Road Warrior: First Practical Assignment of STI

Tiago Ventura – 2019243695 Sancho Simões – 2019217590

Masters in Intelligent Systems University of Coimbra

Coimbra, 12th March 2023

Conteúdo

1 Introduction

This report presents an overview of a project aimed at establishing a secure communication system between a remote client, also known as a "road warrior,"and a VPN gateway. The primary objective of this project is to enable access to internal resources, in this case, an Apache server, while ensuring the security of the VPN tunnel. In addition, the project includes implementing two-factor authentication for both OpenVPN and Apache services and managing Public Key Infrastructure (PKI), including certification authorities, X.509 certificates, revocation, and an Online Certificate Status Protocol (OCSP) service.

To achieve the project's primary goal, we will use OpenVPN to create a VPN tunnel in the "road warrior"scenario that allows access to services in the internal network, specifically a web server running Apache. To ensure the security of the VPN tunnel, both the road warrior and the VPN gateway must have valid X.509 certificates. These certificates will be created using a private Certification Authority (CA).

In addition to authentication for the VPN tunnel, users who establish remote connections to the VPN gateway and those who connect to the Apache server must use two-factor authentication. Client authentication via X.509 certificates must also be implemented in Apache. The VPN gateway and Apache web server will verify the validity of certificates using OCSP and revocation information from the CA.

This report provides a comprehensive description of the project, including the primary goals, results, tests, and technologies used to achieve them. It also covers various aspects of PKI management, including certification authorities, X.509 certificates, revocation, and OCSP. Finally, the report concludes with a summary of the project's achievements and any recommendations for future improvements.

2 Concepts Definitions

This report aims to explore several important concepts that have been frequently used in the work that has been done. These concepts are essential to comprehend the technology and processes involved in the work and are commonly encountered in various fields related to information security and networking.

The report will cover various key concepts, including Virtual Private Networks (VPN), Online Certificate Status Protocol (OCSP), X.509 digital certificates, Two-Factor Authentication (2FA), and other essential topics. These concepts are fundamental to securing network communication and ensuring the confidentiality, integrity, and availability of sensitive data.

It is essential to comprehend these concepts for anyone who deals with network security and authentication as they form the basis of modern security infrastructure. Thus, the following sections will define each concept, explain its significance, and discuss its practical applications.

2.1 Virtual Private Network (VPN)

A VPN is a technology that creates a secure and private network connection between two devices over the internet or any other public network. It enables users to access the internet securely by encrypting the data and establishing a secure tunnel through which the data travels between the devices. VPNs are commonly used by businesses and individuals to protect their online privacy, secure their internet connections, and access resources that are restricted by geographical location or other access restrictions.

2.2 Road Warrior

In the context of VPN (Virtual Private Network), a road warrior refers to a remote user who needs secure access to a private network from any location outside the network. A road warrior can be an employee, contractor, or any authorized individual who requires remote access to the internal resources of an organization, such as files, email, and applications. Road warriors are often mobile and may use various devices, such as laptops, smartphones, and tablets, to connect to the VPN. The term "road warrior"is commonly used to refer to individuals who frequently travel or work remotely and need secure access to the private network from various locations. VPN technology enables road warriors to establish a secure encrypted connection to the internal network and access resources as if they were physically present on the organization's premises.

2.3 Online Certificate Status Protocol (OCSP)

OCSP is a protocol used to determine the revocation status of an SSL/TLS digital certificate. When a user visits a website that uses SSL/TLS encryption, their browser checks the certificate's revocation status to ensure it has not been revoked by the certificate authority. The OCSP protocol enables this check to be performed quickly and securely by querying the certificate authority's OCSP responder.

2.4 X.509

X.509 is a standard format for digital certificates used in public key infrastructure (PKI). X.509 certificates contain information about the identity of the certificate holder and the public key associated with that identity. They are commonly used to establish secure connections over the internet and are an essential component of SSL/TLS encryption. X.509 certificates are issued by trusted third-party certificate authorities and are used to verify the identity of the certificate holder in digital transactions.

2.5 Certification Authority - CA

A Certification Authority (CA) is an entity that issues digital certificates that certify the ownership of a public key by the named subject of the certificate. In other words, a CA is a trusted third party that verifies the identity of individuals, organizations, or devices and vouches for their public keys.

The CA assures others that the public key in a certificate belongs to the entity identified in the certificate. This is done through a process called certificate signing, where the CA verifies the identity of the certificate's subject and signs the certificate with its private key. By doing so, the CA confirms that the subject is trustworthy and the public key in the certificate can be trusted for secure communication.

