

Projeto 1 - STI

Giovanni Maffeo de Medeiros: 2024232210
Diogo Pinto: 2024156583

Source code: GitHub Repository

Contents

Contents	1	
1	Objetivo	2
2	Certificados	3
2.1	Criação da CA	3
2.1.1	Objetivo	3
2.1.2	Execução	3
2.2	Emissão dos Certificados	4
2.2.1	Objetivo	4
2.2.2	Execução	5
2.2.3	Testes	6
3	OCSP	7
3.1	Criar Servidor OCSP	7
3.1.1	Objetivo	7
3.1.2	Execução	7
3.1.3	Testes	8
4	VPN	8
4.1	Autenticação Mútua	8
4.1.1	Objetivo	8
4.1.2	Execução	9
4.1.3	Testes	11
4.2	OTP	12
4.2.1	Objetivo	12

4.2.2	Execução	12
4.2.3	Testes	13
5	Apache	14
5.1	Autenticação Mútua	14
5.1.1	Objetivo	14
5.1.2	Execução	14
5.1.3	Testes	16
5.2	OTP	17
5.2.1	Objetivo	17
5.2.2	Execução	17
5.2.3	Testes	18
6	OCSP	18
6.1	Objetivo	18
6.2	Execução	19
6.3	Testes	20
7	Envio PGP	20
8	Reconhecimento	21
9	Conclusão	21

1 Objetivo

O projeto consiste na configuração de uma VPN no modelo road warrior, garantindo comunicações seguras entre clientes remotos e um gateway VPN. A segurança será reforçada através da autenticação mútua com certificados X.509, validação via OCSP e autenticação multifator (OTP). Além da VPN, um servidor Apache será configurado para exigir autenticação baseada em certificados, OTP e verificadas por OCSP. A infraestrutura de segurança será gerida por uma Autoridade Certificadora (CA) privada, responsável pela emissão e revogação de certificados.

2 Certificados

2.1 Criação da CA

2.1.1 *Objetivo*

O objetivo desta etapa é criar uma Autoridade Certificadora (Certificate Authority – CA) responsável por emitir os certificados X.509 para cada um dos componentes envolvidos na comunicação do projeto. Esses certificados garantem a confidencialidade, autenticidade e integridade das informações transmitidas.

É fundamental criar a CA, pois ela atua como a entidade confiável que assina e validará os certificados utilizados pelos componentes do sistema. Dessa forma, os componentes podem se autenticar mutuamente e estabelecer comunicações seguras, criptografando e assinando os dados com base em suas respectivas chaves.

2.1.2 *Execução*

Para isso, instalamos o OpenSSL. Caso não esteja instalada por padrão, executamos:

```
sudo apt update && sudo apt install openssl -y
```

Criámos um diretório para armazenar os arquivos da CA:

```
mkdir demoCA  
cd demoCA
```

Criámos um diretório para armazenar as informações necessárias para emitir novos certificados:

```
mkdir newcerts  
cd newcerts
```

Criámos o ficheiro de "índice de certificados". A CA utilizará para rastrear os certificados emitidos e suas respectivas revogações, isto é, um registro daqueles que estão ativos e revogados.

```
touch index.txt
```

Cada certificado emitido pela CA precisa ter um identificador único. Assim, criámos um arquivo para armazenar o "número de série" dos certificados. Esse arquivo mantém o próximo número de série a ser usado, logo, precisamos inicializá-lo com o número "01".

```
nano serial  
Escrever "01"  
Salvar
```

Navegadores e clientes modernos exigem que os certificados de uma CA incluam extensões específicas para serem corretamente reconhecidos como certificados raiz confiáveis. Essas extensões são definidas em um ficheiro, utilizado durante a criação do certificado da CA.

```
nano v3_ca.ext  
Escrever "keyUsage = cRLSign, digitalSignature, keyCertSign"  
Escrever "basicConstraints=critical,CA:true,pathlen:0"  
Salvar
```

Geramos a chave privada da CA, especificando o arquivo onde será armazenada e utilizando o algoritmo DES3 para protegê-la com uma senha.

```
cd ..  
openssl genrsa -out ca.key -des3
```

Geramos o Pedido de Assinatura de Certificado (CSR - Certificate Signing Request), indicando a chave privada da CA e o ficheiro onde armazenará a CSR. Esse ficheiro contém as informações da CA, incluindo sua chave pública e os detalhes necessários para a criação do certificado.

```
openssl req -new -key ca.key -out ca.csr
```

Criámos o certificado X.509 da CA, definindo o período de validade, o CSR da CA gerado, o arquivo onde o certificado será armazenado, a chave privada da CA para assinatura e o ficheiro de extensão, que será utilizado na emissão dos certificados pela CA.

```
openssl x509 -req -days 365 -in ca.csr -out ca.crt -signkey  
ca.key -extfile v3_ca.ext
```

2.2 Emissão dos Certificados

2.2.1 Objetivo

O objetivo desta etapa é emitir um certificado digital para cada componente envolvido na comunicação do sistema, garantindo que sejam reconhecidos e autorizados pela Autoridade Certificadora (CA). Esses certificados permitem que os componentes se autentiquem de forma confiável e estabeleçam comunicações seguras por meio da criptografia baseada em suas respectivas chaves.

