark. As shown in Figure 109, the certificate request is done by the client by sending an HTTP GET PKIOperation message to the CA. However, the client certificate can't be extracted from the 200 OK response because it is encrypted with the user's public key, and requires the private key to decrypt.

Source	Destination	Protocol	Lengt	Info
200.2.2.2	1.1.1.1	HTTP	157	GET /cgi-bin/pkiclient.exe?operation=PKIOperation&message=MIIIqgYJKoZI
1.1.1.1	200.2.2.2	TCP	310	80 → 44467 [ACK] Seq=1 Ack=3320 Win=4025 Len=256 [TCP segment of a rea
200.2.2.10	224.0.0.5	OSPF	94	Hello Packet
1.1.1.1	200.2.2.2	TCP	335	HTTP/1.1 200 OK [TCP segment of a reassembled PDU]

Figure 109 - SCEP 2nd part Wireshark capture

The command “show crypto pki certificate verbose Trusted-CA” shows all the certificates that a device has saved. Figure 110 shows the command when run at the client. It prints its own certificate first, with issuer information, subject information, validity dates, key information, signature algorithm and fingerprints. Figure 111 shows the rest of the print, which is the same information but regarding the CA Certificate.

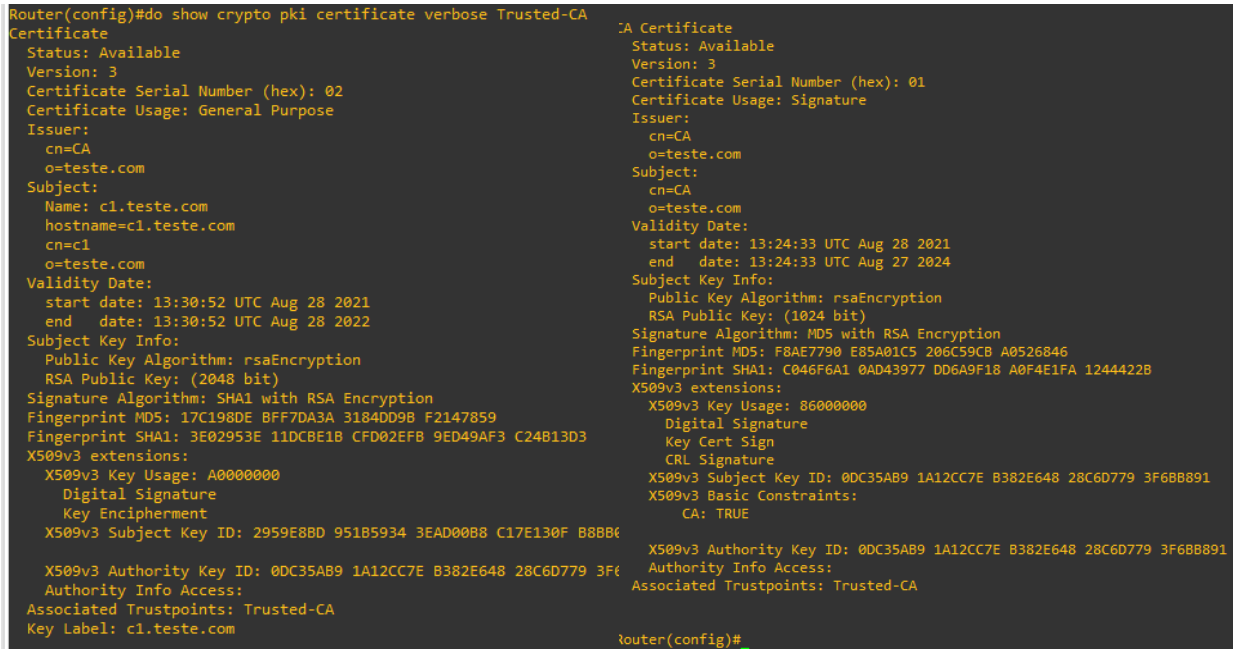


Figure 110 - Client Certificate (on the left) and CA Certificate (on the right) shown in PKIClient

The client then has a certificate, containing its public key, that can be used for establishing secure communications.

11.2 Hierarchical

The objective of this experiment is to perform key distribution using Digital Certificates in a two-level PKI. The PKI_SUB_CA will act as a subordinate CA, and get its certificate from ROOT_CA. The PKIClient will get its certificate from the SubCA. Figure 111 shows the topology. To build this topology the GNS3 network tool was used, one PKI ROOT CA (Server), one PKI SubCA, and one PKI Client/User were set up, using Cisco IOSv, one more Cisco router (R1, C3725) was set up, it represents the public network.

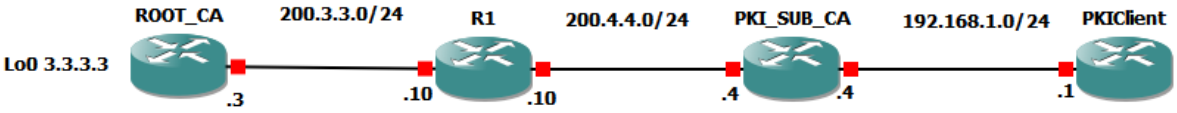


Figure 111 - Hierarchical PKI Topology

This process begins by generating an RSA Key pair in all routers except R1. The configurations are very similar to 3.5.1, but after configuring the CA, we need to configure a secondary CA, and when configuring the client, the configured CA will be the SubCA and not the Root CA. After this we configure the Certificate Server in the CA like in chapter 11.1. Figure 112 shows the certificate server.

```
Router#sh crypto pki server
*Aug 28 14:22:10.341: %SYS-5-CONFIG_I: Configured from console by console
Router#sh crypto pki server
Certificate Server CA:
    Status: enabled
    State: enabled
    Server's configuration is locked  (enter "shut" to unlock it)
    Issuer name: CN=CA, O=teste.com
    CA cert fingerprint: 35F4D119 81F570C0 63150131 C012682E
    Granting mode is: auto
    Last certificate issued serial number (hex): 1
    CA certificate expiration timer: 14:21:14 UTC Aug 27 2024
    CRL NextUpdate timer: 20:21:14 UTC Aug 28 2021
    Current primary storage dir: nvram:
    Database Level: Minimum - no cert data written to storage
Router#
```

Figure 112 - PKI Server Configuration

After generating the Keys in the SubCA, it authenticates the root CA and enrolls as an issuing subordinate SA (the trustpoint configuration is the same as the CA configuration in 3.5.1, but with the added command “mode sub-cs”).

Unlike a normal certificate request, the SubCA enrollment must be manually approved. The requests can be viewed with the command “show crypto pki server CA requests”, and can be accepted with “crypto pki server CA grant <id>”. Figure 113 shows both these commands. First we see the request no. 1 in a pending state (Common Name = SubCA1, Organization=teste.com). Then, after the request is accepted, it is shown as granted.

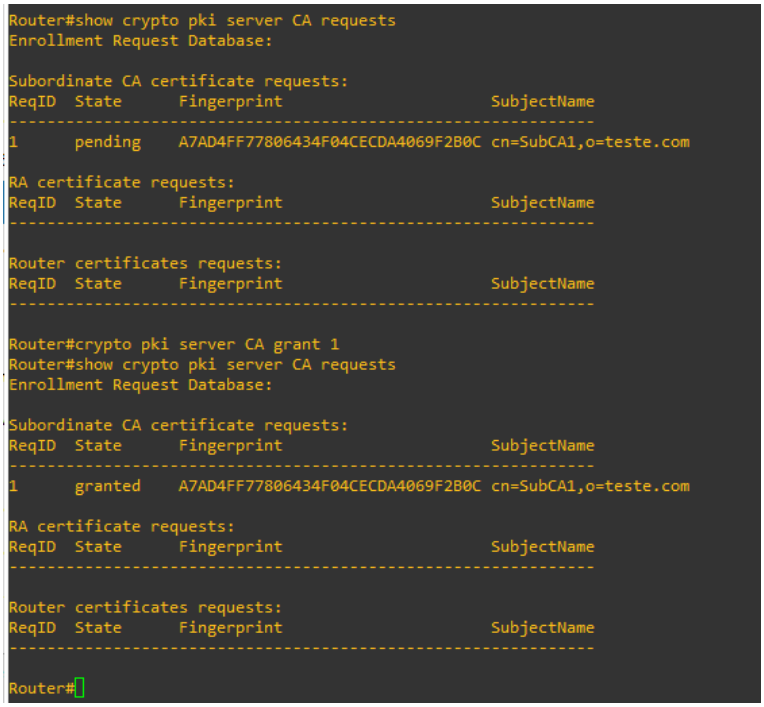


Figure 113 - Certificate request on CA

Figure 114 shows the certificate being received from the CA, and the SubCA server being enabled.