In addition to issuing and signing certificates, a CA is also responsible for managing and maintaining a Certificate Revocation List (CRL), which contains information about certificates that have been revoked or are no longer valid. The CA must also provide an Online Certificate Status Protocol (OCSP) service that can be used to check the validity of a certificate in real time.

Certification Authorities are a critical component of public key infrastructure (PKI), which is a set of policies, processes, and technologies used to establish and manage digital certificates and public-private key pairs. PKI enables secure communication over insecure networks by providing confidentiality, integrity, and authentication services.

2.6 Two Factor Authentication

Two Factor Authentication (2FA) is a security mechanism that requires two different methods of authentication to verify a user's identity. Typically, this involves something the user knows (such as a password or PIN) and something the user has (such as a security token or smartphone). By requiring two different forms of authentication, 2FA can significantly improve the security of online accounts and reduce the risk of unauthorized access.

3 Used Technologies

3.1 OpenVPN

OpenVPN is an open-source software application used to create a secure and private network connection between two devices over the internet or any other public network. It uses SSL/TLS encryption to secure the connection and can be used to create both site-to-site VPNs and remote access VPNs. OpenVPN is highly configurable and can be used on a wide range of operating systems and devices, making it a popular choice for businesses and individuals looking to secure their internet connections.

3.2 OpenSSL

OpenSSL is an open-source software library used to implement SSL/TLS encryption in applications and services. It provides a range of cryptographic functions, including encryption, decryption, digital signature generation and verification, and key generation and management. OpenSSL is used in a wide range of applications and services, including web servers, email servers, and VPNs, and is considered one of the most widely used and tested SSL/TLS implementations.

3.3 Apache

Apache is an open-source web server software that is used to serve web pages over the internet. It is highly configurable and supports a wide range of technologies and programming languages, including PHP, Perl, and Python. Apache also supports SSL/TLS encryption and can be used to serve secure web pages using HTTPS. Apache is one of the most popular web server software applications and is used by millions of websites worldwide.

3.4 Google Authenticator

Google Authenticator is a software-based two-factor authentication (2FA) solution that provides an additional layer of security for user accounts. It works by generating one-time passwords (OTPs) that are required in addition to the user's regular password to log in to their account.

To use Google Authenticator, users must first install the app on their smartphone or tablet. Once installed, they can then link it to their account by scanning a QR code or manually entering a secret key provided by the service they are trying to secure. From that point on, the app generates a new OTP every 30 seconds, which the user must enter in addition to their password when logging in.

Google Authenticator is widely used by a variety of online services, including social media sites, email providers, and financial institutions, to enhance the security of user accounts. It is known for its ease of use and effectiveness in thwarting unauthorized access to accounts.

3.5 PAM

PAM stands for Pluggable Authentication Modules, which is a mechanism in Linux and other Unix-like operating systems that provides a way for applications and services to authenticate users. PAM allows for authentication to be handled by separate modules that can be plugged into the system, rather than being hard-coded into the application or service itself.

PAM provides a flexible and extensible framework for authentication, allowing administrators to configure authentication policies that meet the needs of their organization. PAM modules can implement a wide range of authentication methods, including passwords, biometrics, smart cards, and more.

PAM is used by a wide range of Linux and Unix-like applications and services, including login, sshd, su, sudo, and more. By using PAM, these applications and services can leverage the authentication capabilities of the system, without having to implement authentication themselves.

PAM configuration is typically done using text-based configuration files located in the /etc/pam.d directory. These configuration files define the PAM service name, the order in which PAM modules should be executed, and the options to be passed to each module.

Overall, PAM is an important component of Linux and other Unix-like operating systems, providing a powerful and flexible mechanism for authentication.

3.6 PGP

PGP (Pretty Good Privacy) is a software program used to encrypt and decrypt email messages and other data. PGP uses public key cryptography to ensure that only the intended recipient can decrypt the message. PGP keys are used to encrypt and decrypt messages and consist of a public key (which can be shared with others) and a private key (which must be kept secret). PGP keys are commonly used for secure email communication and file sharing and can help protect sensitive information from unauthorized access.

4 Scenario

The scenario involves setting up a VPN tunnel for remote access, allowing road warriors to connect to the VPN gateway and remotely access hosts in the Internal network. Additionally, the road warriors should be able to access an Apache web server via HTTPS once the VPN tunnel is established.

To ensure secure access, several configuration requirements need to be considered. Firstly, remote clients must possess a valid X.509 certificate created with the private Certification Authority (CA) of the scenario to establish a VPN tunnel with the VPN gateway. Secondly, mutual authentication is required between the road warrior and the VPN gateway using X.509 digital certificates. Thirdly, the VPN gateway should verify the validity of the X.509 certificate presented by the road warrior using the Online Certificate Status Protocol (OCSP) and refuse the connection if the certificate is revoked. Finally, two-factor authentication is required, where the user must present a valid username and password, plus a one-time password (authentication token) to authorize remote access.