Para assegurar a integridade e a confidencialidade dos dados transmitidos, cada componente receberá um certificado único, assinado pela CA, que permitirá sua autenticação dentro do ambiente seguro. Dessa forma, serão emitidos certificados para os seguintes componentes:

- **VPN:**
 - Servidor VPN
 - Cliente VPN
- **Apache:**
 - Servidor Apache
 - Cliente Apache
- **OCSP:**
 - Servidor OCSP

2.2.2 Execução

Para a emissão de cada certificado, os comandos seguem um padrão semelhante, alterando apenas as referências para cada componente. A seguir, detalhamos os passos necessários:

1. **Emissão da chave privada única** Geramos a chave privada do cliente, utilizando o algoritmo DES3 para protegê-la com uma senha. O arquivo resultante armazenará a chave privada.
2. **Emissão do CSR (Certificate Signing Request)** Criámos um CSR contendo as informações necessárias para a geração do certificado. Este arquivo incluirá os detalhes de identificação do cliente e será assinado com a chave privada gerada no passo anterior.
3. **Criação do certificado X.509** Finalmente, geramos o certificado X.509 para o cliente, especificando o período de validade, o CSR correspondente, o arquivo onde o certificado será armazenado, a chave privada da CA para assinatura e o ficheiro de extensão necessário para a emissão dos certificados.

Por fim, comandos para emissão dos certificados para cada componente são:

```

# Servidor VPN
openssl genrsa -out server_vpn.key -des3
openssl req -new -key server_vpn.key -out server_vpn.csr -
    subj "/C=PT/ST=Coimbra/L=Coimbra/O=UC/OU=DEI/CN=server_vpn
    /emailAddress=uc2024156583@student.uc.pt" -passin pass:
    test
sudo openssl ca -in server_vpn.csr -cert ca.crt -keyfile ca.
    key -out server_vpn.crt -extfile v3.ext
# Cliente VPN
openssl genrsa -out client_vpn.key -des3
openssl req -new -key client_vpn.key -out client_vpn.csr -
    subj "/C=PT/ST=Coimbra/L=Coimbra/O=UC/OU=DEI/CN=client_vpn
    /emailAddress=uc2024156583@student.uc.pt" -passin pass:
    test
sudo openssl ca -in client_vpn.csr -cert ca.crt -keyfile ca.
    key -out client_vpn.crt -extfile v3.ext
# Servidor Apache
openssl genrsa -out server_apache.key -des3
openssl req -new -key server_apache.key -out server_apache.
    csr -subj "/C=PT/ST=Coimbra/L=Coimbra/O=UC/OU=DEI/CN=
    server_apache/emailAddress=uc2024156583@student.uc.pt" -
    passin pass:test
sudo openssl ca -in server_apache.csr -cert ca.crt -keyfile
    ca.key -out server_apache.crt -extfile v3.ext
# Cliente Apache
openssl genrsa -out client_apache.key -des3
openssl req -new -key client_apache.key -out client_apache.
    csr -subj "/C=PT/ST=Coimbra/L=Coimbra/O=UC/OU=DEI/CN=
    client_apache/emailAddress=uc2024156583@student.uc.pt" -
    passin pass:test
sudo openssl ca -in client_apache.csr -cert ca.crt -keyfile
    ca.key -out client_apache.crt -extfile v3.ext
# Servidor OCSP
openssl genrsa -out ocsp.key -des3
openssl req -new -key ocsp.key -out ocsp.csr -subj "/C=PT/ST=
    Coimbra/L=Coimbra/O=UC/OU=DEI/CN=ocsp/emailAddress=
    uc2024156583@student.uc.pt" -passin pass:test
sudo openssl ca -in ocsp.csr -cert ca.crt -keyfile ca.key -
    out ocsp.crt -extfile v3.ext

```

2.2.3 Testes

Para garantir que os certificados foram corretamente emitidos e assinados pela CA, devemos verificar se eles podem ser validados utilizando o certificado da CA. Para isso, o retorno do comando abaixo deve ser:

```
+--(kali㉿kali)-[~/Desktop/Seguranca/Project]
|__ $ openssl verify -CAfile ca.crt server_vpn.crt
server_vpn.crt: OK
```

3 OCSP

3.1 Criar Servidor OCSP

3.1.1 *Objetivo*

Nesta etapa, foi configurado um servidor OCSP (Online Certificate Status Protocol) responsável por verificar a validade dos certificados utilizados na comunicação segura, permitindo que os clientes desse serviço consultem, em tempo real, o status de um certificado.

Esse servidor será integrado aos serviços OpenVPN e Apache, garantindo que esses componentes rejeitem conexões de clientes com certificados expirados, não verificados ou revogados pela CA, assegurando uma comunicação segura nesses serviços.

3.1.2 *Execução*

Começamos por lançar o servidor que vai receber os pedidos:

```
openssl ocsp -index CA/index.txt -port 8081 -rsigner ocsp.crt
              -rkey ocsp.key -CA ca.crt -text -out log_OCSP.txt
```

Criámos o script que vai verificar a autenticidade dos certificados:

```
#!/bin/bash

OCSP_SERVER="http://192.168.1.76:8081"
CERT=$1
CA_CERT="ca.crt"

if [ ! -f "$CA_CERT" ]; then
    echo "ERROR: CA certificate not found at $CA_CERT"
    exit 1
fi

openssl ocsp -issuer "$CA_CERT" -serial "$CERT" -url "
$OCSP_SERVER" -noverify
```

3.1.3 Testes

Para os testes se o servidor OCSP funciona da forma esperada para verificar o status de um certificado, precisaremos iniciar o servidor OCSP.