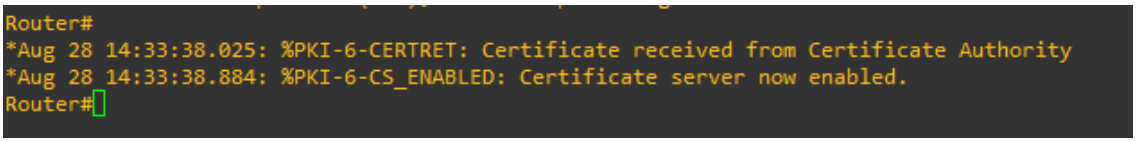


Figure 114 - Certificate reception on SubCA

Figure 115 shows the server configuration in the SubCA, where it says that the server is configured in subordinate server mode (there are only two modes: CA and SubCA).

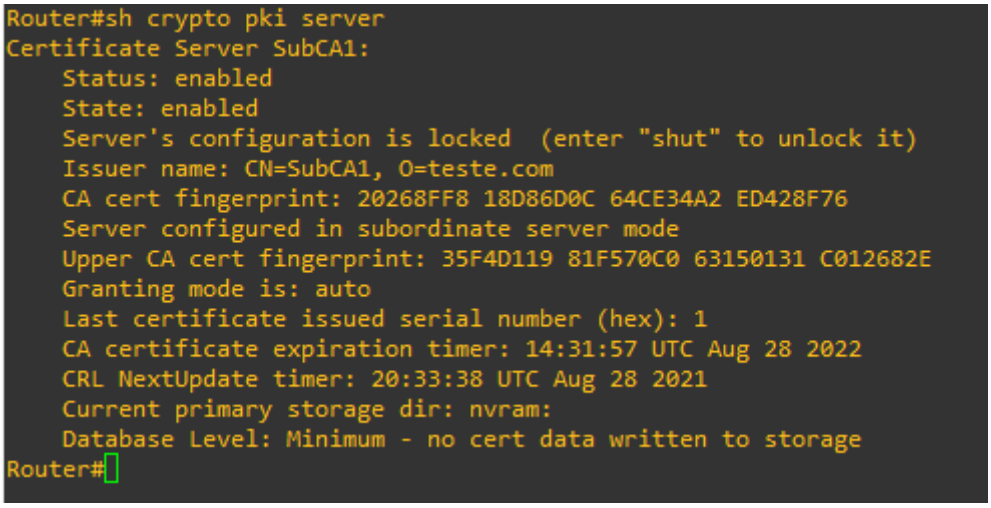


Figure 115 - SubCA Server Configuration

Figure 116 shows the certificates on the SubCA. First is its own certificate with the indication that it belongs to a subordinate CA, and after it is the CA certificate.

```
Router#sh crypto pki certificates
Certificate (subordinate CA certificate)
  Status: Available
  Certificate Serial Number (hex): 02
  Certificate Usage: Signature
  Issuer:
    cn=CA
    o=teste.com
  Subject:
    cn=SubCA1
    o=teste.com
  Validity Date:
    start date: 14:31:57 UTC Aug 28 2021
    end   date: 14:31:57 UTC Aug 28 2022
  Associated Trustpoints: SubCA1

CA Certificate
  Status: Available
  Certificate Serial Number (hex): 01
  Certificate Usage: Signature
  Issuer:
    cn=CA
    o=teste.com
  Subject:
    cn=CA
    o=teste.com
  Validity Date:
    start date: 14:21:14 UTC Aug 28 2021
    end   date: 14:21:14 UTC Aug 27 2024
  Associated Trustpoints: SubCA1
```

Figure 116 - SubCA Installed Certificates

In the client, after authenticating the root CA, there's no need to manually authenticate the SubCA, which has a certificate signed by the root CA. However, if there is a manual authentication the certificate is automatically accepted, with the indication that it is signed by an existing trustpoint CA certificate, as shown in Figure 117. This happens because there is a chain of authentication, where the user trusts the root CA, and will trust the SubCA because its certificate was signed by the trusted root CA. When there are more than two levels, the user needs to authenticate each level above its issuing CA in order to trust it. This is called Chain of Trust.

```
Router(config)#crypto pki authenticate Trusted-SubCA
Certificate has the following attributes:
       Fingerprint MD5: 20268FF8 18D86D0C 64CE34A2 ED428F76
      Fingerprint SHA1: 3A7DF734 71877D8D F9377DAE 2D8D7591 80D0E6A6
Certificate validated - Signed by existing trustpoint CA certificate.

Trustpoint CA certificate accepted.
Router(config)#
```

Figure 117 - SubCA Manual Authentication

All that's left to do is to enroll the client in the SubCA (unlike the configuration in 3.5.1, here the enrollment url is http.//192.168.1.4) and to obtain its certificate. This is shown in figure 118.

```
Router(config)#crypto pki enroll Trusted-SubCA
%
% Start certificate enrollment ..
% Create a challenge password. You will need to verbally provide this
   password to the CA Administrator in order to revoke your certificate.
   For security reasons your password will not be saved in the configuration.
   Please make a note of it.

Password:
Re-enter password:

% The subject name in the certificate will include: Router
% Include the router serial number in the subject name? [yes/no]: n
% Include an IP address in the subject name? [no]: n
Request certificate from CA? [yes/no]: y
% Certificate request sent to Certificate Authority
% The 'show crypto pki certificate verbose Trusted-SubCA' commandwill show the fingerprint.

Router(config)#
*Aug 28 14:42:27.102: CRYPTO_PKI:  Certificate Request Fingerprint MD5: D8DC9337 F580F631 6CE74A05 0512F0B9
*Aug 28 14:42:27.109: CRYPTO_PKI:  Certificate Request Fingerprint SHA1: 05515A35 70D066BB 1129A799 2BEEB5D8 03F
Router(config)#
*Aug 28 14:42:29.324: %PKI-6-CERTRET: Certificate received from Certificate Authority
Router(config)#
```

Figure 118 - Client Enrollment and Certificate Reception

Certificate Revocation Lists (CRL)

In order to add an entry to the router's local CRL, a certificate needs to be revoked. Figure 119 shows the revocation of the SubCA's certificate, with serial number 0x2 as shown before in Figure 116.

```
Router#crypto pki server CA revoke 0x2
% Certificate 02 succesfully revoked.
```

Figure 119 - Revocation of SubCA Certifiate at the root CA

The CRL can be verified using the command "show crypto pki server <CA_name> crl", as shown in Figure 120. The issuer is the root CA, there's one CRL entry, which is the revoked certificate (0x2) and the revocation date.

```
Router#sh crypto pki server CA crl
Certificate Revocation List:
    Issuer: cn=CA,o=teste.com
    This Update: 00:15:44 UTC Oct 8 2021
    Next Update: 06:15:44 UTC Oct 8 2021
    Number of CRL entries: 1
    CRL size: 254 bytes
Revoked Certificates:
    Serial Number (hex): 02
    Revocation Date: 00:15:44 UTC Oct 8 2021

Router#
```

Figure 120 - Certificate revocation list at the CA

For the user to check the CA's CRL when authenticating a certificate, the command "revocation-check crl" needs to be added to the client configuration from chapter 11.1. Figure 121 shows what happens when the user tries to authenticate the SubCA after its certificate had been revoked, the authentication fails.

```
Router(config)#crypto pki authenticate Trusted-SubCA
Trustpoint 'Trusted-SubCA' is a subordinate CA.
Authentication failed - could not validate certificate
% Error in saving certificate: status = FAIL

Router(config)#
```

Figure 121 - SubCA authentication fails at the user

12. FlexVPN using Digital Certificate based authentication (PKI) - Experiment

The objective of this experiment is to configure a FlexVPN Site-to-Site topology using Digital Certificates as the IKEv2 authentication method to establish a VPN. BGP will also be used between the Hub and Spoke to share routes. To perform the experiment, the GNS3 network tool was used, one FlexVPN Hub, one FlexVPN Spoke and one Certificate Authority were set up, using Cisco IOSv. One more Cisco router (RA, C3725) was set up to represent the public network. Figure 122 shows the experiment topology.