In the case of accessing the Apache web server, two-factor authentication is also required. The client must present a valid X.509 certificate (issued with the private CA of the scenario) and a valid one-time password (authentication token) to authenticate with the server.

To generate one-time passwords, the Time-based One-time Password Algorithm (TOTP) is used. This algorithm employs a secret key shared between the user (client) and the remote service plus a timestamp obtained from the current system time, to obtain a one-time password. An application like Google Authenticator can be used to generate a one-time password that can be used to authenticate the user with a remote service.

To implement the private CA, OpenSSL is used. OpenSSL is a robust, full-featured open-source toolkit implementing the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols, as well as a full-strength general-purpose cryptography library. The private CA is used to issue certificates for the VPN gateway, VPN client, Apache web server, and clients (users) connecting to the Apache web server. The CA also allows the revocation of certificates previously issued and supports an OCSP responder.

In terms of testing the implemented functionalities, various tests can be performed to validate the VPN tunnel's functionality and the two-factor authentication required for remote access and Apache web server access.

In summary, the scenario involves setting up a VPN tunnel for remote access, with the addition of two-factor authentication for remote access and Apache web server access. OpenSSL is used to create a private CA, issue and revoke X.509 digital certificates, and support an OCSP responder. The implementation can be tested to validate the functionality of the VPN tunnel and the two-factor authentication requirements.