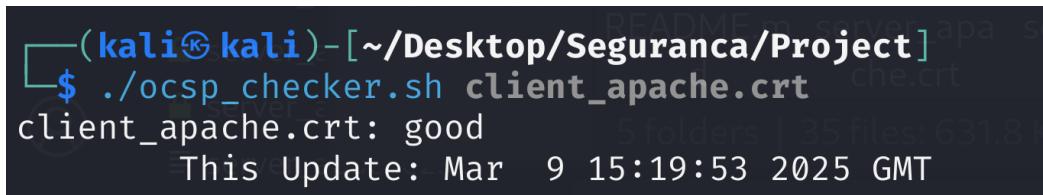
```
openssl ocsp -index CA/index.txt -port 8081 -rsigner ocsp.crt  
-rkey ocsp.key -CA ca.crt -text -out log_OCSP.txt
```

Depois, podemos listar o status de cada certificado através dos comandos:

```
cd demoCA  
cat index.txt  
cd ..
```

Se a primeira coluna indica "V", isso significa que o certificado está válido. Dessa forma, ao verificar um dos certificados através do servidor, o retorno deve ser o mesmo abaixo:

```
openssl ocsp -CAfile demoCA/ca.crt -issuer demoCA/ca.crt -  
cert demoCA/client_vpn.crt -url http://localhost:8081
```



```
(kali㉿kali)-[~/Desktop/Segurança/Project]  
$ ./ocsp_checker.sh client_apache.crt  
client_apache.crt: good  
This Update: Mar 9 15:19:53 2025 GMT
```

Figure 1: Verificação através do Servidor OCSP

4 VPN

4.1 Autenticação Mútua

4.1.1 Objetivo

O objetivo desta etapa é permitir a conexão do VPN Client (road warrior) ao VPN Server (VPN Gateway). Isso é necessário para garantir que o acesso do road warrior aos componentes da Virtual Machine, como o Apache, só seja permitido caso ele esteja conectado à VPN.

Para isso, esta etapa será implementada com duas camadas de segurança. A primeira será a autenticação mútua (Mutual Authentication) entre cliente e servidor VPN, utilizando certificados X.509. A segunda, detalhada na seção 4.2, será baseada na autenticação por One-Time Passwords (OTP).

4.1.2 Execução

Começamos por colocar a placa de rede da nossa VM do Kali Linux com bridged adapter para permitir a conexão entre o Windows e o Kali.

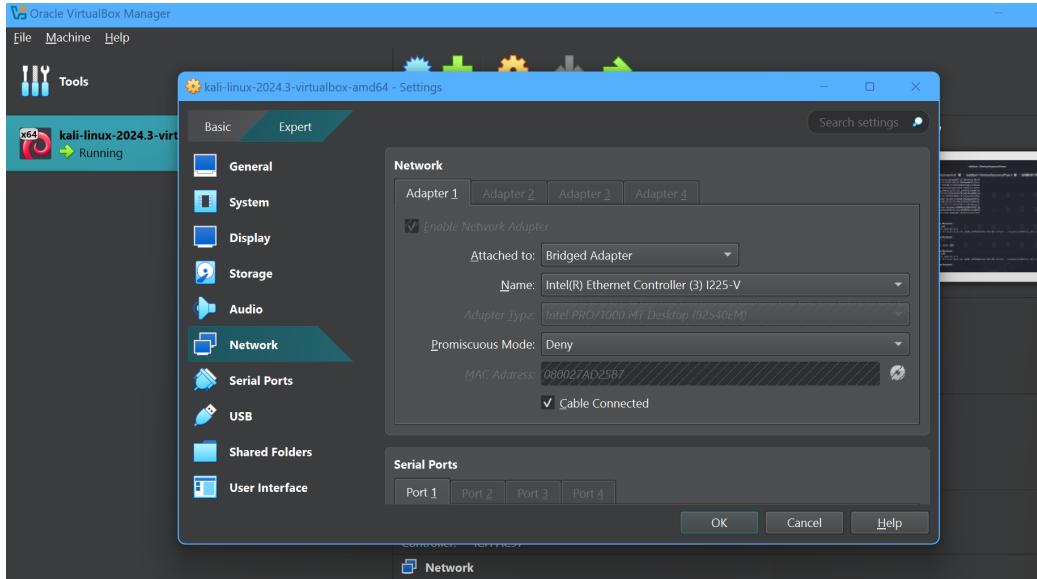


Figure 2: Alteração Interface de Rede

Instalar a ferramenta OpenVPN que será utilizada para a inicialização e configuração do VPN Gateway.

```
sudo apt update && sudo apt install openvpn -y
```

Gerar os parâmetros Diffie-Hellman (DH) utilizados na negociação segura de chaves entre o cliente e o servidor VPN, garantindo um canal criptografado. O comando especifica o ficheiro onde será armazenado e o tamanho da chave.

```
openssl dhparam -out dh.pem 2048
```

Executamos o seguinte comando na VM para verificar o IP, pois será utilizado como o IP do servidor VPN. Vale ressaltar que, como utilizamos o Windows como cliente VPN, o IP utilizado é o da Rede Local.

```
ip -4 addr show | grep "inet " | awk '{print $2}'
```