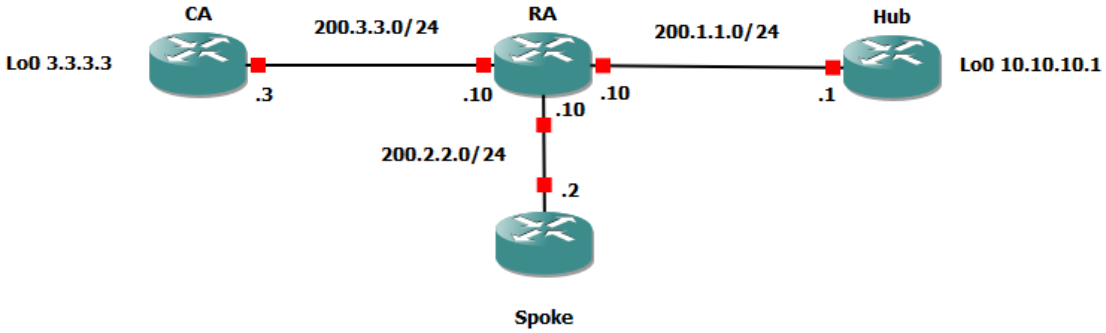


Figure 122 - FlexVPN using PKI Topology

The CA is configured as a certificate server, and its certificate is shown in figure 123.

```
Router(cs-server)#do sh crypto pki cert
CA Certificate
  Status: Available
  Certificate Serial Number (hex): 01
  Certificate Usage: Signature
  Issuer:
    cn=CA
    o=teste.com
  Subject:
    cn=CA
    o=teste.com
  Validity Date:
    start date: 16:55:38 UTC Aug 28 2021
    end   date: 16:55:38 UTC Aug 27 2024
  Associated Trustpoints: CA


Router(cs-server)#
```

Figure 123 - CA Certificate

After the Hub and Spoke both have their own signed certificates, they need to be configured to use IKEv2 to establish a VPN. The Hub configuration is as follows.

In both the Hub and Spoke, RSA keys are generated, the CA is manually authenticated, and the certificates are received by both devices.

After the certificate distribution is completed, the FlexVPN configuration must also be done. This starts by creating a certificate MAP to match all certificates containing the Organization set up in the experiment. The following configuration shows this.

```
! Create Certificate map to match all certificates containing "teste.com" (used in this experiment)
crypto pki certificate map CERT_MAP 10
subject-name co teste.com
exit
```

After the certificate map is configured, we need to configure AAA and set an authorization policy in order to use Dynamic IP addresses, for this we need to create a local IP pool to set the address range.

```
! Activate AAA to use authorization policy with an IP pool
! So its possible to assign dynamic IP addresses (only necessary if using dynamic IPs)
aaa new-model
aaa authorization network FLEXVPN_LOCAL local
crypto ikev2 authorization policy IKEV2_AUTHORIZATION
pool FLEXVPN_POOL
route set interface
exit
! Set local ip pool to use IP range 10.10.10.2-200
ip local pool FLEXVPN_POOL 10.10.10.2 10.10.10.200
```

The VPN configuration continues with the set up of a virtual-template interface, using the Hub loopback address (this will be the connection IP for the Spokes). Then, the IKEv2/IPSec configuration needs to be done. We need to configure the IKEv2 proposal, then an IKEv2 Policy using the configured proposal, an IKEv2 Profile needs to be configured where we include the Certificate Map, local identity, authentication method and PKI trustpoint to be used with the profile.

```
! Configure virtual-template with loopback IP address (10.10.10.1)
int virtual-template 1 type tunnel
ip unnumbered loopback 0
exit
```

```
! Configure IKEv2 proposal with the encryption/integrity algorithms to send in the proposal
crypto ikev2 proposal PROP
encryption aes-cbc-256
integrity sha256
group 2
exit
! Configure IKEv2 policy to use the previously created IKEv2 Proposal
crypto ikev2 policy 10
proposal PROP
exit
! Configure IKEv2 profile and set authentication to use certificates
crypto ikev2 profile PROFILE
! Match certificates in CERT_MAP (containing teste.com)
match certificate CERT_MAP
identity local dn
!Authentication must be rsa-sig
authentication local rsa-sig
authentication remote rsa-sig
!Set PKI trustpoint to use in this profile
pki trustpoint Trusted-CA
aaa authorization group cert list FLEXVPN_LOCAL IKEV2_AUTHORIZATION
virtual-template 1
exit
```

The IPSec configuration involves a transform-set, where the IPSec algorithms are set, an IPSec profile, where the transform set and the IKEv2 profile are included. The virtual-template is then configured to use interface g0/0 and the IPSec profile.

```
! Configure IPSect transform-set with the IPSec algorithms to be used
crypto ipsec transform-set TranSet esp-sha-hmac
mode tunnel
exit
! Configure IPSec profile
crypto ipsec profile IpsecPROFILE
set transform-set TranSet
set ikev2-profile PROFILE
exit
! Set virtual-template to use interface g0/0 and use the configured IPSec profile
int virtual-template 1 type tunnel
```

```
tunnel source g0/0
tunnel mode ipsec ipv4
tunnel protection ipsec profile IpsecPROFILE
exit
```

BGP is then used for the HUB to create BGP connections with the Spokes, using their dynamically obtained IP addresses (this connection is then used to share local routes). A peer-group is configured, in the 10.10.10.0/24 range in order to connect to the devices in the configured IP pool. This connection between Hub and Spokes will also allow the detection of failures in the VPN elements by using the fall-over command, and the local routes are shared by using the address-family command.

```
! Configure BGP to listen to new peers in the range 10.10.10.0/24 (configured IP pool).
router bgp 1
bgp router-id 10.10.10.1
bgp listen range 10.10.10.0/24 peer-group FLEXVPN_SPOKES
! Configure BGP fall-over to detect router failures
neighbor FLEXVPN_SPOKES peer-group
neighbor FLEXVPN_SPOKES remote-as 1
neighbor FLEXVPN_SPOKES fall-over
neighbor FLEXVPN_SPOKES transport connection-mode passive
neighbor FLEXVPN_SPOKES update-source Lo 0
network 192.168.1.0 mask 255.255.255.0
! Enable the sharing of Spoke LAN Routes using BGP address-family command
address-family ipv4
neighbor FLEXVPN_SPOKES activate
neighbor FLEXVPN_SPOKES default-originate
neighbor FLEXVPN_SPOKES prefix-list DENY_DEFAULT in
```

The configured authorization policy, which assigns IPs in the range of 10.10.10.2-200, defines attributes which are pushed to the peers when they connect to the VPN. The policy is shown in figure 124.

```
IKEv2 Authorization Policy : IKEV2_AUTHORIZATION
  IPV4 Address Pool : FLEXVPN_POOL
  route set interface
  route accept any tag : 1 distance : 1
Router#sh ip local pool FLEXVPN_POOL
 Pool                 Begin           End             Free  In use   Blocked
 FLEXVPN_POOL         10.10.10.2      10.10.10.200     198       1         0
```

Figure 124 - IKEv2 Authorization Policy

The Spoke configuration is the same as the Hub except for the virtual-template Tunnel and for the BGP configuration. The created tunnel has a negotiated IP address, and tunnel destination is the HUB ip address. The added configuration for the Spoke is as follows.

```
! Configure Tunnel interface to connect to Hub
int Tunnel0
!IP address is negotiated
ip address negotiated
tunnel source g0/0
tunnel mode ipsec ipv4
tunnel destination 200.1.1.1
tunnel protection ipsec profile IpsecPROFILE
shutdown
exit
! Configure BGP to connect to Hub, and detect fall-over
router bgp 1
bgp router-id 2.2.2.2
neighbor 10.10.10.1 remote-as 1
neighbor 10.10.10.1 update-source Tunnel 0
neighbor 10.10.10.1 fall-over
network 2.2.2.2 mask 255.255.255.255
network 192.168.2.0 mask 255.255.255.0
! Configure address family to obtain routes from HUB (for other Spoke LANs)
address-family ipv4
neighbor 10.10.10.1 activate
end
```

After the Tunnel interface is activated on the Spoke, it initiates the connection to the Hub. Figure 125 shows the debug of the IKEv2 exchange on the Hub (command “debug crypto ikev2”). This shows the reception of the IKE_SA_INIT Request. The Hub starts to authenticate the Spoke in the exchange.

Figure 125 – PKI validation after IKE_SA_INIT Message received by Hub

The Spoke then sends an IKE_AUTH Request, shown in figure 126. It first gets its own authentication data, and signs it with its private key, it also sends its identification (hostname=c1.teste.com, cn=c1, o=teste.com).