5 Illustrations

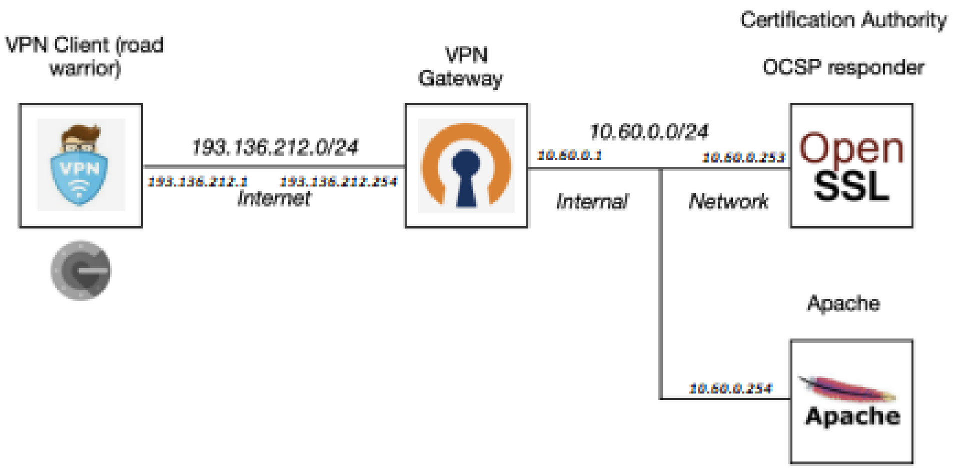

Figure 1 – Scenario for the Practical Assignment #1

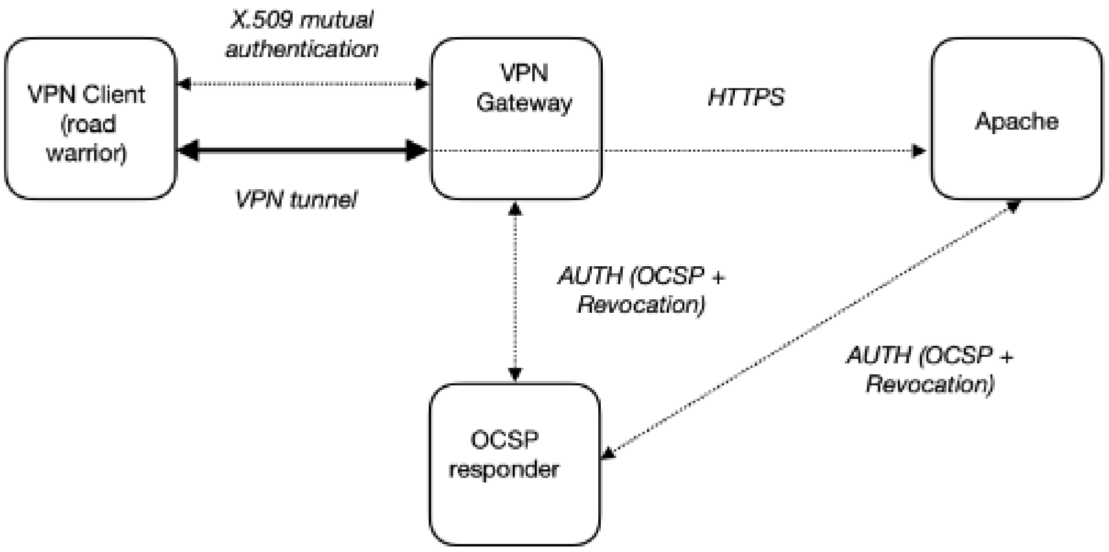

Figure 2 – X.509 mutual authentication and OCSP

For our project, we used a virtual machine-based setup, where each service was deployed on a separate virtual machine. This allowed us to simulate a realistic network environment and test our implementation under controlled conditions. Specifically, we used four virtual machines, three for the services: one for the OCSP service, one for the Apache web server, and one for the VPN gateway (i.e., the endpoint of the VPN tunnel). Also one machine was used for the VPN client (road-warrior).

By using virtual machines, we were able to isolate each service and prevent potential conflicts or interference between them. Additionally, we could easily reconfigure or replicate the setup as needed for testing purposes. Overall, this approach provided a flexible and efficient way to validate the functionalities implemented in our project. Even though this approach needs high computational resources, fortunately, we had one setup that could correspond to those needs.

6 Machine Configuration

We created four virtual machines by cloning an initial CentOS virtual machine image, and each virtual machine had its network adapter configured in Bridged mode to ensure internet connectivity.

By using the Bridged mode, each CentOS virtual machine was able to connect to the host machine's network interface and obtain its IP address on the same network as the host and other machines. This allowed us to simulate a realistic network environment in which each virtual machine could communicate with each other and with other machines on the network, just like in a physical network.

7 Network/IP Configuration

Regarding the IP addresses, the VPN client has the IP address of **193.136.212.1** and the VPN tunnel has the IP address of **193.136.212.254** on the client side. On the Apache and OCSP side, the VPN tunnel has the IP address of **10.60.0.1**, the OpenSSL has the IP address of **10.60.0.253**, and the Apache has the IP address of **10.60.0.254**.

To configure the Apache server to be able to contact the client, a static route needs to be created through the OpenVPN server. This can be achieved by adding the following command:

ip route add 10.8.0.0/24 via 10.60.0.1 dev enp0s10 metric 5

This command sets a static route for the IP range 10.8.0.0/24 to go through the OpenVPN server at 10.60.0.1, using the enp0s10 network interface. The metric value of 5 is specified to give this route a lesser preference than the default gateway, ensuring that it is only used when necessary.

By using this approach, it is not necessary to create a new default gateway, which would result in losing access to the internet. This static route provides a specific route for the OpenVPN client network while allowing the server to continue using the existing default gateway for all other traffic.

During the implementation of the practical assignment, we faced some issues with the firewall settings that were preventing proper communication between the different components of the setup. After some investigation, we identified that the firewalld service was blocking some necessary ports for the VPN and Apache services. To avoid further issues, we decided to temporarily stop and disable the firewalld service using the commands "**systemctl stop firewalld**"and "**systemctl disable firewalld**". While this is not a recommended security practice for a production environment, it helped us to ensure the correct functioning of the setup during our testing and validation phase. Once we completed our tests and ensured that all components were properly configured, we re-enabled the firewall and configured the necessary rules to allow the required traffic. Additionally, we disabled SELinux to ensure that it did not interfere with the functioning of the project.

8 Creation of the Private Certification Authority using OpenSSL

To create a private certification authority (CA) using OpenSSL, we used the following commands in the OCSP VM that has a database with the certificates which are distributed by the remaining VMs:

8.1 Command 1: Generation of the private key

We generated a private key for the CA using the following command:

```
openssl genrsa -out ca.key 4096
```

With this, we created 4096-bit RSA key and saved it to a file named "ca.key".

8.2 Command 2: Creation of the self-signed certificate

Then we used the private key to create a self-signed certificate for the CA with the following command:

```
openssl req -new -x509 -days 3650 -key ca.key -out ca.crt
```

With this we created a certificate that is valid for 3650 days (about 10 years) and saved it to a file named "ca.crt".

8.3 Command 3: Creation of a directory for the CA

We created a directory to store the CA's files using the following command:

```
mkdir /path/to/ca
```

Replace "/path/to/ca"with the desired path for the directory.