De seguida como já tínhamos a openvpn instalada, nós desenvolvemos uma configuração personalizada para o servidor, como fizemos em aula. Vale ressaltar que O keepalive foi adicionado para a conexão não ir abaixo se o cliente não estiver a mandar pacotes durante muito tempo, forçando o servidor a enviar de 10 em 10 segundos um ping. Além disso, adicionamos uma

linha para executar o nosso custom script para a validação dos certificados através do OCSP server:

```
local 192.168.1.76 # IP do servidor VPN
port 1194 # Porta de escuta da VPN
proto udp # Protocolo
dev tun # Interface TUN (VPN baseada em IP)

ca ../ca.crt # Certificado da CA
cert ../server_vpn.crt # Certificado do servidor VPN
key ../server_vpn.key # Chave privada do servidor
dh dh.pem # Diffie-Hellman

server 10.8.0.0 255.255.255.0 # Sub-rede da VPN

# Explicado posteriormente
plugin /usr/lib/openvpn/openvpn-plugin-auth-pam.so openvpn

keepalive 10 60

# OCSP Server
tls-server
tls-verify ../ocsp_checker.sh
script-security 2
```

Para lançar o servidor usamos o comando, passando a configuração personalizada como parâmetro:

```
sudo openvpn --config ./server.conf
```

De seguida, criámos o ficheiro para o cliente da seguinte forma:

```
client # Define como cliente OpenVPN
dev tun # Interface TUN (VPN baseada em IP)
proto udp # Protocolo
remote 192.168.1.76 1194 # IP e porta do servidor VPN

persist-tun # Mantem a interface TUN ativa apesar de reconexão
persist-key # Mantem a chave privada carregada apesar de reconexão

ca ../ca.crt # Certificado da CA
cert ../client_vpn.crt # Certificado do cliente
key ../client_vpn.key # Chave privada do cliente

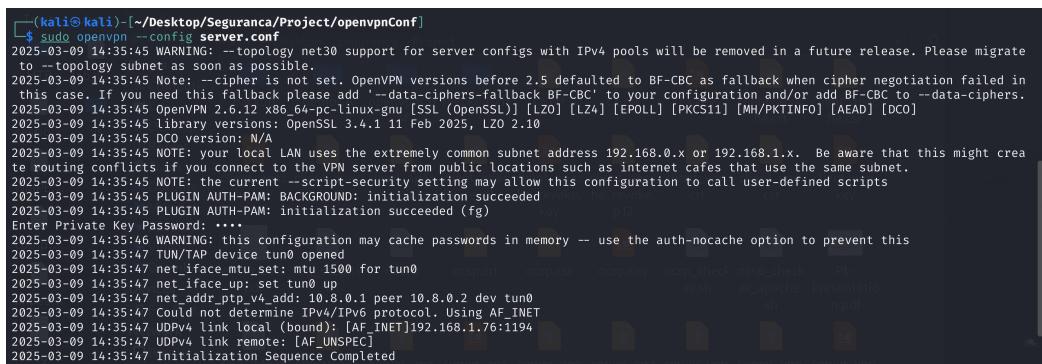
# Explicado posteriormente
auth-user-pass
```

Por fim, é importante que esse ficheiro seja salvo com a referência "client.ovpn", de modo a testar a conexão do Windows usando a aplicação do OpenVPN.

4.1.3 Testes

Instalar a aplicação do OpenVPN no Windows, copiar os ficheiros relacionados aos certificados (ca.crt, client_vpn.crt, client_vpn.key) e a configuração (client.ovpn) do Cliente VPN para um diretório do Windows e importar o ficheiro de configuração na aplicação. Em seguida, inicializar o servidor VPN na VM:

```
sudo openvpn --config ./server.conf
```



```
(kali㉿kali)-[~/Desktop/Segurança/Project/openvpnConf]
$ sudo openvpn --config server.conf
2025-03-09 14:35:45 WARNING: --topology net30 support for server configs with IPv4 pools will be removed in a future release. Please migrate to --topology subnet as soon as possible.
2025-03-09 14:35:45 Note: --cipher is not set. OpenVPN versions before 2.5 defaulted to BF-CBC as fallback when cipher negotiation failed in this case. If you need this fallback please add '--data-ciphers-fallback BF-CBC' to your configuration and/or add BF-CBC to --data-ciphers.
2025-03-09 14:35:45 OpenVPN 2.6.12 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] [DCO]
2025-03-09 14:35:45 library versions: OpenSSL 3.4.1 11 Feb 2025, LZO 2.10
2025-03-09 14:35:45 DCO version: N/A
2025-03-09 14:35:45 NOTE: your local LAN uses the extremely common subnet address 192.168.0.x or 192.168.1.x. Be aware that this might create routing conflicts if you connect to the VPN server from public locations such as Internet cafes that use the same subnet.
2025-03-09 14:35:45 NOTE: the current script-security setting may allow this configuration to call user-defined scripts
2025-03-09 14:35:45 PLUGINS AUTH-PAM: BACKGROUND: initialization succeeded
2025-03-09 14:35:45 PLUGINS AUTH-PAM: initialization succeeded (fg)
Enter Private Key Password: ****
2025-03-09 14:35:46 WARNING: this configuration may cache passwords in memory -- use the auth-nocache option to prevent this
2025-03-09 14:35:47 TUN/TAP device tun0 opened
2025-03-09 14:35:47 net_iface_mtu_set: mtu 1500 for tun0
2025-03-09 14:35:47 net_iface_up: set tun0 up
2025-03-09 14:35:47 net_addr_ptp_v4_add: 10.8.0.1 peer 10.8.0.2 dev tun0
2025-03-09 14:35:47 Could not determine IPv4/IPv6 protocol. Using AF_INET
2025-03-09 14:35:47 UDPv4 link local (bound): [AF_INET]192.168.1.76:1194
2025-03-09 14:35:47 UDPv4 link remote: [AF_UNSPEC]
2025-03-09 14:35:47 Initialization Sequence Completed
```