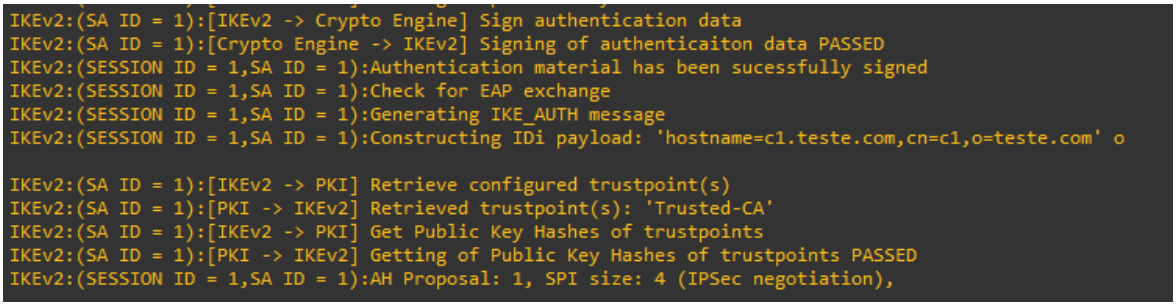


Figure 126 - Spoke sends IKE_AUTH request to Hub

When the Hub receives the IKE_AUTH Request, it searches for a policy matching the peer's identity, in this case the policy configured matches all addresses (0.0.0.0). It then retrieves the peer's authentication data and validates it, also verifying that the signature on the authentication data is correct. When all these verifications are done, the Hub sends its own authentication information for the Spoke to authenticate, which it does. Figure 127 shows this process.

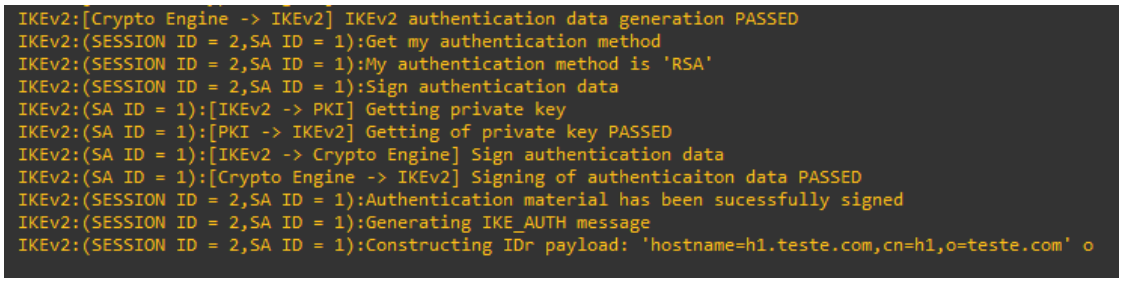


Figure 127 - Hub Processes Auth Request and Generates Auth Response

In this configuration, BGP was also used to share the routes between the Hub and Spoke. We can see in figure 128 a VPN Route being added to 10.10.10.3, and a BGP neighbor coming up. This IP address belongs to the pool mentioned previously in figure 125 and was attributed dynamically.

Figure 128 - BGP Peer Establishment

Figure 129 shows a Wireshark packet capture of the IKEv2 Exchange, and also shows a BGP packet sent through the VPN. The outer IP header has the physical interface IPs, while the inner header has the Tunnel addresses.

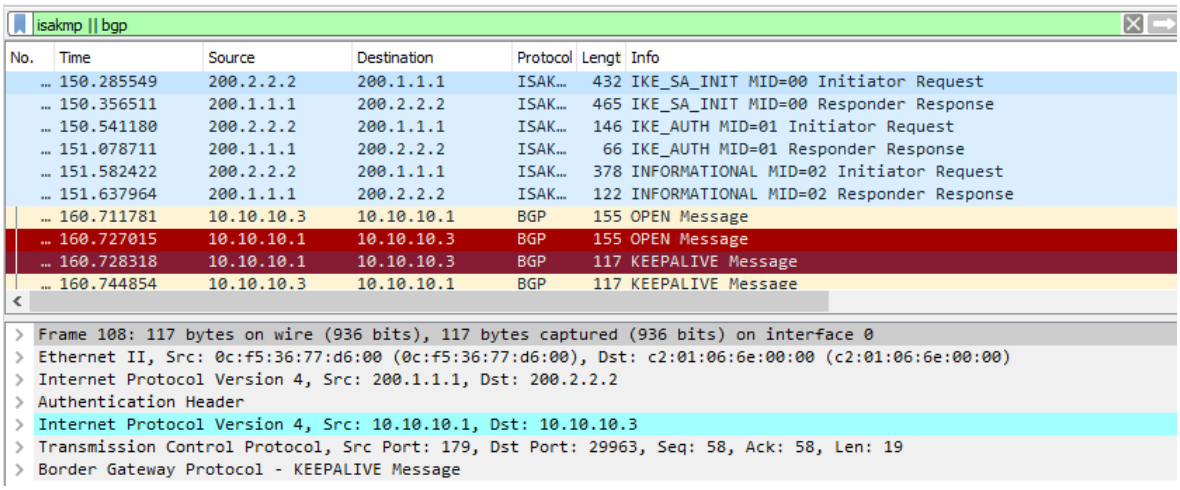


Figure 129 - FlexVPN Wireshark capture

In figure 130, the Hub interfaces and routing table are shown. The route to 10.10.10.3 (Spoke Tunnel Interface) is static, and was learned via IKEv2, but the route to the Spoke's private network (192.168.2.0) was sent via BGP by the Spoke, through the VPN.

```
Router#sh ip int br
Interface                  IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0         200.1.1.1       YES manual up                    up
GigabitEthernet0/1         192.168.1.1     YES manual up                    up
GigabitEthernet0/2         unassigned      YES unset  administratively down down
GigabitEthernet0/3         unassigned      YES unset  administratively down down
Loopback0                  10.10.10.1      YES manual up                    up
Virtual-Access1            10.10.10.1      YES unset  up                    up
Virtual-Template1          10.10.10.1      YES unset  up                    down
Router#sh ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is not set

      2.0.0.0/32 is subnetted, 1 subnets
O        2.2.2.2 [110/12] via 200.1.1.10, 05:40:29, GigabitEthernet0/0
      3.0.0.0/32 is subnetted, 1 subnets
O        3.3.3.3 [110/3] via 200.1.1.10, 05:44:47, GigabitEthernet0/0
      10.0.0.0/8 is variably subnetted, 3 subnets, 2 masks
C        10.10.10.0/24 is directly connected, Loopback0
L        10.10.10.1/32 is directly connected, Loopback0
S        10.10.10.3/32 is directly connected, Virtual-Access1
      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.1.0/24 is directly connected, GigabitEthernet0/1
L        192.168.1.1/32 is directly connected, GigabitEthernet0/1
B     192.168.2.0/24 [200/0] via 10.10.10.3, 05:35:13
      200.1.1.0/24 is variably subnetted, 2 subnets, 2 masks
C        200.1.1.0/24 is directly connected, GigabitEthernet0/0
L        200.1.1.1/32 is directly connected, GigabitEthernet0/0
O     200.2.2.0/24 [110/11] via 200.1.1.10, 05:44:47, GigabitEthernet0/0
O     200.3.3.0/24 [110/2] via 200.1.1.10, 05:44:47, GigabitEthernet0/0
O     200.4.4.0/24 [110/2] via 200.1.1.10, 05:44:47, GigabitEthernet0/0
Router#
Router#
```

Figure 130 - Hub Interfaces and Routig Table

This means that the VPN was established successfully using PKI and Digital Certificates.

13. 802.1X (EAPoL) – Experiment

To build this topology the GNS3 network tool was used. The set up includes a CiscoIOSv1 switch, a Docker appliance, which acted as the AAA server, and an EAP client using docker(ist-tools-1) with wpa_supplicant. There are two vlans, VLAN1 in which the AAA server is located, and VLAN20, where the client is connected. The objective of this experiment is to have the client authenticated through EAP, specifically EAPoL (EAP over LAN). Figure 131 shows the toplogy used in this experiment.

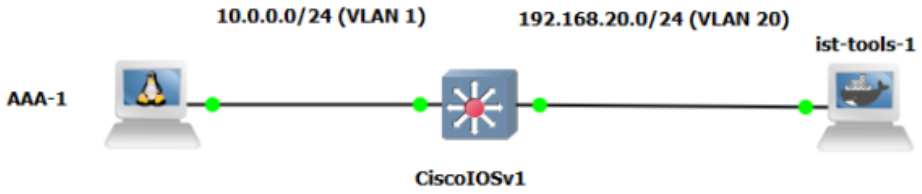


Figure 131 - EAPoL topology

To configure a client with username “bob”, password “gns3”, using PEAP (Protected EAP), on the docker container, the following configuration was used.

```
!Change wpa.conf file
cat - > /root/wpa.conf << EOF
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
eapol_version=2
ap_scan=0
network={
 key_mgmt=IEEE8021X
!Set EAP protocol to use
 eap=PEAP
!Set username and password
 identity="bob"
 password="gns3"
 phase2="autheap=MSCHAPV2"
 eapol_flags=0
}
EOF
```

Afterwards, turn on wpa_supplicant using the following command:

- wpa_supplicant -Dwired -ieth0 -c/root/wpa.conf &

After this configuration, all devices are turned on and the client will try to be authenticated. Figure 132 shows the EAP authentication on the user.

