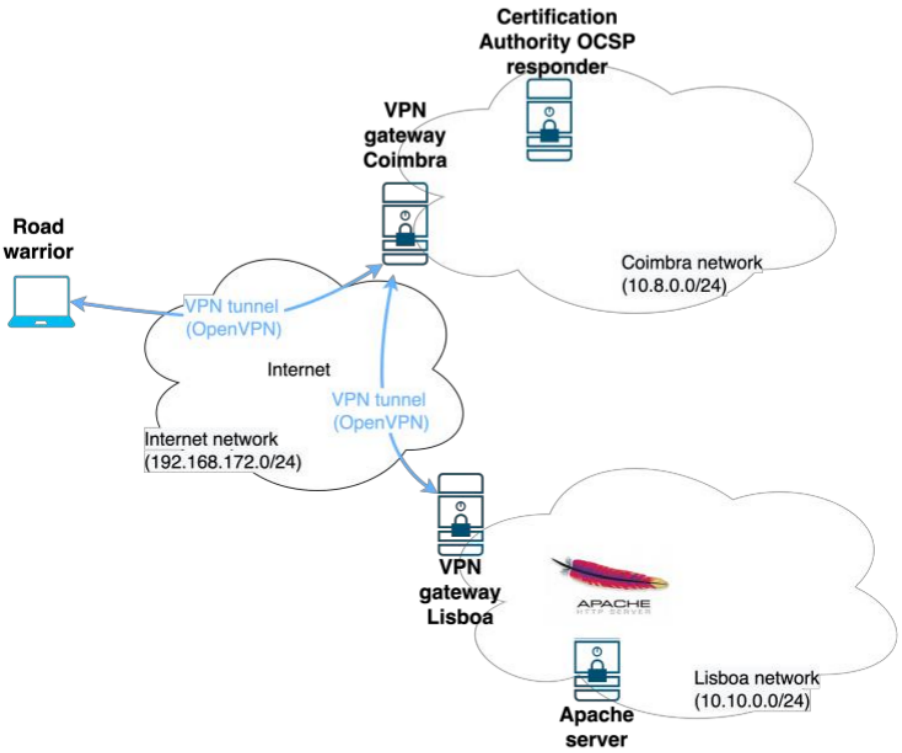
1 2 9 0
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Segurança em Tecnologias da Informação

Practical Assignment #1



2021/2022

Mestrado em Engenharia Informática

PL2	Joana Brás	2021179983	joanabras@student.dei.uc.pt
PL2	Pedro Rodrigues	2018283166	pedror@student.dei.uc.pt

---

1. Introdução

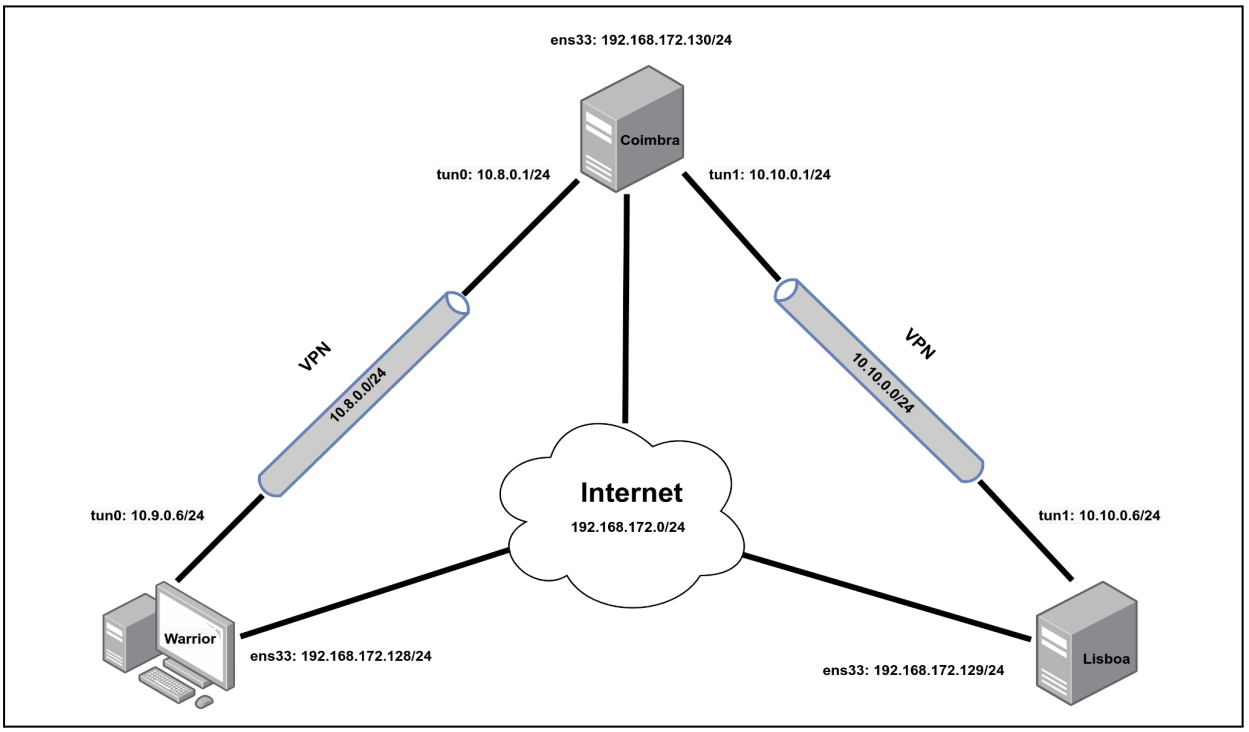
O presente relatório aborda uma série de componentes necessários para o primeiro trabalho prático da disciplina de Segurança em Tecnologias da Informação. Este tem vários objetivos, sendo os principais:

1. Configuração de ligações entre servidores (no nosso cenário, um em Coimbra e outro em Lisboa), estabelecendo estas através de serviços de segurança garantidos pelo protocolo VPN.
2. Ligações entre servidor e cliente (no nosso cenário, *road warrior* e servidor de Coimbra), estabelecendo, mais uma vez, estas através de serviços de segurança garantidos pelo protocolo VPN.

3. Implementar autenticação de duas etapas (recorrendo ao algoritmo TOTP) e usando o serviço do *Google Authenticator*.
4. Configurar um serviço de verificação de status de certificados (recorrendo ao protocolo OCSP), a fim de garantir a validade de todos os certificados a serem emitidos por uma Autoridade Certificadora (CA) privada, criada por nós (e operando no servidor de Coimbra).
5. Disponibilizar o acesso a um servidor http(s) *Apache* ao *road warrior* por túnel VPN. Esta última ligação terá de ser verificada pelo OCSP, e certificada pela CA, o que garante que o cliente tenha um certificado válido, facultando-lhe o acesso ao túnel VPN, e por sua vez o acesso ao *website* disponibilizado pelo Apache.

2. Cenário de Rede

Para efeitos de visualização e melhor perceção do cenário que nos foi proposto, construímos este diagrama simplificado que mostra os principais aspetos do trabalho nomeadamente as redes, túneis VPN e IP 'são atribuídos às interfaces de todos os dispositivos.



3. Instalações

Para a implementação das funcionalidades necessárias fizemos uso do seguinte software:

- apache2 (servidor Lisboa)
- openvpn
- openssl
- libqrencode4 (servidor Coimbra)
- libpam-google-authenticator (servidor Coimbra)
- aplicação de Google Authenticator (telemóvel)

Este foi instalado nas diversas *Virtual Machines* usadas para a realização deste projeto. O software de virtualização utilizado para a sua criação foi o VMware.

4. Configurações

4.1. Gerais

De forma a garantir que os IPs das interfaces ethernet (ens33) das máquinas virtuais terem um IP da rede 192.168.172.0 com máscara 255.255.255.0 (/24), por requisito do enunciado, foi alterada a configuração do software de virtualização (VMware). Para mais informações acerca da forma de como configurar a rede entre VM's e sobre como serviço de DHCP está configurado no VMware a seguinte documentação contém informações que poderão ser bastante relevantes: https://docs.vmware.com/en/VMware-Workstation-Pro/16.0/com.vmware.ws.using.doc/GUID-9831F49E-1A83-4881-BB8A-D4573F2C6D91.html.

4.2. Servidor de Coimbra

4.2.1. Criação da CA

A CA (Certification Authority) é uma componente comum a todas as máquinas virtuais, já que esta entidade será considerada como sendo qualificada para emitir certificados que validem a autenticidade/confiabilidade de todos os hosts envolvidos num processo de comunicação (neste trabalho servidores de Coimbra e Lisboa e *road warrior*). No nosso trabalho esta entidade corresponde ao servidor de Coimbra, que funcionará como uma autoridade de certificação privada. Desta forma, a nossa CA (privada) começará por emitir um pedido de emissão de certificado, sendo este utilizado para gerar um certificado *self-signed*, necessário na emissão dos certificados para outros hosts a fim de permitir o estabelecimento de túneis VPN (entre *road warrior*, coimbra, e lisboa) entre outros serviços (servidor Apache).

