

Practical Assignment #2

João Neto – 2023234004

Vasco Alves – 2022228207

3 de maio de 2026

Conteúdo

1	Introdução	3
2	Preparação Inicial	3
2.1	Criação de Certificados	3
2.2	Configuração geral	4
3	VPN Gateway	5
3.1	Configuração da Máquina	5
3.2	Configuração do Serviço OpenVPN	6
3.3	Erros	7
3.4	Configurar o utilizador com TOTP	7
4	VPN Client (Road Warrior)	8
4.1	Configuração da Máquina	8
4.2	Configuração do Cliente OpenVPN	8
4.3	Testes	9
5	Servidor Apache e OCSP	9
5.1	Configuração da Máquina	10
5.1.1	Testes	12
6	Teste Integrado	13
7	Conclusão	13

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 interna, 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 "perence" 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 Comon Name (CN) pois servem para a validação do certificado ambos pelo OpenSSL e pelo browser.

Nós optamos por assumir que num cenário real, teríamos acesso físico às máquinas, por isso em vez de utilizar, por exemplo SCP ou FTP, escolhemos partilhar os ficheiros a partir da máquina host. No entanto, outra abordagem também estaria correta.

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

```

Como o CA foi criado *"in place"*, e não na sua pasta prédefinida, foi necessário utilizar um configuração própria para definir os ficheiros *index.txt* e *serial*.

cheese.cfg

```

1 [ ca ]
2 default_ca = CA_default
3 [ CA_default ]
4 default_days = 365
5 database = index.txt
6 serial = serial
7 copy_extensions = copy
8 new_certs_dir = .
9 default_md = sha256
10 policy = policy_any
11 [ policy_any ]
12 commonName = supplied

```

2.2 Configuração geral

Para evitar repetição e redundancia; e para garantir consistencia na elaboração do projeto criamos varios shell scripts, um destinado a cada maquina virtual.

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 visto 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 VPN Gateway

3.1 Configuração da Máquina

Como já foi dito anteriormente, cada máquina vem com um *script* que instala toda a configuração necessária.

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).

VM_VPN_GATEWAY.sh

```

1 #!/bin/bash
2
3 # --- configuracao --- #
4 source VM_CONFIG.sh
5 yum install -y google-authenticator qrencode ntpsec
6
7 # --- forwarding --- #
8 if_fora="enp0s8"
9 ip_fora="193.136.212.1"
10 if_dentro="enp0s9"
11 ip_dentro="10.60.0.3"
12 mega_tunel="tun0"
13 ip_mega_tunel="10.8.0.0/24"
14
15 ifconfig $if_fora $ip_fora netmask 255.255.255.0
16 ifconfig $if_dentro $ip_dentro netmask 255.255.255.0
17
18 echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.conf
19 sysctl -p /etc/sysctl.conf
20
21 iptables -I INPUT 1 -p udp --dport 1194 -j ACCEPT
22 iptables -I FORWARD 1 -i $mega_tunel -o $if_dentro -j ACCEPT
23 iptables -I FORWARD 1 -i $if_dentro -o $mega_tunel -j ACCEPT
24 iptables -I FORWARD 1 -i $mega_tunel -o $if_fora -j ACCEPT
25 iptables -I FORWARD 1 -i $if_fora -m state --state ESTABLISHED,RELATED -j
    ACCEPT
26 iptables -t nat -A POSTROUTING -s $ip_mega_tunel -o $if_fora -j MASQUERADE
27 iptables-save > /etc/sysconfig/iptables
28

```

```

29 # --- vpn server --- #
30 vpn_dir="/etc/openvpn/server"
31 cp ca/ta.key $vpn_dir
32 cp ca/ca.crt $vpn_dir
33 cp ca/vpn.key $vpn_dir
34 cp ca/vpn.crt $vpn_dir
35 cp ca/dh2048.pem $vpn_dir
36 cp conf/vpn.conf $vpn_dir
37 cp conf/ocsp-verify.sh $vpn_dir
38 cp conf/totp /etc/pam.d/
39
40 # --- utilizador --- #
41 id -u john &>/dev/null || useradd john
42 echo "password" | passwd --stdin john
43
44 openvpn --config /etc/openvpn/server/vpn.conf

```

3.2 Configuração do Serviço OpenVPN

O servidor OpenVPN utiliza um certificado X.509 assinado pelo nosso *Certificate Authority* (CA). E faz uso de um script `ocsp-verify.sh` para validar ou revogar os certificados através do servidor OCSP.

vpn.conf

