

Practical Assignment #2

João Neto – 2023234004

Vasco Alves – 2022228207

27 de abril de 2026

Índice

1	Introdução	2
2	Preparação Inicial	2
2.1	Criação de Certificados	2
3	Configuração geral	3
3.1	Configurar TOTP	3
3.2	Encaminhamento e Firewall	4
4	Configuração do Cliente (Road Warrior)	5
5	Servidor Apache e OCSP	5
5.1	Revocation e OCSP	5
5.2	Testes	6
6	Conclusão	6

1 Introdução

Este projeto tem como âmbito implementar, uma rede virtual privada (VPN) num cenário de road-warrior, configurar *two-factor authentication* (2FA) com os serviços OpenVPN e Apache, e gerir certificados X.509 utilizando OCSP.

Decidimos utilizar apenas três máquinas virtuais: o cliente (ou *road warrior*), a *gateway* que utiliza OpenVPN e um servidor interno com OpenSSL e Apache. Isto simplifica a elaboração do projecto, mas por razões de segurança poderia querer separar a máquina de OpenSSL de outras máquinas destinadas a serviços da rede inteira, pois esta contém o *certificate authority* CA.

Nome	Script	Rede
Road Warrior	VM_ROAD_WARRIOR.sh	Rede Externa 193.168.0.0/24
VPN Gateway	VM_OPENVPN_GATEWAY.sh	Router
OpenSSL / Apache	VM_OPENSSL_APACHE.sh	Rede Interna 10.60.0.0/24

2 Preparação Inicial

2.1 Criação de Certificados

Os certificados utilizados foram auto-certificados por uma autoridade central que "per-tence" à máquina de OpenSSL. Esta mesma faz a gestão da lista de revogação.

Todas as chaves foram criadas no mesmo computador, com as variáveis que estão neste código, aspetos importantes para mais tarde serão os parâmetros de CN que precisam de ser passados mais tarde para aceder ao Apache e ao gateway. Numa situação normal te-riamos uma autoridade de certificação para enviar e no fundo gerir todos, mas para este cenário podemos inicializar as máquinas com as chaves, requests e certificados necessá-rios.

O código para gerar os certificados X.509:

create_all_keys.sh

```
1 cert_ca="/C=PT/ST=Coimbra/L=Coimbra/O=UC/CN=CoimbraVPN"
2 cert_vpn="/C=PT/ST=Coimbra/L=Coimbra/O=UC/CN=gateway"
3 cert_user="/C=PT/ST=Coimbra/L=Coimbra/O=UC/CN=warrior"
4 cert_apache="/C=PT/ST=Coimbra/L=Coimbra/O=UC/CN=apache.coimbra"
5
6 openssl genrsa -out "ca.key" 2048
7 openssl req -x509 -nodes -days 365 -key "ca.key" -out "ca.crt" -subj "
  $cert_ca"
8 openssl genrsa -out "vpn.key" 2048
9 openssl req -new -key "vpn.key" -out "vpn.csr" -subj "$cert_vpn"
10 openssl ca -batch -in "vpn.csr" -cert "ca.crt" -keyfile "ca.key" -out "vpn
  .crt" -config cheese.cfg
11 openssl dhparam -out "dh2048.pem" 2048
12 openvpn --genkey secret "ta.key"
13 openssl genrsa -out user.key
14 openssl req -new -key user.key -out user.csr -subj "$cert_user"
15 openssl ca -batch -in "user.csr" -cert "ca.crt" -keyfile "ca.key" -out "
```

```

    user.crt" -config cheese.cfg
16 openssl genrsa -out apache.key
17 openssl req -new -key apache.key -out apache.csr -subj "$cert_apache" -
    addext "subjectAltName = IP:10.60.0.1,DNS:apache"
18 openssl ca -batch -in "apache.csr" -cert "ca.crt" -keyfile "ca.key" -out "
    apache.crt" -config cheese.cfg
19 openssl --genkey secret ta.key

```

3 Configuração geral

Para configurar as VMs era preciso introduzir os mesmos comandos várias vezes, o que levava muitas vezes a erros de escrita, ou a correr o mesmo comando várias vezes, por isso criamos vários ficheiros .sh para conseguir facilitar o processo. A utilização de ficheiros .sh também vem com outros positivos pois facilita a testagem, e a recriação do cenário rapidamente.

No entanto para os serviços que configuramos, instalar, desativar e dar flush às iptables não foi suficiente, tivemos que criar pastas e sincronizar os relógios de todas as VMs visto que elas estarem ligeiramente atrasadas nunca conseguimos acertar na password do google-authenticator que utiliza o tempo local para calcular a sua chave.