Antes da criação da chave e ficheiro *CSR* (*certificate signing request*), necessários à produção do certificado *self-signed* da nossa CA privada, começamos por alterar algumas configurações usadas pela ferramenta openSSL, presentes no ficheiro "`/etc/ssl/openssl.cnf`":

1. Para diretoria onde serão colocados todos os ficheiros relacionados com a geração criação e revogação de certificados pela nossa CA, consideramos o seguinte path "`/etc/pki/CA`". Assim no ficheiro de configuração do openSSL, na secção "`CA_default`", no campo "`dir`", o seguinte foi acrescentado: "`dir = /etc/pki/CA`".
2. Para permitir a criação de certificados com atributos diferentes do certificado criado para a CA (por exemplo no que diz respeito à City/Location e State/Province), na secção "`CA_default`", no campo "`policy`", onde são identificadas as políticas consideradas na emissão de certificados, o seguinte foi acrescentado: "`policy = policy_anything`".
3. Para permitir que na emissão de certificados, estes tenham informação acerca do servidor OCSP a contactar para verificação da sua validade e estado, na secção "`usr_cert`" adicionamos a seguinte extensão "`authorityInfoAccess = OCSP;URI:http://ocsp.coimbra.pt:81`"

Comandos Utilizados

Na criação de uma autoridade certificadora, criou-se uma chave privada de 2048 bits com o algoritmo de encriptação "des3" com o seguintes comandos:

```
cd /etc/pki/CA/private
sudo openssl genrsa -out cakey.pem des3 2048
```

Este último comando irá requerer uma palavra-passe necessária para encriptar a informação contida na chave que denominamos de "`cakey.pem`".

Posteriormente, criou-se o ficheiro contendo o *CSR (Certificate Signing Request)* da nossa CA privada intitulado "`ca.crt`", fazendo uso da chave privada criada anteriormente, usando-se para o efeito os seguintes comandos:

```
cd /etc/pki/CA
sudo openssl req \
        -new -key /etc/pki/CA/private/cakey.pem \
        -out ca.crt
```

Após introdução da palavra-passe para criar o ficheiro com o *CSR*, irão ser pedidos vários parâmetros para descrever o mesmo.

O seguinte exemplo mostra a descrição dos parâmetros que foram usados no processo de emissão do CSR após correr o comando acima:

```
Enter pass phrase for private/cakey.pem:
    Country Name (2 letter code) [PT]:PT
    State or Province Name (full name) [Some-State]:Coimbra
    Locality Name (eg, city) []:Coimbra
    Organization Name (eg, company) [Internet Widgits Pty Ltd]:UC
    Organizational Unit Name (eg, section) []:DEI
    Common Name (e.g. server FQDN or YOUR name) []:CA
    Email Address []:ca@dei.uc.pt

    Please enter the following 'extra' attributes
    to be sent with your certificate request
    A challenge password []: *******
    An optional company name []:DEI
```

Como passo final para criação da CA, corremos a seguinte linha para criar um certificado X.509 *self-signed* (de nome "`cacert.pem`"):

```
cd /etc/pki/CA/certs
sudo openssl x509 -req \
        -days 3650 \
        -in ca.csr \
        -out cacert.pem \
        -signkey /etc/pki/CA/private/cakey.pem
```

4.2.2. Emissão de certificados

De forma a facilitar o processo de transferência de ficheiros e a transição de máquina virtual em máquina virtual, todos os certificados foram criados no servidor de Coimbra, bem como as chaves privadas e CSR 's de cada um deles. Num contexto real sabemos que isto não aconteceria visto que cada cliente iria gerar a sua chave privada e CSR, sendo este último enviado à CA que após verificar a autenticidade/confiabilidade da entidade/utilizador a pedir a certificação, emitiria o certificado correspondente.

No âmbito do nosso trabalho, foram necessários sete certificados X.509 adicionais, e portanto sete chaves, sete ficheiros *CSR* (três destes para o estabelecimento de túneis VPN entre as máquinas, outro para o servidor Apache, e três adicionais para representar utilizadores que acedam ao servidor Apache em cada uma destas máquinas). Os comandos usados para a criação de todos estes certificados , são em tudo semelhante ao já referido, com exceção do conteúdo presente em cada um e nas senhas usadas na criação de chaves privadas.