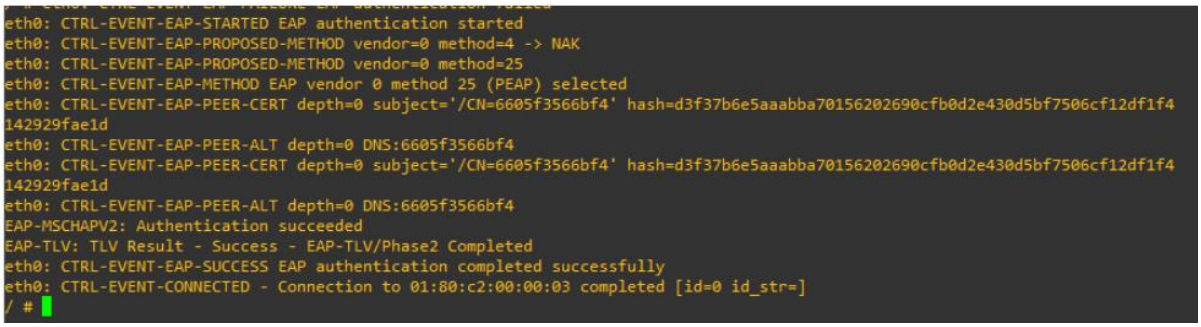


Figure 132 - EAP authentication on the user

Using Wireshark its possible to capture the EAP exchange between the authenticatior and the user, and its also possible to capture the RADIUS exchange between the authenticator and the AAA server. Figure 133 shows the RADIUS exchange. The messages Access-Request and Access-Challenge are shown. In the request message the EAP payload is shown, demonstrating that the EAP information is encapsulated in the RADIUS packet, with the identity “bob” configured in this experiment.

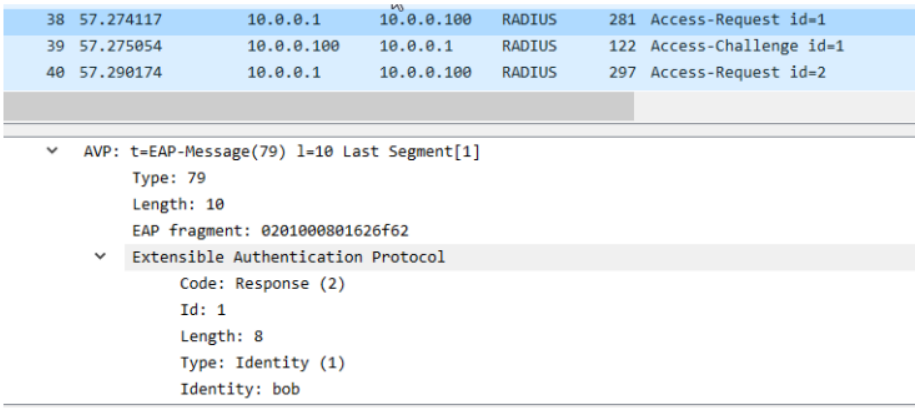


Figure 133 - RADIUS exchange between the AAA server and the authenticator

Figure 134 shows the EAP exchange between the user and the authenticator. This exchange begins with a Request Identity from the authenticator to the user, who replies with a Response message, sending its identity. The authenticator then sends a Request Message to use Protected EAP (PEAP), and the user sends an EAP Response Message. The user and the authenticator will then establish a TLS tunnel, through which the authenticator will challenge the user for the password. This exchange continues until the authenticator has finished authenticating the user and sends a Success message.

1	0.0000…	0c:d2:c0:63:66:01	Nearest	EAP	60	Request, Identity
2	0.0005…	8a:51:b0:f1:c2:79	Nearest	EAP	26	Response, Identity
4	5.1418…	0c:d2:c0:63:66:01	Nearest	EAP	60	Request, MD5-Challenge EAP (EAP-MD5-CHALLENGE)
5	5.1421…	8a:51:b0:f1:c2:79	Nearest	EAP	24	Response, Legacy Nak (Response Only)
6	5.1585…	0c:d2:c0:63:66:01	Nearest	EAP	60	Request, Protected EAP (EAP-PEAP)
7	5.1593…	8a:51:b0:f1:c2:79	Nearest	TLSv1.2	212	Client Hello
8	5.1763…	0c:d2:c0:63:66:01	Nearest	EAP	1022	Request, Protected EAP (EAP-PEAP)
9	5.1767…	8a:51:b0:f1:c2:79	Nearest	EAP	24	Response, Protected EAP (EAP-PEAP)
10	5.1939…	0c:d2:c0:63:66:01	Nearest	TLSv1.2	196	Server Hello, Certificate, Server Key Exchange, Server Hello Done
11	5.1956…	8a:51:b0:f1:c2:79	Nearest	TLSv1.2	154	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
12	5.2164…	0c:d2:c0:63:66:01	Nearest	TLSv1.2	75	Change Cipher Spec, Encrypted Handshake Message
13	5.2168…	8a:51:b0:f1:c2:79	Nearest	EAP	24	Response, Protected EAP (EAP-PEAP)
14	5.2327…	0c:d2:c0:63:66:01	Nearest	TLSv1.2	60	Application Data
15	5.2330…	8a:51:b0:f1:c2:79	Nearest	TLSv1.2	57	Application Data
16	5.2493…	0c:d2:c0:63:66:01	Nearest	TLSv1.2	92	Application Data
17	5.2496…	8a:51:b0:f1:c2:79	Nearest	TLSv1.2	111	Application Data
18	5.2655…	0c:d2:c0:63:66:01	Nearest	TLSv1.2	100	Application Data
19	5.2659…	8a:51:b0:f1:c2:79	Nearest	TLSv1.2	55	Application Data
20	5.2819…	0c:d2:c0:63:66:01	Nearest	TLSv1.2	64	Application Data
21	5.2822…	8a:51:b0:f1:c2:79	Nearest	TLSv1.2	64	Application Data
22	6.1111…	0c:d2:c0:63:66:01	Nearest	EAP	60	Success

Figure 134 - EAP exchange between user and authenticator

The following Dockerfile configuration was used to build the container.

```
FROM ubuntu:20.04
ENV TZ=Europe/Lisbon
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
RUN mkdir /pers
VOLUME /pers
RUN apt-get update && apt-get install -y iproute2 nmap nano tcpdump telnet openssh-client dnsutils yersinia ettercap-text-only dsniff fping hping3 python3-scapy apache2 wpasupplicant
```

14. FlexVPN Remote Access using AAA (EAP) and PKI - Experiment

The objective of this experiment is to demonstrate a FlexVPN Remote Access topology with EAP authentication, establishing an IPSec tunnel between a remote user and the Hub. When the client establishes a connection to the HUB, the objective is to authenticate the user with EAP (between user and HUB) and RADIUS (between HUB and AAA Server). For EAP to be used the Hub needs to have a certificate but the user requires only credentials.

In order for the Windows VM to connect to the HUB, Cisco AnyConnect Mobility Client was used (free download at https://developer.cisco.com/site/sandbox/anyconnect/). To build this FlexVPN topology the GNS3 network tool was used, one Windows 10 VM, one FlexVPN Hub (Cisco IOSv), and one AAA Server (Docker) were set up, and one more Cisco router (RA, C3725) was set up, that represents the public network. Figure 135 shows the experiment topology. The NAT cloud in the topology was used only to download Anyconnect to the windows device.