Figure 3: Inicialização do Servidor VPN

Para transferir os ficheiros necessários para os testes do Kali para o Windows começamos por abrir um simples servidor temporário usando:

```
python3 -m http.server 6060
```

E do lado do cliente acedemos através do IP da máquina e do porto 6060 no browser para transferir os ficheiros.

Por fim, conectar-se ao servidor VPN utilizando a aplicação do Windows. Vale ressaltar que será necessário preencher com a private key utilizada no seu certificado:

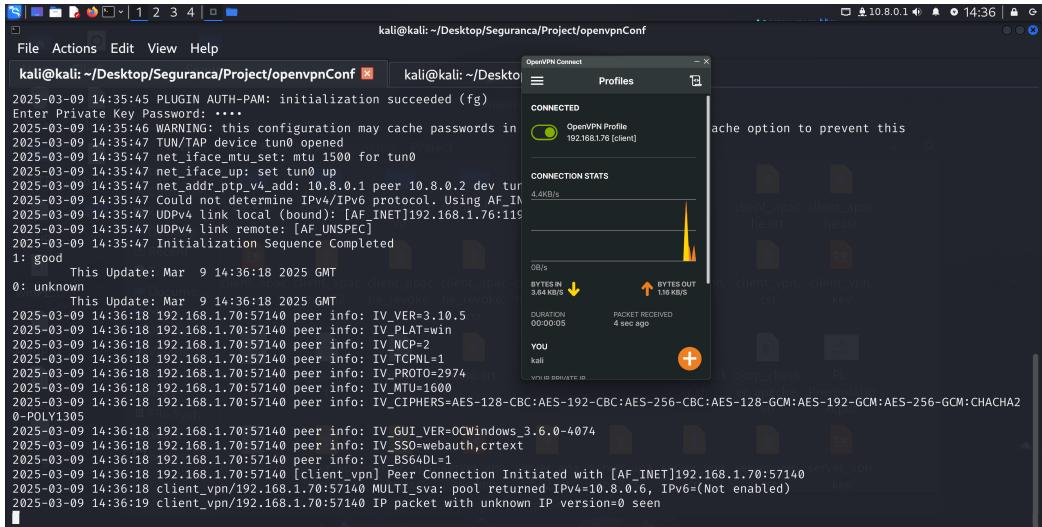


Figure 4: Conexão do Cliente VPN

4.2 OTP

4.2.1 Objetivo

O objetivo desta etapa é reforçar a segurança do acesso à VPN, adicionando uma camada de autenticação adicional. Dessa forma, além da necessidade de certificados válidos, será exigido um mecanismo de autenticação baseado no Google Authenticator, que gera senhas temporárias de uso único (OTP).

4.2.2 Execução

Começamos por sincronizar o tempo usando:

```
sudo timedatectl set-timezone Europe/Lisbon
sudo timedatectl set-ntp on
sudo systemctl restart systemd-timesyncd
sudo ntpdate -u pool.ntp.org
```

E instalamos o google-authenticator:

```
sudo apt install libpam-google-authenticator
```

De seguida geramos um chave OTP dando scan do QR code gerado ao executar o comando:

```
google-authenticator
```

Adicionamos a seguinte linha no ficheiro /etc/pam.d/openvpn:

```
auth required pam_google_authenticator.so
```

E adicionamos a seguinte linha no arquivo de configuração do Servidor VPN (server.conf):

```
plugin /usr/lib/openvpn/openvpn-plugin-auth-pam.so openvpn
```

E adicionamos a seguinte linha no arquivo de configuração do Cliente VPN (client.ovpn):

```
auth - user - pass
```

4.2.3 Testes

Para inicializar, podemos seguir os mesmos passos indicados em 4.1.3. Assim, agora ao exceder o Cliente VPN, será necessário passar o código OTP além da private key utilizada no seu certificado:

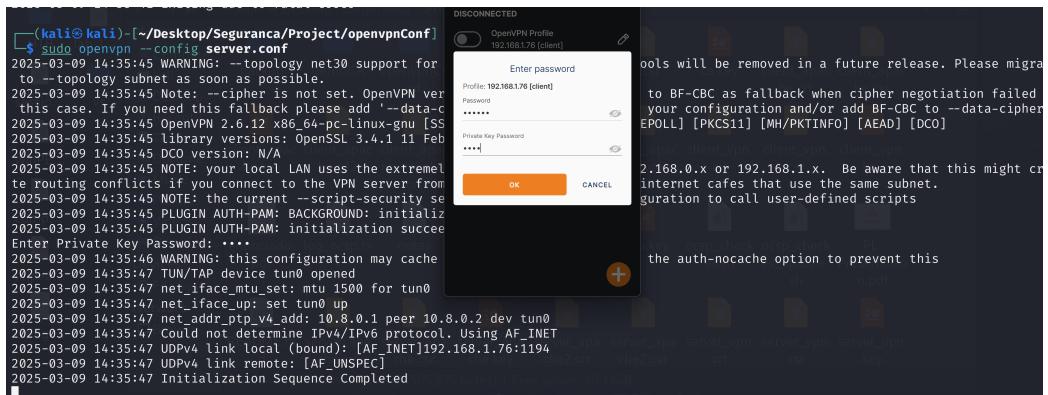


Figure 5: Conexão do Cliente VPN com OTP

Vale ressaltar que o código OTP é exibido no aplicativo Google Authenticator através smartphone utilizado para escanear o QR Code no 4.2.2:

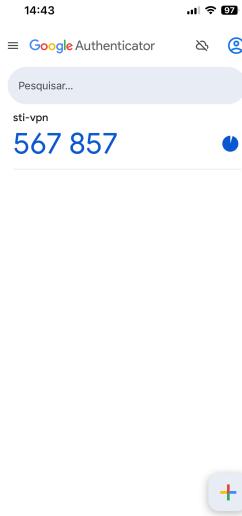


Figure 6: Exemplo código OTP na Aplicação

Para garantir que a conectividade do OpenVPN estava funcionando corretamente e que apenas credenciais válidas eram aceitas, realizamos testes analisando tanto as mensagens exibidas pelo cliente OpenVPN quanto os logs gerados no servidor, testando as credenciais corretas e incorretas.

5 Apache

5.1 Autenticação Mútua

5.1.1 *Objetivo*

O objetivo nesta etapa é configurar um Servidor Apache que apenas será acessível pelo Cliente VPN se a conexão VPN estiver estabelecida. Para isso, utilizamos a VM como Servidor Apache e o Windows como Cliente VPN.

Assim como a VPN, configuraremos duas partes: Autenticação Mútua e OTP (datalhado em 5.2).

5.1.2 *Execução*

O primeiro passo é a instalação do Apache na VM, caso já não esteja instalado

```
sudo apt install apache2 -y
```

Em seguida, configuramos o Apache para operar em um dos IPs atribuídos pelo servidor VPN, como estabelecido no arquivo de configuração da vpn "server.config":

```
server 10.8.0.0 255.255.255.0
```

Para isso, hospedamos o Apache no IP "10.8.0.1". Nesse caso, precisaremos editar os seguintes ficheiros de configuração. Primeiro, o ficheiro "/etc/apache2/ports.conf":

```
Listen 10.8.0.1:80

<IfModule ssl_module>
    Listen 10.8.0.1:443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 10.8.0.1:443
</IfModule>
```

Depois, o arquivo "/etc/apache2/sites-available/000-default.conf" da seguinte forma:

```
<VirtualHost 10.8.0.1:80>
...
```

Por fim, alteramos o arquivo "/etc/apache2/sites-available/default-ssl.conf" da seguinte forma:

```
<VirtualHost 10.8.0.1:443>
...
```

Depois, configuramos o Apache para exigir a autenticação através dos certificados alterando o arquivo "/etc/apache2/sites-available/default-ssl.conf" da seguinte forma:

```
...
SSLEngine on
ServerName "www.security.uc.pt"
SSLCertificateFile /home/kali/Desktop/Seguranca/Project/
    server_apache2.crt
SSLCertificateKeyFile /home/kali/Desktop/Seguranca/Project/
    server_apache.key
SSLVerifyClient require
SSLVerifyDepth 3
...
```

De seguida, ativamos os seguintes módulos:

```
authnz_pam_module
```

```
socache_shmcb_module  
ssl_module
```

Por fim, reiniciamos o Servidor Apache para aplicar as alterações:

```
sudo systemctl restart apache2
```

5.1.3 Testes

Para os testes, temos que nos conectar o cliente (Windows) ao servidor (VM), pois, sem isso, não será possível conectarmos ao Apache. Isso porque o IP apenas é acessível se a VPN estiver conectada:

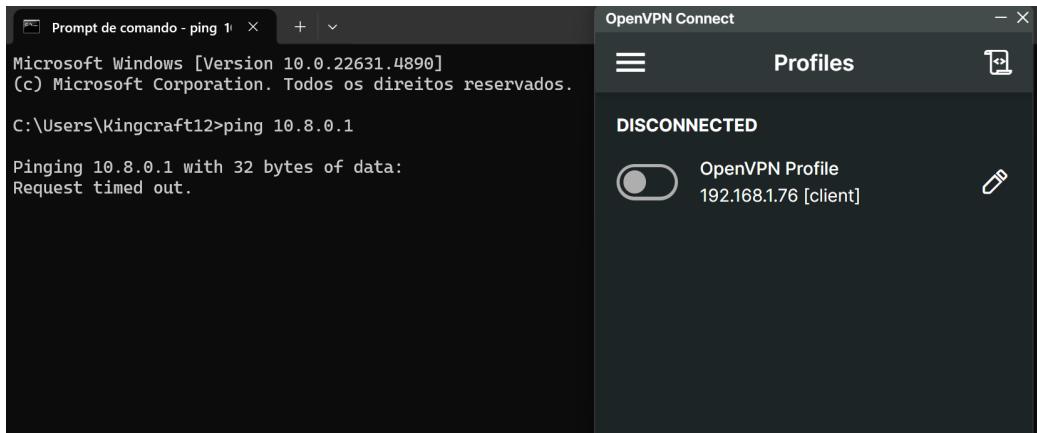


Figure 7: Apache inacessível sem VPN

Em seguida, como o "ServerName" é "www.security.uc.pt", precisamos alterar isso no arquivo de hosts do cliente (Windows), que sobrepõe o DNS. Assim, alteramos o arquivo "C:/Windows/System32/drivers/etc/hosts":

```
10.8.0.1 www.security.uc.pt
```