```

1 local 193.136.212.1
2 port 1194
3 proto udp
4 dev tun
5
6 verb 4
7
8 ca /etc/openvpn/server/ca.crt
9 cert /etc/openvpn/server/vpn.crt
10 key /etc/openvpn/server/vpn.key
11 dh /etc/openvpn/server/dh2048.pem
12
13 topology subnet
14 server 10.8.0.0 255.255.255.0
15 push "route 10.60.0.0 255.255.255.0"
16
17 # ocsp and revocation
18 script-security 2
19 tls-verify /etc/openvpn/server/ocsp-verify.sh
20 # auth
21 cipher AES-256-GCM
22 auth SHA256
23
24 plugin /usr/lib64/openvpn/plugins/openvpn-plugin-auth-pam.so totp
25 tls-auth /etc/openvpn/server/ta.key 0

```

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

totp

```
1 auth      required    pam_google_authenticator.so forward_pass
2 auth      required    pam_unix.so use_first_pass
3 account   required    pam_unix.so
```

Este script simplesmente comunica com o servidor OpenSSL e verifica o resultado.

ocsp_verify.sh

```
1 #!/bin/bash
2 depth=$1
3 if [ "$depth" -eq 0 ]; then
4     if [ -n "$tls_serial_0" ]; then
5         # e preciso converter o serial para hexadecimal porque o openssl
        espera em hex
6         hex_serial=$(printf '%x' "$tls_serial_0")
7         status=$(openssl ocsp -issuer /etc/openvpn/server/ca.crt -serial
            "0x$hex_serial" -url http://10.60.0.1:8888 -CAfile /etc/openvpn/server
            /ca.crt 2>/dev/null)
8         if echo "$status" | grep -q "good"; then
9             exit 0 # sucesso
10        fi
11        exit 1 # revogado ou nao encontrado
12    fi
13    exit 1
14 fi
```

3.3 Erros

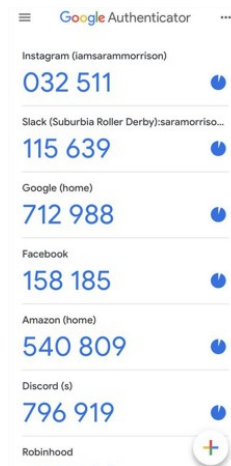
Um dos erros que encontramos pelo caminho foi que o OpenSSL OCSP espera que o *serial* esteja num formato diferente do que o esperado. Foi necessário converter para hexadecimal primeiro.

Adicionalmente, devido às restrições de segurança do *systemd*, tentamos 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. Mas isto não foi suficiente, por isso acabamos por correr os serviços pela linha de comandos.

3.4 Configurar o utilizador com TOTP

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
```



4 VPN Client (Road Warrior)

4.1 Configuração da Máquina

Para a configuração da Máquina, configuramos o endereço, o default gateway e adicionamos apache aos Hosts:

VM_ROAD_WARRIOR.sh

```
1 #!/bin/bash
2
3 # --- configuracao --- #
4 source VM_CONFIG.sh
5 ifconfig enp0s8 193.136.212.10 netmask 255.255.255.0
6 route add default gw 193.136.212.1
7
8 if ! grep -q "apache" /etc/hosts; then
9     echo "10.60.0.1 apache" >> /etc/hosts
10 fi
11
12 # --- vpn client --- #
13 vpn_dir="/etc/openvpn/client/"
14 cp ca/ta.key $vpn_dir
15 cp ca/ca.crt $vpn_dir
16 cp ca/user.key $vpn_dir
17 cp ca/user.crt $vpn_dir
18 cp conf/client.conf $vpn_dir
19
20 openvpn --config "${vpn_dir}/client.conf"
```

Também foram movidos os certificados e chaves necessarias para as pastas do serviço openvpn, para que o Road Warrior consiga comunicar e ser validado pela gateway.