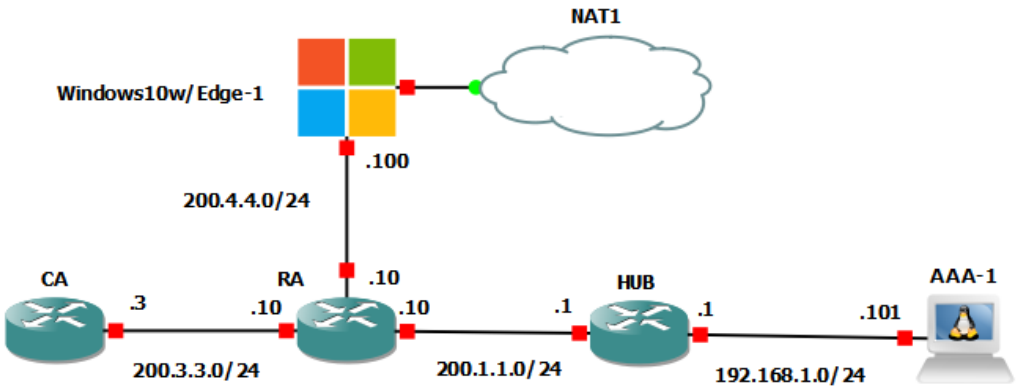


Figure 135 - Remote Access Topology

For it to be possible for the Windows device to connect via Anyconnect, the subject on the certificate must have a DNS entry with the same name used on the certificate. A DNS server entry was configured at RA using the following configuration.

```
!RA
conf t
ip dns server
ip domain-lookup
ip host h1.teste.com 200.1.1.1
```

end

The windows device was manually configured to have an IP address of 200.4.4.4 and to use RA as a DNS server. Figure 136 shows that there is connectivity between the Windows VM and the HUB, using the DNS entry.

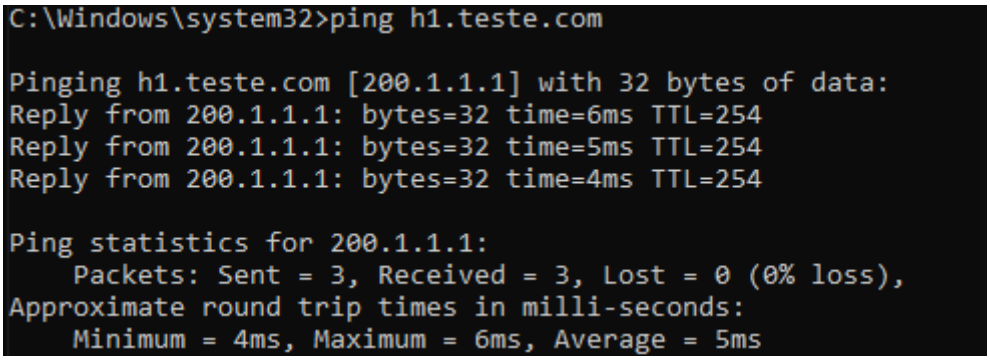


Figure 136 – Result of ping from user to h1.teste.com (Hub)

The FlexVPN server authentication needs to remain certificate-based to use EAP. This means that to use EAP, a PKI is still required. The server needs to enroll in the CA as shown in chapter 11.

For the HUB to use a AAA server, we need to configure the router to use the RADIUS server in the topology, and we need to change the previous certificate configuration to use EAP authentication. The following configurations were used.

!Configure group server
aaa group server radius freeradius
!Configure AAA server with IP, authentication and accounting ports, and server key
server-private 192.168.1.101 auth-port 1812 acct-port 1813 key gns3
aaa authentication login AUTHEN_RADIUS group freeradius
exit
!Match Anyconnect identity and enable remote EAP authentication
crypto ikev2 profile PROFILE
!Match any address
match identity remote address 0.0.0.0
!Match AnyConnect Clients
match identity remote key-id *$AnyConnectClient$*
!Set remote authentication to EAP
authentication remote anyconnect-eap aggregate
!Active AAA authentication and authorization with the details configured before
aaa authentication anyconnect-eap AUTHEN_RADIUS

aaa authorization group anyconnect-eap list FLEXVPN_LOCAL IKEV2_AUTHORIZATION
aaa authorization user anyconnect-eap cached
identity local dn

After the configuration is done, AnyConnect can be used to connect to the VPN. Figure 137 shows the name used to connect to the VPN (h1.teste.com).

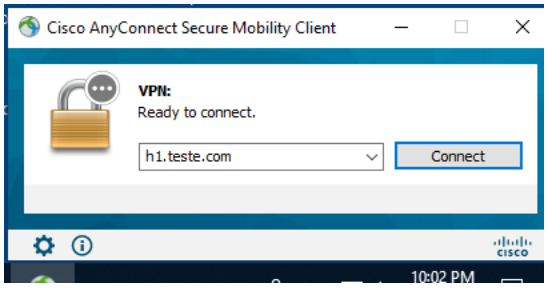


Figure 137 - Anyconnect from user to HUB

There is a security warning because the user device cannot verify the HUB certificate. This is normal because the certificate was not installed on this Windows device. Figure 138 shows this warning, but because the server was also a part of the experiment, we connect anyway.

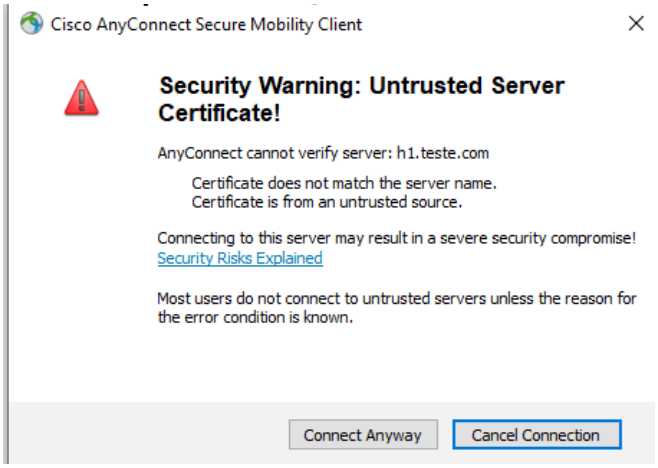


Figure 138 - Anyconnect Security Warning

During the connection, the HUB will inquire about the user EAP credentials. Figure 139 shows the popup where the username “bob” and password “gns3” were used.

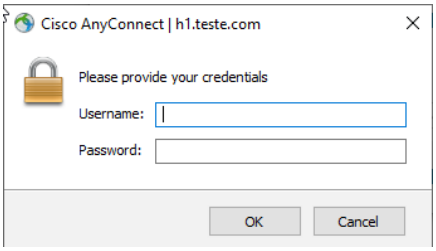


Figure 139 - EAP credentials popup

After entering the credentials, there is no visible change to the normal IKEv2 exchange, because the EAP messages are encapsulated in the IKE_AUTH message, which is encrypted. Figure 140 shows the IKEv2 exchange.

```
59 89.849762 200.4.4.100    200.1.1.1      ISAKMP    711 IKE_SA_INIT MID=00 Initiator Request
60 89.934471 200.1.1.1      200.4.4.100    ISAKMP    503 IKE_SA_INIT MID=00 Responder Response
61 89.959367 200.4.4.100    200.1.1.1      ISAKMP    766 IKE_AUTH MID=01 Initiator Request
63 90.138413 200.1.1.1      200.4.4.100    ISAKMP   1270 IKE_AUTH MID=01 Responder Response
```

Figure 140 - EAP IKEv2 Exchange

The RADIUS exchange between the HUB and the AAA server can also be captured using Wireshark. Figure 141 shows the Access Request RADIUS message from the HUB to the authentication server, and also shows the Access Accept message from the server back to the HUB.

```
15 110.3095… 192.168.1.1      192.168.1.101    RADIUS   149 Access-Request id=1
16 110.3140… 192.168.1.101    192.168.1.1      RADIUS    74 Access-Accept id=1
```

Figure 141 - RADIUS exchange between HUB and AAA server

The Access-Request message can be further analysed to verify its fields. This is shown in figure 142. The username has value “bob”, the password is encrypted, the supplicant (Calling-Station-ID) is 200.4.4.100 (Windows ip address), and the Authenticator IP (NAS) which is 192.168.1.1.

```
▼ RADIUS Protocol
    Code: Access-Request (1)
    Packet identifier: 0x1 (1)
    Length: 107
    Authenticator: a6f2779a4d55cb219482ea819924ab4f
    [The response to this request is in frame 16]
  ▼ Attribute Value Pairs
    > AVP: t=User-Name(1) l=5 val=bob
    > AVP: t=User-Password(2) l=18 val=Encrypted
    > AVP: t=Vendor-Specific(26) l=45 vnd=ciscoSystems(9)
    > AVP: t=Calling-Station-Id(31) l=13 val=200.4.4.100
    > AVP: t=NAS-IP-Address(4) l=6 val=192.168.1.1
```

Figure 142 - RADIUS Access-Request

The Access Accept message can also be analysed. This is shown in Figure 143. The only information sent (besides the user being accepted, which is inferred from the message type) is the reply message configure on the AAA server, which in this case is “Hello, <username>”.