Para propósito de demonstração, as linhas de comando descritas posteriormente serão referentes à emissão do certificado para o servidor de Coimbra (os ficheiros terão o formato "`coimbra.*`" em que o "`*`" representa a extensão do ficheiro - "`key`" para a chave privada, "`csr`" para o *CSR* e "`crt`" para o certificado X.509).

Assim para criação desta chave correu-se o seguinte comando:

```
cd /etc/pki/CA/private
sudo openssl genrsa -out coimbra.key des3 2048
```

A criação dos ficheiros *CSR* fez-se seguindo a mesma lógica usada na criação do CSR da nossa CA (preenchendo toda a informação necessária):

```
cd /etc/pki/CA
sudo openssl req -new -key /etc/pki/CA/private/coimbra.key -out coimbra.csr
```

A maior diferença surge na criação do certificado X.509, já que este irá ser assinado pela CA, ao contrário do que tinha sido feito anteriormente, em que o certificado é *self-signed*. Como tal, o comando usado é o seguinte:

```
sudo openssl ca \
        -in coimbra-vpn.csr \
        cert /etc/pki/CA/certs/cacert.pem \
        -keyfile /etc/pki/CA/private/cakey.pem \
        -out /etc/pki/CA/certs/coimbra.crt
```

No caso dos certificados emitidos para testar a capacidade do servidor Apache de verificar a validade dos certificados dos utilizadores que se ligam a ele em cada uma das máquinas, (através do browser Firefox), procedemos ainda a um passo adicional, e emitindo cada um destes certificados no formato *PKCS#12*, possibilitando assim a sua importação no browser. O comando abaixo exemplifica a emissão do certificado a ser importado pelo o utilizador *road warrior* - "`warrior-user.p12`".

```
sudo mkdir exports
sudo openssl pkcs12 \
        -export \
        -out exports/warrior-user.p12 \
        -inkey /etc/pki/CA/private/warrior-user.key \
        -in /etc/pki/CA/certs/warrior-user.crt \
        -certfile /etc/pki/CA/cacert.pem
```

Para testar a revogação de certificados fizemos também uso ao seguinte comando (o comando seguinte usa como certificado exemplo o "`warrior-user.crt`") :

```
cd /etc/pki/CA
sudo openssl ca \
        -revoke certs/warrior-user.crt \
        -keyfile private/cacey.pem \
        -cert cacert.pem
```

4.2.3. Criação do servidor OCSP

Para a criação do servidor OCSP, tendo já alterado a configuração do ficheiro openssl, para que todos os certificados fossem emitidos contendo a informação acerca do URI da instância do servidor procedemos à criação de um script, denominado "`ocsp.sh`", que será utilizado pelo serviço de VPN que iremos configurar, permitindo a validação do certificado da máquina junto do servidor OCSP aquando a tentativa de estabelecimento do túnel de ligação VPN.

O mesmo encontra-se no *snippet* abaixo, e de uma forma geral efetua a chamada de um comando que pede à instância do servidor OCSP a correr no endereço "`http://ocsp.coimbra.pt`" e porto "`81`", para verificar a validade do certificado, e caso o resultado do comando contenha a string "*good*" num dos seus campos este termina com *exit code* de zero ou um caso o comando não funcione, por motivo de erro ou caso o certificado do utilizador tenha sido revogado. Este script foi adaptado dos exemplos contidos no livro *"OpenVPN 2 Cookbook"* de Jan Just Keijser.

```
CADIR="/etc/pki/CA"
VPNDIR="/etc/openvpn"
OCSP="http://ocsp.coimbra.pt:81"

[ "$1" -ne 0 ] && exit 0

cd "${VPNDIR}"

if [ -n "{tls_serial_0}" ]; then
        status=$(openssl ocsp \
                -issuer "${CADIR}/cacert.pem" \
                -CA "${CADIR}/cacert.pem" \
                -url ${OCSP} \
                -serial "0x${tls_serial_0}" 2> /dev/null)
        if [ $? -eq 0 ]; then
                echo "[INFO]: OCSP Server Status: ${status}"
                echo "${status}" | grep -Fq "0x${tls_serial_0}: good" && exit 0
        fi
        echo "[ERROR]: openssl ocsp command failed!!"
fi
exit 1
```