8.4 Command 4: Moved the files to the directory

Then we moved the private key and certificate files to the directory with the following commands:

```
mv ca.key /path/to/ca
mv ca.crt /path/to/ca
```

8.5 Command 5: Setting the permissions

Finally, but not least important, we set up the appropriate permissions for the directory and files using the following commands:

```
chmod 700 /path/to/ca
chmod 400 /path/to/ca/ca.key
chmod 444 /path/to/ca/ca.crt
```

We did this to set the directory to be readable and executable only by the owner, set the private key file to be readable only by the owner, and set the certificate file to be readable by everyone, because of running and security reasons.

9 Issuing and Revoking X.509 Certificates

To issue and revoke the X.509 Certificates we followed these steps:

In the first step in issuing X.509 certificates, we created a private CA using OpenSSL. This involves generating a private key and a self-signed root certificate that will be used to sign the certificates issued by the CA.

Once we had set up the private CA, our next step was to issue X.509 certificates for the VPN gateway, VPN client, Apache web server, and clients connecting to the Apache web server. To do this, we generated a certificate signing request (CSR) for each entity requiring a certificate. The CSR contains information such as the entity's name, organization, and public key.

The private CA then uses its root certificate to sign the CSR and generate a new X.509 certificate for the entity.

X.509 certificates may need to be revoked if, for example, a private key is compromised or an employee leaves the company. To revoke a certificate, the private CA must maintain a Certificate Revocation List (CRL) that lists all revoked certificates. When a certificate is revoked, its serial number is added to the CRL. The CRL must be published and made available to all parties that rely on the certificates issued by the CA.

In addition to maintaining a CRL, the private CA should also support an Online Certificate Status Protocol (OCSP) responder. This enables entities such as the VPN gateway and Apache web server to check the status of a certificate in real time before allowing access. The OCSP responder receives a request containing the serial number of a certificate and responds with a status indicating whether the certificate is valid or revoked.

10 Tests performed to validate the functionalities implemented

To validate the functionalities implemented, we performed a series of tests that aimed to confirm the correct configuration and operation of the VPN tunnel, two-factor user authentication, PKI management, and OCSP verification.

10.1 VPN Tunnel Tests

For the VPN tunnel, we tested the connection between the remote client (road warrior) and the VPN gateway, verifying the mutual authentication using X.509 digital certificates, the enforcement of two-factor authentication (using username, password, and one-time password), and the verification of certificate revocation using OCSP. To do so, we attempted to establish a connection from the remote client to the VPN gateway using an invalid or revoked X.509 certificate, and we verified that the connection was refused by the gateway. We also checked that valid certificates and credentials allowed us to access hosts in the internal network, as expected.

10.2 2FA Tests

For the two-factor user authentication, we tested the enforcement of presenting a valid X.509 certificate and a one-time password (generated using Google Authenticator) to access the Apache web server via HTTPS. We also tested the verification of the certificate's validity using OCSP. To do so, we attempted to access the web server using an invalid or revoked certificate, or without presenting a valid one-time password, and we verified that the access was refused by the server. We also checked that valid certificates and credentials allowed us to access the web server, as expected.

10.3 PKI and OCSP Tests

For the PKI management and OCSP verification, we tested the creation and revocation of X.509 certificates using the private Certification Authority (CA) created with OpenSSL. We verified that new certificates were correctly issued and signed by the CA, and that revoked certificates were properly marked as such in the CA's Certificate Revocation List (CRL), as well as in the CA database and the index.txt file.

To test the revocation of client certificates, we used the following commands:

openssl ca -config openssl.cnf -revoke filename.crt openssl ca -config openssl.cnf -gencrl -out filename2.pem

The first command revokes the certificate for "client2"as specified by the filename.crt file. The second command generates a Certificate Revocation List (CRL) in the filename2.pem file and updates the CA database and the index.txt file to mark the revoked entity with an "R".

After running these commands, we verified that the revoked certificate was included in the CRL and that the CA database and the index.txt file were also updated to mark the revoked entity with an "R". This demonstrated that our certificate revocation process was working as expected and that the CA database and the index.txt file were being properly maintained.

As part of our security testing, we included a test to verify that the Online Certificate Status Protocol (OCSP) was working correctly. After revoking the certificate, we attempted to access the Apache web server using the revoked certificate. We expected that the OCSP would detect the revoked certificate and deny access to the server. Our test confirmed that the OCSP was working correctly, as access was denied and the server logs indicated that the certificate had been revoked.

This test provided us with confidence that our OCSP implementation was working as intended and that the security of the system was maintained even in the event of a revoked certificate.