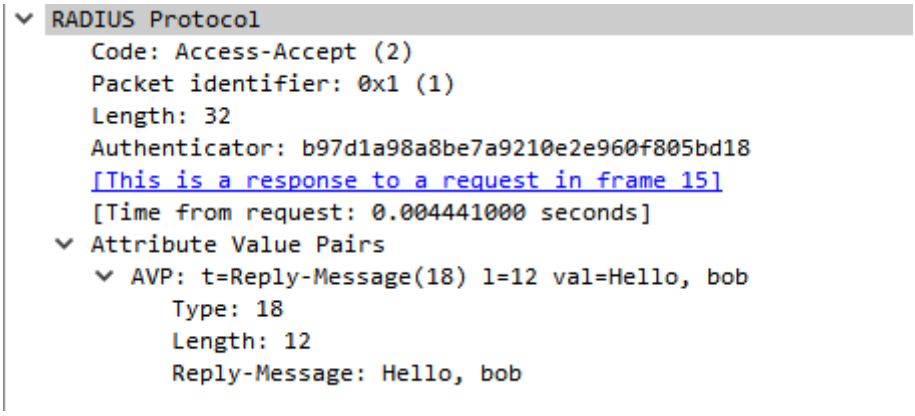


Figure 143 - RADIUS Access-Accept

When the EAP/RADIUS exchange is complete, and the user is authenticated, the IKEv2 Exchange ends with the establishment of the IPSec SA, shown in figure 144.

```
Router#sh crypto ikev2 sa
 IPv4 Crypto IKEv2  SA

Tunnel-id Local                 Remote                fvrf/ivrf            Status
1         200.1.1.1/4500        200.4.4.100/55845     none/none            READY
      Encr: AES-CBC, keysize: 256, PRF: SHA384, Hash: SHA256, DH Grp:19, Auth sign: RSA, Auth verify: AnyConnect-EAP
      Life/Active Time: 86400/11 sec

 IPv6 Crypto IKEv2  SA

Router#
```

Figure 144 - SA between HUB and Remote Access User

15. FlexVPN Redundancy - Experiment

The objective of this experiment is to configure a FlexVPN Site-to-Site topology with fault tolerance using redundancy. When the client establishes a connection to the first choice HUB, the objective is to shut down that device and have the client remain connected, or reconnect automatically, with the VPN. Figure 145 shows the experiment topology. To build this FlexVPN topology the GNS3 network tool was used, two FlexVPN Hubs and one FlexVPN Spoke were set up, using Cisco IOSv, one more Cisco router (RA, C3725) was set up, that represents the public network.

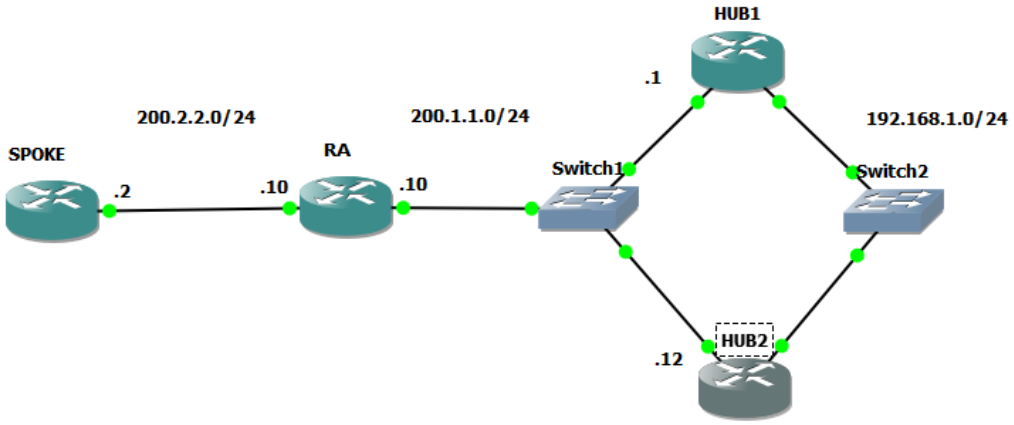


Figure 145 - Redudancy FlexVPN topology

15.1 Dual HUB Single-Cloud

Single-Cloud is when a redundant topology uses only one connection at a time. When the main HUB is down, the client needs to detect the failure, and connect automatically to the backup HUB.

We need to configure SLA tracking for the Spoke to detect if the Hub is unreachable. We then need to configure the second Hub for the Spoke to connect to when the main Hub is down. The configuration added to the Spoke to enable the Dual HUB Single-Cloud topology was the following.

```
!Enable SLA tracking to track HUB reachability
track 1 ip sla 1 reachability
ip sla 1
icmp-echo 200.1.1.1
```

```
frequency 5
ip sla schedule 1 start-time now
!Enable multiple peers and peer reactivation in FlexVPN configuration.
crypto ikev2 client flexvpn flex-client
!Main HUB
peer 1 200.1.1.1 track 1
!Secondary HUB
peer 2 200.1.1.12
peer reactivate
client connect Tunnel 0
end
```

The peer number in the configuration is the peer priority, which means that the client will prioritize peer 1 over peer 2. The client will track only the main HUB. If the connection goes down, it will still be tracked. When the connection is working again, the client will reconnect with the main HUB. For this reason, there's no need to track the secondary HUB, because if the client is connected to it, it means that the main HUB is also offline.

Figure 146 shows the Spoke (200.2.2.2) connecting to the main HUB (200.1.1.1), and also shows the established SA.

```
*Oct 12 20:55:06.106: %FLEXVPN-6-FLEXVPN_CONNECTION_UP: FlexVPN(flex-client) Client_public_addr = 200.2.2.2 Server_public_ad
dr = 200.1.1.1ikev2-flexvpn)#end
Router#sh crypto ikev2 sa
 IPv4 Crypto IKEv2  SA

Tunnel-id Local                 Remote                fvrf/ivrf            Status
2         200.2.2.2/500         200.1.1.1/500         none/none            READY
      Encr: AES-CBC, keysize: 192, PRF: SHA256, Hash: SHA256, DH Grp:2, Auth sign: PSK, Auth verify: PSK
      Life/Active Time: 86400/68 sec

 IPv6 Crypto IKEv2  SA

Router#
```

Figure 146 - First connection and SA at the Spoke

When the main HUB is shutdown, the client automatically detects the failure connects to the backup up. This is shown in Figure 147.

```
*Oct 12 21:07:50.570: %TRACK-6-STATE: 1 ip sla 1 reachability Up -> Down
*Oct 12 21:07:50.574: %FLEXVPN-6-FLEXVPN_CONNECTION_DOWN: FlexVPN(flex-client) Client_public_addr = 200.2.2.2 Server_public_addr = 200.1.1.1
*Oct 12 21:07:51.267: %FLEXVPN-6-FLEXVPN_CONNECTION_UP: FlexVPN(flex-client) Client_public_addr = 200.2.2.2 Server_public_addr = 200.1.1.12
```

Figure 147 - SLA failure detection and connection to backup HUB

After the Spoke detects the main HUB's failure, it will delete the SA it had before, and will keep only the SA established with the backup HUB. This is shown in Figure 148.

```
Router#sh crypto ikev2 sa
 IPv4 Crypto IKEv2  SA

Tunnel-id Local                 Remote                fvrf/ivrf            Status
1         200.2.2.2/500         200.1.1.12/500        none/none            READY
      Encr: AES-CBC, keysize: 192, PRF: SHA256, Hash: SHA256, DH Grp:2, Auth sign: PSK, Auth verify: PSK
      Life/Active Time: 86400/304 sec

 IPv6 Crypto IKEv2  SA

Router#
```

Figure 148 - SA between the Spoke and Backup HUB

15.2 Dual HUB Dual-Cloud

Dual-Cloud is when a redundant topology uses two connections at a time. The client has two active tunnels, one with each HUB. When the main HUB is down, the client remains connected to the VPN with the secondary tunnel. In order to configure Dual-Cloud redundancy, all that is required is to setup a second tunnel at the Spoke, directed at the backup HUB. The following configuration was added.

```
int Tunnel1
ip address negotiated
tunnel source g0/0
tunnel mode ipsec ipv4
tunnel destination 200.1.1.12
tunnel protection ipsec profile IpsecPROFILE
exit
```

Wireshark can be used to check the IKEv2 exchanges. This is shown in Figure 149. Checking the Time column, we can see that both exchanges were completed in less than second.