VM_CONFIG.sh

```

1 yum install -y epel-release
2 yum install -y openvpn iptables-services dhcp-client
3 systemctl stop firewalld
4 systemctl disable firewalld
5 systemctl mask firewalld
6 systemctl enable iptables
7 iptables -F
8
9 CA_DIR="/etc/pki/CA"
10 mkdir -p "${CA_DIR}/newcerts"
11 mkdir -p "${CA_DIR}/private"
12 touch "${CA_DIR}/index.txt"
13 cp ca/serial "${CA_DIR}/serial"
14
15 mkdir -p /etc/openvpn/server
16 mkdir -p /etc/openvpn/client
17
18 # NOTE(vasco): tive problemas com a sincronizacao de tempo
19 # se nao tiver sincronizado, o TOTP nao funciona
20 systemctl stop chronyd
21 ntpdate pool.ntp.org
22 systemctl start chronyd

```

3.1 Configurar TOTP

Foi criado o ficheiro totp com a configuração de autenticação a ser utilizada pelo plugin de PAM para o openvpn.

```
/etc/pam.d/totp
```

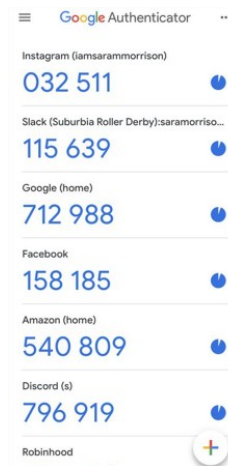
```
1 plugin /usr/lib64/openvpn/plugins/openvpn-plugin-auth-pam.so totp
```

Adicionalmente, devido às restrições de segurança do *systemd*, foi necessário desativar o *ProtectHome* no serviço do OpenVPN para que o plugin PAM consiga ler os ficheiros de segredo do Google Authenticator localizados nas diretorias *home* dos utilizadores.

```
override.conf
```

```
1 [Service]
2 ProtectHome=false
```

Primeiro, na gateway, entramos como o utilizador desejado e obtemos a chave do gerador de palavras passas temporárias. Ao inserir a chave no *google authenticator* podemos obter um código QR, a nossa primeira chave de 6 dígitos.



```
1 su john
2 google-authenticator
```

3.2 Encaminhamento e Firewall

Para que a gateway funcione como router entre a rede externa e a rede interna, foi necessário ativar o *IP forwarding* no kernel e configurar as regras de *iptables* para permitir o tráfego da VPN e realizar o mascaramento de IP (NAT).

```
firewall.sh
```

```
1 # Ativar encaminhamento
2 echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.conf
3 sysctl -p /etc/sysctl.conf
4
5 # Regras de Firewall
6 iptables -I INPUT 1 -p udp --dport 1194 -j ACCEPT
7 iptables -I FORWARD 1 -i tun0 -o enp0s9 -j ACCEPT
```

```
8 iptables -I FORWARD 1 -i enp0s9 -o tun0 -j ACCEPT
9 iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o enp0s8 -j MASQUERADE
```

4 Configuração do Cliente (Road Warrior)

O cliente encontra-se na rede externa (193.136.212.10) e liga-se à VPN gateway na porta 1194. Para garantir a segurança, utilizamos autenticação mútua (os certificados X.509) e um *two factor authentication* (2FA) como palavras-passe temporárias, geradas através do *Google Authenticator*.

client.conf

```
1 client
2 dev tun
3 proto udp
4 remote 193.136.212.1 1194
5 ca ca.crt
6 cert user.crt
7 key user.key
8 auth-user-pass
9 cipher AES-256-GCM
10 auth SHA256
```

5 Servidor Apache e OCSP

O servidor interno (10.60.0.1) alberga o serviço Apache e o responder OCSP da autoridade de certificação.

5.1 Revocation e OCSP

1. Estabelecer a ligação VPN e verificar a conectividade à rede interna.
2. No diretório da autoridade de certificação (máquina *host*), revogar o certificado do utilizador:

revoke.sh

```
1 openssl ca -revoke user.crt -config cheese.cfg -keyfile ca.key -cert
  ca.crt
```

3. Atualizar o ficheiro `index.txt` no servidor OCSP e reiniciar o serviço para carregar o novo estado de revogação.
4. Tentar estabelecer uma nova ligação VPN e verificar que a autenticação falha devido à resposta `revoked` do responder OCSP.

5.2 Testes

Podemos validar que o OCSP

6 Conclusão

Atingimos o objetivo deste trabalho, conseguimos configurar o VPN tunnel, o two-factor authentication e conseguimos criar e retirar acesso aos certificados que emitimos. Utilizar mais maquinas para simular um cenario maior seria redundante, teriamos que emitir mais certificados mas não iamos aprender muito mais. Se fossemos aplicar o que fizemos no trabalho anterior podiamos dar DROP aos pacotes que não nos interessa nesta cenario, e implementar suricata para identificar possiveis ataques nos serviços.