Por fim, importamos o certificado da CA e o do Cliente Apache no browser, adicionando na aba "Autoridades de certificação de raiz fidedigna" porque nas outras abas o google chrome diz que não confia no certificado:

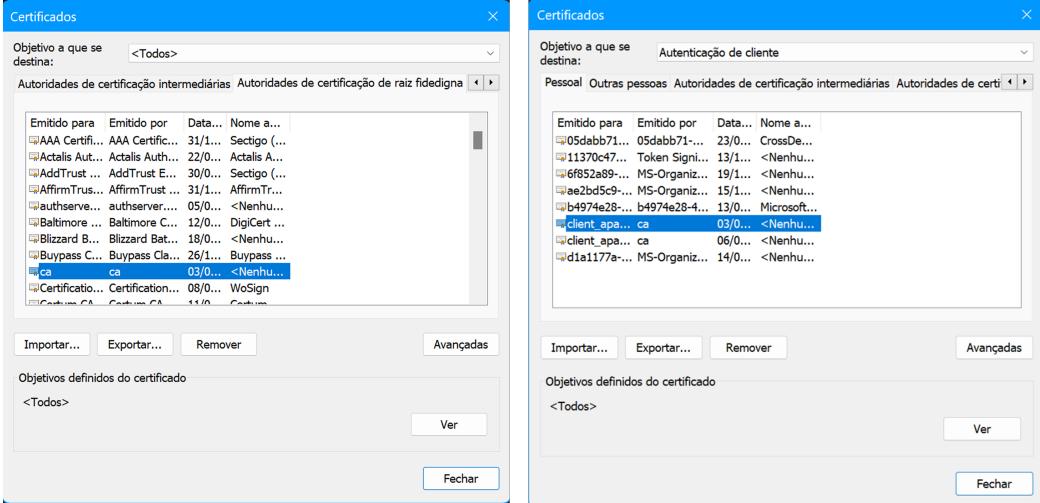


Figure 8: Importação certificados Google Chrome

E conseguimos estabelecer a conexão de forma segura:



Figure 9: Importação certificados Google Chrome

5.2 OTP

5.2.1 *Objetivo*

O objetivo dessa segunda fase é configurar a segunda camada de segurança para o acesso ao Apache, através do OTP.

5.2.2 *Execução*

O primeiro passo será alterar o ficheiro "/etc/apache2/sites-available/default-ssl.conf" para configurar o acesso apenas por OTP:

```

<Location />
    AuthType Basic
    AuthName "OTP Authentication"
    AuthBasicProvider PAM
    AuthPAMService apache2
    Require valid-user
    AuthBasicAuthoritative Off
</Location>

```

Em seguida, como o Apache é executado por padrão com o usuário "www-data". Como esse usuário não tem permissão para acessar o ficheiro .google_authenticator, que armazena as configurações de autenticação OTP, é necessário alterar o usuário com que o Apache é executado para o mesmo que configurou o OTP na etapa 4.2.2. Isso é feito, alterando o ficheiro "/etc/apache2/apache2.conf":

```

User ${APACHE_RUN_USER} --> User kali
Group ${APACHE_RUN_GROUP} --> Group kali

```

Por fim, reiniciamos o Servidor Apache para aplicar as alterações:

```

sudo systemctl restart apache2

```

5.2.3 Testes

Agora, podemos seguir os mesmos passos de 5.1.3 e, ao conectarmos ao Apache através do cliente, será solicitado o "username" (utilizado na configuração do Google Authenticator) e o código OTP:

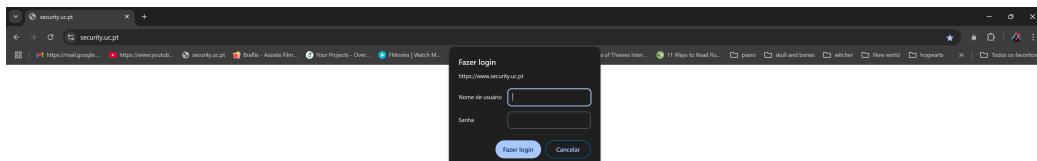


Figure 10: Solicitação Código OTP no Apache

6 OCSP

6.1 Objetivo

Para essa etapa, faremos a integração e o teste do nosso servidor OCSP ao Apache, através da tentativa de acesso com um cliente com um certificado revogado, ou seja, inválido. Para isso, primeiramente, criaremos e revogaremos um certificado e depois configuraremos a integração com o OCSP.