No.	Time	Source	Destination	Protocol	Lengt	Info
51	65.068675	200.2.2.2	200.1.1.1	ISAKMP	432	IKE_SA_INIT MID=00 Initiator Request
52	65.127305	200.1.1.1	200.2.2.2	ISAKMP	432	IKE_SA_INIT MID=00 Responder Response
53	65.176249	200.2.2.2	200.1.1.1	ISAKMP	602	IKE_AUTH MID=01 Initiator Request
54	65.327838	200.1.1.1	200.2.2.2	ISAKMP	298	IKE_AUTH MID=01 Responder Response
55	65.388549	200.2.2.2	200.1.1.12	ISAKMP	432	IKE_SA_INIT MID=00 Initiator Request
58	65.447877	200.1.1.12	200.2.2.2	ISAKMP	432	IKE_SA_INIT MID=00 Responder Response
61	65.544111	200.2.2.2	200.1.1.12	ISAKMP	602	IKE_AUTH MID=01 Initiator Request
62	65.711498	200.1.1.12	200.2.2.2	ISAKMP	298	IKE_AUTH MID=01 Responder Response

Figure 149 - IKEv2 Exchanges between Spoke and both HUBs

The command “sh crypto ikev2 sa” can be used on the Spoke to validate that there are two active SAs at this time. This is shown in Figure 150.

```
Router#sh crypto ikev2 sa
 IPv4 Crypto IKEv2  SA

Tunnel-id Local                 Remote                fvrf/ivrf            Status
2         200.2.2.2/500         200.1.1.12/500        none/none            READY
      Encr: AES-CBC, keysize: 192, PRF: SHA256, Hash: SHA256, DH Grp:2, Auth sign: PSK, Auth verify: PSK
      Life/Active Time: 86400/191 sec

Tunnel-id Local                 Remote                fvrf/ivrf            Status
1         200.2.2.2/500         200.1.1.1/500         none/none            READY
      Encr: AES-CBC, keysize: 192, PRF: SHA256, Hash: SHA256, DH Grp:2, Auth sign: PSK, Auth verify: PSK
      Life/Active Time: 86400/191 sec
```

Figure 150 - IKEv2 SAs at the Spoke

16. Conclusion

Network security and secure access is a vital issue in modern times, and if done correctly can bring great benefits to network usage, and avoid many problems and complications from using an unsafe network. There are many techniques and protocols that make a network infrastructure secure. The deployment of conventional VPNs, namely those based on IPSec, ensure that the information and accesses to networks are secured. FlexVPN is a solution resulting from the integration of older, conventional, VPN technologies (like DMVPN) with PKIs and AAA. It allows the usage of several key features and benefits from other VPN technologies like direct Spoke-to-Spoke communications and load balancing and integrates them with AAA to assure that someone accessing a network is authenticated and authorized to access specific network resources. It also integrates those same features with PKI, to have a secure key distribution mechanism, used to establish encrypted tunnels securely.

SDWAN is a newer, centralized technology that can be used to deploy VPNs quickly across the Internet, with minimal configuration. This technology uses Software-Defined networking to create a virtualized WAN in which VPNs can be easily configured and managed. The centralization aspect of this technology allows the VPN networks to be easily scalable because there's only one configuration point, and the integration of SDWAN with PKIs also improve this aspect by providing each new SDWAN element with a digital certificate used to establish secure communications.

PKIs are infrastructures in charge of authenticating users and devices, they can also be used to provide secure communications. Only by itself, PKIs aren't useful, but by integrating them with other solutions, like SDWAN and VPNs, we can make these become more secure and scalable by performing certificate distribution, which facilitates the authentication of users.

Securing routing protocols is of the utmost importance as well. The protocols form the network infrastructure used by VPNs and SDWAN. Without them, no information could be transmitted, not on the Internet, nor on private networks. The routing protocols must be correctly configured, and secured, for the higher-level technologies to run correctly and safely.

With this study, we can conclude that conventional VPNs are better in smaller environments, where there is no need for a real, scalable infrastructure in order to support the user demands, and SDWAN is better in larger environments, where there is a need for scalability, centralized control, and monitoring.

In this MSc disseration, concepts and techniques related to network security were demonstrated, and experiments were done containing those concepts and techniques. Many experiences were carried out regarding FlexVPN and its integration with AAA and PKIs. The operation of PKIs and its design were also demonstrated in the experiments. Several SDWAN solutions where shown and demonstrated, and its protocols and features studied as well and compared to conventional VPNs. Regarding routing security, a RIPv2 attack and it's corresponding countermeasure was also shown.

The work developed in this dissertation was able to show how to strengthen and secure a network, to allow secure accesses from other sites and remote users, and also what protocols and systems can be used behind a secure network infrastructure.

Bibliography

[1] M. S. Charlie Kaufman, Radia Perlman, *Network Security*. 2002.

[2] P. E. C. Kaufman, P. Hoffman, Y. Nir, "Internet Key Exchange Protocol Version 2 (IKEv2)," 1998. https://tools.ietf.org/html/rfc5996.

[3] J. T. D. Maughan, M. Schertler, M. Schneider, "Internet Security Association and Key Management Protocol (ISAKMP)," 1998. https://tools.ietf.org/html/rfc2408.

[4] A. I. Graham Bartlett, *IKEv2 IPsec Virtual Private Networks*. 2017.

[5] B. C. J. Luciani, D. Katz, D. Piscitello, "NBMA Next Hop Resolution Protocol (NHRP)," 1998. https://tools.ietf.org/html/rfc2332.

[6] W. S. C. Rigney, S. Willens, A. Rubens, "Remote Authentication Dial In User Service (RADIUS)," 2000. https://tools.ietf.org/html/rfc2865.

[7] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, "Extensible Authentication Protocol (EAP)," 2004. https://tools.ietf.org/html/rfc3748.

[8] H. T. F. Bersani, "The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method," 2007. https://tools.ietf.org/html/rfc4764.

[9] R. H. D. Simon, B. Aboba, "The EAP-TLS Authentication Protocol," 2008. https://tools.ietf.org/html/rfc5216.

[10] E. Y. M. Chiba, A. Clemm, S. Medley, J. Salowey, S. Thombare, "Cisco Service-Level Assurance Protocol," 2013. https://tools.ietf.org/html/rfc6812.

[11] R. V. Haseeb Niazi, Nipul Shah, Biao Zhou, Varun SethiShashi Shastry, “Group Encrypted Transport VPN (GETVPN) Design and Implementation Guide,” 2018. https://www.cisco.com/c/dam/en/us/products/collateral/security/group-encrypted-transport-vpn/GETVPN_DIG_version_2_0_External.pdf.

[12] Cisco, “Dynamic Multipoint VPN Configuration Guide,” 2020. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/sec_conn_dmvpn/configuration/15-mt/sec-conn-dmvpn-15-mt-book/sec-conn-dmvpn-dmvpn.html.

[13] B. Atri, Z. Wen, and N. Nehal, “IPSec Anti-Replay Check Failures,” 2016. https://www.cisco.com/c/en/us/support/docs/ip/internet-key-exchange-ike/116858-problem-replay-00.html.

[14] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,” 2008. https://tools.ietf.org/html/rfc5280.

[15] P. Gutmann, “Simple Certificate Enrolment Protocol,” 2020. https://datatracker.ietf.org/doc/html/rfc8894.

[16] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, “X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP,” 2013. https://datatracker.ietf.org/doc/html/rfc6960.

[17] J. Hawkinson and T. Bates, “Guidelines for creation, selection, and registration of an Autonomous System (AS),” 1996. https://tools.ietf.org/html/rfc1930.

[18] G. Malkin, “RIP Version 2,” 1998. https://tools.ietf.org/html/rfc2453.

[19] J. Moy, “OSPF Version 2,” 1998. https://tools.ietf.org/html/rfc2328.

[20] S. H. Y. Rekhter, T. Li, “Border Gateway Protocol (BGP),” 2006. https://tools.ietf.org/html/rfc4271.

[21] RIPE, “What is RPKI?” https://www.ripe.net/manage-ips-and-asns/resource-management/rpki/what-is-rpki.

[22] K. Robertson, “SDWAN,” 2021. https://www.ibm.com/services/network/sd-wan.

[23] Nokia, *Nuage VNS User Guide*. 2020.

[24] M. Mahalingam *et al.*, “Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks,” 2014. https://datatracker.ietf.org/doc/html/rfc7348.

[25] Cisco, “The Cisco SD-WAN Solution,” 2021. https://www.cisco.com/c/en/us/td/docs/routers/sdwan/configuration/sdwan-xe-gs-book/system-overview.html#c_The_Viptela_SEN_9286.xml.

[26] FlexiWan, “FlexiWan Documentation,” 2019. https://docs.flexiwan.com/.