We also tested the revocation of a certificate before accessing the VPN. We attempted to connect to the VPN using a revoked certificate. The VPN server correctly rejected the connection request due to the revoked status of the certificate, demonstrating that the Online Certificate Status Protocol (OCSP) was working as intended. This test confirms that our implementation of OCSP effectively prevents the use of compromised or invalid certificates for VPN access.

10.4 OCSP Server Verification

We also tested the functionality of the OCSP server by putting it down and verifying if the user was unable to authenticate. To perform this test, we set the OCSP cache timeout to 0 on the server. As expected, the user was unable to authenticate while the OCSP server was down.

10.5 Attempting to Access the Apache Server without Client Certificates and CA

We tested whether the Apache server was enforcing two-factor authentication by attempting to access the server without presenting a valid client certificate and CA. As expected, we were unable to access the server without these credentials.

10.6 Extra Tests

In addition to these tests, we also performed a connectivity test by pinging every interface from every other interface, including the VPN client, the VPN gateway, the OCSP, and the Apache web server. We verified that all interfaces were properly configured and connected and that there were no connectivity issues.

A test made was when a client is not connected to the VPN, they are outside the internal network established after the VPN. Therefore, they cannot directly access any resources within the network. When attempting to ping a resource within the VPN network, such as the Apache server, the client's request cannot reach the destination since it is outside of their network scope.

However, after the client connects to the VPN, they are effectively part of the VPN network and can now access resources within the internal network. As a result, the client can successfully ping the Apache server or any other resource that is part of the internal network. This is because the VPN server acts as an intermediary, allowing the client to access resources within the network as if they were directly connected to it, hence, the VPN tunnel.

Overall, the tests confirmed that the functionalities implemented were correctly configured and operating as expected, providing a secure and reliable communication channel for remote access to internal network services.

11 Configuration Files

11.1 OpenVPN Client File

The client file in this project will be responsible for configuring the OpenVPN client and generating a valid X.509 certificate, which is required for mutual authentication within the OpenVPN server. The client file will also be responsible for generating a one-time password using TOTP, which will be required for authentication with the OpenVPN server and the Apache web server.

remote 193.136.212.1: This sets the IP address of the server that the client will connect to.

port 1194: This sets the port number that the OpenVPN server is listening on.

proto udp4: This specifies the transport protocol used for communication between the client and server. In this case, UDP version 4 is used.

dev tun: This specifies the network device type that will be used for the VPN tunnel.

comp-lzo: This enables LZO compression for data transmitted over the VPN tunnel, which can improve performance.

persist-tu: This causes the tunnel to be kept open even if the client loses its connection to the server, which can improve stability.

persist-key: This causes the client's encryption keys to be kept in memory even if the client loses its connection to the server, which can improve stability.

nobind: This prevents the VPN client from binding to a specific local IP address and port number.

keepalive 10 60: This sends a keepalive packet to the server every 10 seconds, and will assume the connection is lost if no response is received after

60 seconds.

reneg-sec 10800: This specifies the time in seconds after which the client and server should renegotiate the encryption keys used for the VPN tunnel.

link-mtu 1500: This sets the maximum transmission unit (MTU) size for the VPN tunnel.

resolv-retry infinite: This tells the VPN client to keep retrying DNS resolution indefinitely if the initial attempt fails.

verb 5: This sets the verbosity level of the log output. Level 5 is very verbose and provides a lot of detail about the connection process.

status openvpn-status.log: This specifies the file to which OpenVPN will write status information about the VPN connection.

tls-version-min 1.2: This sets the minimum TLS version to be used for the connection.

tls-cipher TLS-DHE-RSA-WITH-AES-256-GCM-SHA384:TLS-DHE-RSA-WITH-AES-128-GCM-SHA256: This specifies the allowed cipher suites to use for the TLS handshake.

tls-auth ta.key 1: This specifies the location of the shared TLS authentication key file.

remote-cert-tls server: This tells the client to verify the server's certificate during the TLS handshake.

auth SHA256: This specifies the HMAC authentication algorithm to use for packet verification.

script-security 3: This sets the security level for scripts executed by OpenVPN.

ca ca.crt: This specifies the location of the certificate authority (CA) certificate file.

cert client.crt: This specifies the location of the client's certificate file.

key client.key: This specifies the location of the client's private key file.

auth-user-pass: This specifies that the client will use a username and password for authentication.

auth-nocache: This tells OpenVPN not to cache the authentication credentials, for added security.