4.2 Configuração do Cliente OpenVPN

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
```

4.3 Testes

Para verificar que a autenticação foi corretamente implementada, inserimos a password de um utilizador sem os dígitos do TOTP, e identificamos que utilizar somente a password não é suficiente para autenticar. Igualmente ao utilizar ambos a autenticação é bem sucedida.

Para verificar que o tunel foi estabelecido, primeiro corremos na linha de comandos `ip a`. Observamos a existencia de uma nova interface `tun0`, ou seja o tunel foi corretamente estabelecido. Depois demos `ping` ao `route` e depois ao servidor interno, que resultou em pacotes devolvidos para ambos.

Para verificar que o OCSP funciona correctamente, o cliente conectou ao servidor OpenVPN: primeiro, sem o servidor OCSP a correr, uma segunda vez com ele a correr e com o certificado correcto e uma terceira vez com um certificado revogado. Fizemos estes testes sabendo que o cliente e o servidor já estavam correctamente configurados.

Verificamos que, como é suposto: sem OCSP não é possível autenticar; com OCSP e com certificado válido, podemos autenticar; e com OCSP mas com certificado revogado, a autenticação falha.

5 Servidor Apache e OCSP

Para a configuração da ultima maquina, temos o OpenSSL e Apache no mesmo servidor, por isso temos de configurar as pastas necessarias, os utilizadores do serviço, configurar os edereços e uma route:

VM_OPENSSL_APACHE.sh

```
1 #!/bin/bash
2 # configuracao
3 source VM_CONFIG.sh
4
5 sudo yum install -y epel-release
6 sudo yum install -y openssl httpd mod_ssl mod_authnz_pam google-
  authenticator
7 sudo yum install -y mod_session
```

```

8
9 # utilizador
10 id -u john &>/dev/null || useradd john
11 echo "password" | passwd --stdin john
12
13 if_dentro="enp0s8"
14 ip_dentro="10.60.0.1"
15 ifconfig $if_dentro $ip_dentro netmask 255.255.255.0
16
17 # route de volta para comunicar com o warrior
18 route add -net 10.8.0.0 netmask 255.255.255.0 gw 10.60.0.3
19
20 cp conf/openssl.cnf /etc/pki/tls/
21
22 # copiar ca para esta VM
23 cp ca/index.txt $CA_DIR
24 cp ca/ca.crt $CA_DIR
25 cp ca/ca.key $CA_DIR
26 cp ca/serial $CA_DIR
27 cp ca/dh2048.pem $CA_DIR
28
29 # correr oscp
30 killall openssl 2>/dev/null
31 openssl ocsf -index $CA_DIR/index.txt -port 8888 -rsigner $CA_DIR/ca.crt -
    rkey $CA_DIR/ca.key -CA $CA_DIR/ca.crt -text &
32
33 # apache
34 mkdir -p /etc/httpd/ssl
35 cp ca/ca.crt /etc/httpd/ssl/
36 cp ca/apache.crt /etc/httpd/ssl/
37 cp ca/apache.key /etc/httpd/ssl/
38 cp conf/ssl.conf /etc/httpd/conf.d/ssl.conf
39 cp conf/httpd.conf /etc/httpd/conf/httpd.conf
40 cp conf/httpd-totp /etc/pam.d/httpd-totp
41
42 echo "LoadModule session_module modules/mod_session.so" > /etc/httpd/conf.
    modules.d/01-session.conf
43 echo "LoadModule session_cookie_module modules/mod_session_cookie.so" >> /
    etc/httpd/conf.modules.d/01-session.conf
44 echo "LoadModule auth_form_module modules/mod_auth_form.so" > /etc/httpd/
    conf.modules.d/01-auth_form.conf
45
46 cp -r www/* /var/www/html/
47 chown -R apache:apache /var/www/html/
48
49 httpd -X

```

5.1 Configuração da Máquina

Como já referimos a Máquina tem ambos o serviço OpenSSL e Apache, por isso vai precisar de dois .conf files para configurar-los. O httpd.conf tem as portas e módulos enquanto o ssl.conf tem a configuração da autenticação mútua, e o OCSP:

httpd.conf

```
1 ServerRoot "/etc/httpd"
2
3 Include conf.modules.d/*.conf
4 LoadModule authnz_pam_module modules/mod_authnz_pam.so
5 LoadModule mpm_event_module modules/mod_mpm_event.so
6
7 User apache
8 Group apache
9
10 Listen 80
11 Listen 443
12
13 Include conf.d/*.conf
14
15 DocumentRoot "/var/www/html"
16 <Directory "/var/www/html">
17     Options Indexes FollowSymLinks
18     AllowOverride None
19     Require all granted
20 </Directory>
```

ssl.conf