6.2 Execução

Primeiramente, criamos o certificado a ser revogado da mesma forma como detalhado em 2.2:

```
openssl genrsa -out client_apache_revoke.key -des3  
openssl req -new -key client_apache_revoke.key -out  
client_apache_revoke.csr -subj "/C=PT/ST=Coimbra/L=Coimbra  
/O=UC/OU=DEI/CN=client_apache_revoke/emailAddress=  
uc2024156583@student.uc.pt" -passin pass:test
```

Para continuar, precisamos copiar o ficheiro "serial" para um ficheiro "crlnumber":

```
cd demoCA  
cp serial crlnumber
```

Em seguida, revogamos o certificado e geramos a lista de certificados revogados:

```
cd ..  
openssl ca -revoke client_apache_revoke.crt -keyfile ca.key -  
cert ca.crt  
openssl ca -gencrl -out demoCA/crl.pem -keyfile demoCA/  
private/ca.key -cert demoCA/certs/ca.crt -config demoCA/  
openssl.cnf
```

Depois disso, podemos executar o comando para verificar a validade do certificado. O mesmo estará marcado com "R":

```
cat index.txt
```

Por fim, alteramos o arquivo de configuração do Apache para realizar a verificação dos certificados. Assim, caso um certificado revogado seja utilizado, bloqueará a conexão. Alteramos o arquivo "/etc/apache2/sites-available/default-ssl.conf":

```
...  
SSLCertificateChainFile /home/kali/Desktop/Seguranca/Project/  
ca.crt  
SSLCARevocationFile /home/kali/Desktop/Seguranca/Project/  
demoCA/crl.pem  
SSLCARevocationCheck chain  
SSLCACertificateFile /home/kali/Desktop/Seguranca/Project/ca.  
crt  
SSLVerifyClient require  
SSLVerifyDepth 3  
SSLUseStapling on  
SSLStaplingResponderTimeout 5  
SSLStaplingReturnResponderErrors on
```

```
SSLStaplingForceURL http://192.168.1.76:8081/  
...
```

Por fim, reiniciamos o Servidor Apache para aplicar as alterações:

```
sudo systemctl restart apache2
```

6.3 Testes

Para o teste dessa parte, basta importar o certificado revogado e tentar a conexão. Veremos que a conexão será bloqueada:

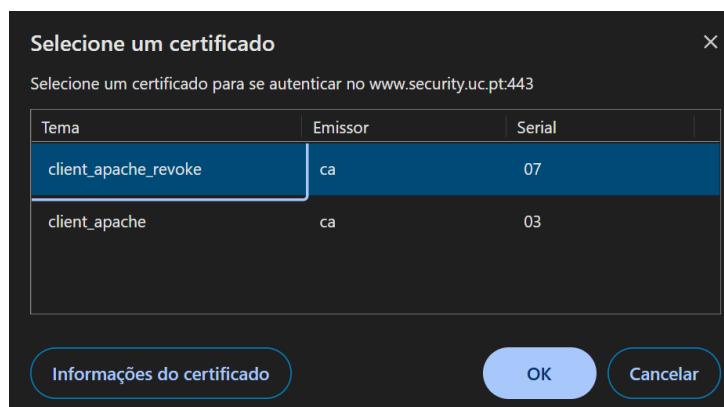


Figure 11: Seleção de Certificado no Google Chrome

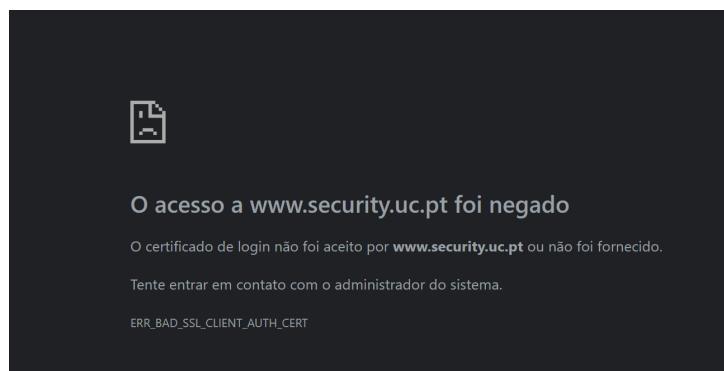


Figure 12: Conexão com Certificado Revogado

7 Envio PGP

Essa seção demonstra como encriptamos e assinamos o ZIP do projeto com PGP keys.

Para isso, acessamos o link enviado pelo professor para copiar sua chave pública que será utilizada para encriptar o nosso ficheiro ZIP. Vale ressaltar que, apenas o professor, que possui acesso a chave privada associada a essa chave pública, conseguirá decriptar o projeto.

```
# Acessar https://flowcrypt.com/pub/bmsousa@ieee.org?show=pubkey  
copiar conteudo da chave publica
```

Importar a chave pública do professor:

```
gpg --import  
colar conteudo da chave publica  
control + D
```

Por fim, executamos o comando para encriptar o ZIP do projeto com a chave pública do professor (através da indicação do ID em `--recipient`) e assinamos com a nossa chave privada (através de `--sign`), garantindo autenticidade:

```
gpg --sign --encrypt --recipient bmsousa@ieee.org --armor  
filename
```

E agora, o professor pode descriptar, importando nossa chave pública para verificar nossa assinatura, e executando o comando:

```
gpg --output filename_decryptado --decrypt filename.asc
```

8 Reconhecimento

O chatgpt foi usado quer para gerar o script do OCSP quer para ajudar a corrigir erros durante todo o trabalho prático.

9 Conclusão

A implementação da VPN no modelo road warrior permitiu estabelecer comunicações seguras entre clientes remotos e a rede interna, através de uma VPN e garantindo autenticação robusta com certificados X.509, OTP e validação via OCSP.