11.2 OpenVPN Server File

The server file in this project will be responsible for configuring the OpenVPN server, Apache web server, and the Certification Authority (CA) using OpenSSL. The server file will also need to manage the PKI, including the creation of X.509 certificates and the revocation and OCSP verification of certificates. Additionally, the server file will need to enforce two-factor authentication for both the OpenVPN and Apache services, requiring valid X.509 certificates and a one-time password generated by TOTP (Time-based One-time Password Algorithm).

local 193.136.212.1: Sets the local IP address of the OpenVPN server.

port 1194: Specifies the port on which the OpenVPN server listens for incoming connections.

proto udp4: Configures the OpenVPN server to use the UDP protocol over IPv4.

dev tun: Specifies that the virtual network device used by OpenVPN should be a TUN device, which operates at the IP layer.

comp-lzo: Enables LZO compression on the data channel to improve performance.

reneg-sec 10800: Specifies the number of seconds after which a rekeying operation should be performed.

keepalive 10 60: Sends a keepalive packet every 10 seconds and waits 60 seconds for a response before timing out.

link-mtu 1500: Sets the maximum transmission unit (MTU) size of the TUN/TAP interface to 1500 bytes.

verb 5: Sets the verbosity level of the OpenVPN server's log output to 5 (the highest level of detail).

status openvpn-status.log: Specifies the file to which OpenVPN will write status information.

tls-version-min 1.2: Configures the OpenVPN server to use TLS version 1.2 or higher.

tls-cipher TLS-DHE-RSA-WITH-AES-256-GCM-SHA384:TLS-DHE-RSA-WITH-AES-128-GCM-SHA256: Specifies the TLS cipher suites that OpenVPN should support.

tls-verify checkOCSP.sh: Configures OpenVPN to verify the server certificate using an OCSP responder.

tls-auth ta.key 0: Enables the use of a shared secret key for additional data channel encryption.

auth SHA256: Specifies the hash algorithm used for data channel HMAC authentication.

script-security 3: Sets the level of security for executing scripts to 3 (the highest level).

ca ca.crt: Specifies the CA certificate used to verify client and server certificates.

cert server.crt: Specifies the server certificate used by the OpenVPN server.

key server.key: Specifies the private key associated with the server certificate.

dh dh2048.pem: Specifies the Diffie-Hellman (DH) parameters used to perform a key exchange.

plugin /usr/lib64/openvpn/plugins/openvpn-plugin-auth-pam.so openvpn: Specifies the PAM plugin to use for authentication.

verify-client-cert require: Enables client certificate authentication and requires that clients provide a valid certificate.

push "route 10.60.0.0 255.255.255.0": Configures the OpenVPN server to push a route for the 10.60.0.0/24 subnet to connecting clients.

server 10.8.0.0 255.255.255.0: Specifies the virtual IP address pool used by the OpenVPN server to assign IP addresses to clients.

ifconfig-pool-persist ipp.txt: Saves the virtual IP address assignments in the ipp.txt file for persistent use across server restarts.

max-clients 10: Limits the maximum number of clients that can connect to the OpenVPN server at the same time to 10.

11.3 PAM OPENVPN File

The PAM (Pluggable Authentication Module) file in this project will be responsible for configuring the authentication process for the Apache web server. It will enforce two-factor authentication, requiring a valid X.509 certificate and a one-time password generated by TOTP.

auth [user_unknown=ignore success=ok ignore=ignore default=bad] /usr/lib64/security/pam_securetty.so: This line specifies that the pam_securetty.so module should be used for authentication. The [user_unknown=ignore success=ok ignore=ignore default=bad] part specifies how to handle different types of users, with the default behavior being to reject authentication attempts by unknown users.

auth required /usr/lib64/security/pam_google_authenticator.so forward_pass: This line specifies that the pam_google_authenticator.so module should be used for authentication and that if the user successfully authenticates with this module, the authentication process should continue to the next module (forward_pass). This module provides two-factor authentication using the Google Authenticator app.

auth include system-auth: This line includes the system-auth file, which contains a set of common authentication rules.

account include system-auth: This line includes the system-auth file, which contains a set of common account management rules.

password include system-auth: This line includes the system-auth file, which contains a set of common password management rules.

11.4 PAM HTTPD File

This PAM file enforces multi-factor authentication via Google Authenticator and includes rules for managing passwords and sessions. It is a security measure intended to ensure that only authorized users can access the Apache service.

auth required pam_google_authenticator.so nullok: This line specifies that authentication via Google Authenticator is required for access to the Apache service. The nullok parameter allows users who have not set up Google Authenticator to still access the service.

auth include password-auth: This line includes the password-auth file, which contains authentication rules related to passwords.

auth required pam_deny.so: This line denies access if none of the previous authentication rules were successful.

account include password-auth: This line includes the password-auth file, which contains account management rules related to passwords.

session optional pam_keyinit.so revoke: This line initializes a user's keyring and ensures that any previously revoked keys are removed. The optional parameter means that this step is not required for a successful login.

session include password-auth: This line includes the password-auth file, which contains session management rules related to passwords.