```
1 <VirtualHost *:443>
2     ServerName 10.60.0.1
3     DocumentRoot /var/www/html
4
5     SSLEngine on
6     SSLCertificateFile /etc/httpd/ssl/apache.crt
7     SSLCertificateKeyFile /etc/httpd/ssl/apache.key
8     SSLCACertificateFile /etc/httpd/ssl/ca.crt
9
10     # mutual authentication
11     SSLVerifyClient require
12     SSLVerifyDepth 1
13
14     # ocsp validation
15     SSL0CSPEnable on
16     SSL0CSPDefaultResponder "http://10.60.0.1:8888"
17     SSL0CSPOverrideResponder on
18     SSL0CSPUseRequestNonce off
19
20     # session management
21     Session On
22     SessionCookieName session path=;/HttpOnly;Secure
23
24     # proteger
25     <Location "/">
26         AuthType Form
27         AuthName "Coimbra VPN"
28         AuthFormProvider PAM
29         AuthPAMService httpd-totp
30         AuthFormLoginRequiredLocation "/login.html"
31         Require valid-user
32     </Location>
33
34     # public login page
35     <Location "/login.html">
```

```

36     AuthType None
37     Require all granted
38 </Location>
39
40 # login handler
41 <Location "/dologin">
42     SetHandler form-login-handler
43     AuthType Form
44     AuthName "Coimbra VPN"
45     AuthFormProvider PAM
46     AuthPAMService httpd-totp
47     Require all granted
48     AuthFormLoginSuccessLocation "/index.html"
49     AuthFormLoginRequiredLocation "/login.html?error=1"
50 </Location>
51
52 # logout handler
53 <Location "/logout">
54     SetHandler form-logout-handler
55     AuthFormLogoutLocation "/login.html?loggedout=1"
56 </Location>
57
58 </VirtualHost>
59
60 # redirect para https
61 <VirtualHost *:80>
62     ServerName 10.60.0.1
63     Redirect permanent / https://10.60.0.1/
64 </VirtualHost>

```

5.1.1 Testes

- **Domínio:** Verificou-se que o acesso só é permitido utilizando o endereço correto, pois se for inserido outro domínio, não é direcionado para o site do Apache.
- **Redirecionamento HTTPS:** Ao testar quando colocamos http, e o domínio certo, era redirecionado para https.
- **Autenticação com o Certificado:** O acesso foi negado ao apresentar certificados inválidos ou ausentes no browser, devolvendo um erro com sobre não conseguir estabelecer conexão porque falta de certificado.

Para Testar o OCSP, fizemos os seguintes passos:

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.

6 Teste Integrado

Para validar a solução completa, efetuámos um teste fim-a-fim englobando todos os requisitos:

1. Na máquina *Road Warrior*, iniciámos a ligação OpenVPN fornecendo o certificado de cliente e, como password, a concatenação da password do utilizador com o código TOTP gerado pelo *Google Authenticator*.
2. O *Gateway* OpenVPN verificou o certificado cliente contra o servidor OCSP e validou o TOTP, permitindo o estabelecimento do túnel.
3. Através do túnel VPN, acedemos ao endereço `https://10.60.0.1` no browser.
4. O servidor Apache solicitou o certificado X.509 do utilizador e validou a sua autenticidade e estado de revogação no OCSP.
5. Finalmente, o Apache apresentou a página de login para o segundo fator, onde inserimos as credenciais e o código TOTP, concedendo acesso à página interna protegida.

Este teste confirmou que a comunicação está segura, com dupla verificação de identidade e validação contínua da integridade dos certificados.

7 Conclusão

Atingimos o objetivo deste trabalho: conseguimos configurar o túnel VPN, o *two-factor authentication* em múltiplos serviços, e conseguimos gerir o ciclo de vida dos certificados emitidos através de uma CA própria e OCSP. Utilizar mais máquinas para simular um cenário maior seria redundante e apenas exigiria a emissão de mais certificados, não acrescentando muito ao nível de aprendizagem.

Aplicando conhecimentos de trabalhos anteriores, poderíamos aplicar políticas mais restritas nas *iptables* (ex: regras de DROP aos pacotes indesejados), e implementar ferramentas como o Suricata para identificar possíveis anomalias e ataques aos serviços.