11.5 Apache Server File

<IfModule mod_ssl.c>: This line indicates that the following configuration directives should only be used if the mod_ssl module is available.

AddDefaultCharset utf-8: This directive sets the default character set for all resources served by this virtual host to UTF-8.

SSLStaplingCache "shmcb:/var/run/ocsp(1500000)": This directive configures the SSL stapling cache, which is used to store and retrieve OCSP responses for SSL certificates. In this case, the cache is stored in shared memory using the shmcb method, and has a size of 1500000 bytes.

<VirtualHost *:443>: This directive begins the virtual host configuration for all requests on port 443.

ServerName 'apacheserver': This directive sets the server name for this virtual host to 'apacheserver'.

SSLCertificateFile '/etc/pki/CA/certs/apache.crt': This directive sets the path to the SSL certificate file for this virtual host.

SSLCertificateKeyFile '/etc/pki/CA/private/apache.key': This directive sets the path to the SSL private key file for this virtual host.

SSLCACertificateFile /etc/pki/CA/certs/ca.crt: This directive sets the path to the SSL CA certificate file for this virtual host.

SSLEngine on: This directive enables SSL for this virtual host.

SSLUseStapling on: This directive enables OCSP stapling for this virtual host.

SSLStaplingResponseMaxAge 900: This directive sets the maximum age for OCSP responses to 900 seconds.

SSLStaplingResponderTimeout 5: This directive sets the timeout for the OCSP responder to 5 seconds.

SSLStaplingReturnResponderErrors on: This directive instructs the server to return any errors encountered when querying the OCSP responder to the client.

SSLStaplingStandardCacheTimeout 3600: This directive sets the cache timeout for OCSP responses to 3600 seconds.

SSLStaplingForceURL http://10.60.0.253:4444/: This directive forces the server to use a specific OCSP responder URL, even if it is not specified in the SSL certificate.

<**Location "/»**: This directive begins the configuration for the root location of this virtual host.

AuthType Basic: This directive sets the authentication type to basic.

AuthBasicProvider PAM: This directive sets the authentication provider to PAM.

AuthName "Google Auth": This directive sets the authentication realm name to 'Google Auth'.

Require valid-user: This directive requires authentication for all users.

AuthPAMService google-authenticator: This directive specifies the PAM service name to be used for authentication.

12 PGP Encryption Delivery

In this section it's demonstrated how we encrypted the files and the report with the PGP key.

12.1 Step 1: Generate a GPG key:

gpg –gen-key

12.2 Step 2:

Follow the prompts to set up the key. We made sure to use a strong passphrase and saved the key pair to a secure location.

12.3 Step 3: Export the public key:

gpg –export –armor <your-email-address> > my-public-key.asc

12.4 Step 4:

Sent the public key to the recipient(s) who will be receiving the encrypted and signed files.

12.5 Step 5: Sign the file:

gpg –sign-files <filename>

12.6 Step 6: Import the recipient key:

gpg –import recipient_public_key.asc

12.7 Step 7: Encrypt the file:

gpg –recipient <recipient_email_address> –encrypt <filename>

12.8 Step 8: Send the file

Attach the encrypted and/or signed file to your email and send it to the recipient(s).

13 Conclusion

In conclusion, this report detailed the implementation and testing of a secure VPN infrastructure using OpenVPN and a Public Key Infrastructure (PKI) with X.509 certificates. The implementation involved setting up the OpenVPN server and configuring it to enforce two-factor authentication using X.509 digital certificates and one-time passwords. A private CA was created with OpenSSL to issue and manage X.509 certificates for the VPN gateway, VPN client, and Apache web server. To ensure security, appropriate permissions were set for the directory and files. Additionally, a series of tests were conducted to validate the correct configuration and operation of the VPN tunnel, two-factor user authentication, PKI management, and OCSP verification. The tests demonstrated that the VPN infrastructure was secure and functioned as expected. Overall, this implementation provides a strong and reliable VPN infrastructure suitable for organizations looking for secure remote access to their resources.

Future work could involve the integration of additional security measures to further enhance the security of the VPN infrastructure. For example, the implementation of a firewall and intrusion detection system could help prevent unauthorized access and attacks. Additionally, regular security audits could be conducted to ensure that the VPN infrastructure remains secure over time. Finally, the implementation could be scaled to support larger user bases or to enable remote access to additional resources. Overall, continued effort to improve and maintain the security of the VPN infrastructure will help to ensure its ongoing effectiveness and reliability.