Tendo estas configurações em ordem, para iniciar o servidor OCSP corre-se o seguinte comando. Por uma questão de simplicidade, e visto que não conseguimos colocar o servidor a correr sem este bloquear a sessão de terminal, utilizamos um programa como o tmux (ou GNU Screen) a fim de deixar este a correr em background, não estando vinculado à sessão do terminal emulator atual:

```
sudo openssl ocsp \
    -index /etc/pki/CA/index.txt \
    -port 81
    -rsigner /etc/pki/CA/certs/cacert.pem \
    -rkey /etc/pki/CA/private/ca.pem \
    -CA /etc/pki/CA/cacert.pem \
    -text -out /etc/pki/CA/log.txt
```

4.2.4. Configuração do servidor VPN (OpenVPN)

Para configuração do serviço de VPN começamos por gerar uma chave simétrica necessária para a comunicação entre as entidades envolvidas, após o estabelecimento do túnel. Esta chave simétrica foi criada recorrendo ao algoritmo de Diffie Hellman (DH), tendo nós gerado chaves com 2048 bits. Para isso foi efetuado o seguinte comando:

```
cd /etc/openvpn/server
sudo openssl dhparam -out 2048 dh2048.pem
```

Além disso, serão criadas duas outras chaves, necessárias para as ligações envolvidas (servidor Coimbra com *road warrior* e servidor Coimbra com servidor Lisboa) que terão os nomes "`ta-warrior.key`" e "`ta-lisboa.key`" respetivamente. Estas chaves são necessárias para o uso do protocolo HMAC, usado na ligação VPN para verificação tanto de autenticidade das mensagens como a sua integridade, desta forma a robustez e segurança da ligação. Para efeitos de demonstração, exemplificamos a criação da chave "`ta-lisboa.key`" , sendo a criação da outra em tudo similar :

```
cd /etc/openvpn/server
sudo openvpn --genkey tls-auth ta-lisboa.key
```

No servidor de Coimbra, foram feitos dois ficheiros de configuração OpenVPN a funcionarem ambos como servidor para o *road warrior* e para o servidor de Lisboa. Ambos são semelhantes, mudando apenas o *port* usado (*port* 1193 e 1194), a linha referente ao "`push route`" (o IP "`10.8.0.0`" passa a "`10.10.0.0`"), o "`tls-auth`", em que o "`ta-lisboa.key`" passa a "`ta-warrior.key`" e a linha "`plugin`", No caso da ligação ao *road warrior* o ficheiro de configuração do servidor VPN é "`server-warrior.conf`", e no caso da ligação ao servidor de Lisboa, um ficheiro denominado "`server-lisboa.conf`",

Todos estes ficheiros foram colocados na directoria "`/etc/openvpn/server`" juntamente com o certificado X.509 e chave privada de Coimbra, criados anteriormente para o efeito. As diferenças entre estes dois ficheiros devem-se:

- Ao facto de estas duas instâncias de servidor VPN não poderem dar bind no mesmo porto visto estarem a correr na mesma máquina.

- Cada uma das instâncias (associadas a um túnel VPN diferente) terá de dar a conhecer a sua rede privada para efeitos de routing de pacotes. Isto é, os pacotes, por exemplo, que passam pelo túnel VPN do *road warrior* e que devem ser entregues aos servidor de Lisboa tem de ser distribuídos pelo servidor de coimbra. Como tal, é necessário enviar informação ao *road warrior* (por parte da instância de servidor VPN que está a gerir esse túnel) de que a rede entre Coimbra e Lisboa existe (sendo esta privada no ponto de vista do warrior) a fim dos pacotes chegarem ao destino e a tabela de routing do warrior saber sua existência encaminhando pacotes para lá.
- Para permitir o serviço de TOPT é necessário que a instância do servidor VPN que gere a ligação entre *road warrior* e Coimbra faça uso do plugin "`usr/lib/openvpn/openvpn-plugin-auth-pam.so`" ao contrário da instância entre o servidor de Coimbra e Lisboa.

Como exemplo da configuração destes servidores VPN, no snippet seguinte é apresentada a configuração do servidor VPN de Coimbra "`server-warrior.conf`", com a seguinte informação (sendo a configuração semelhante como referido anteriormente):

```
local 192.168.172.128
port 1194
proto udp
dev tun1
ca ca.crt
cert coimbra.crt
key coimbra.key
dh dh2048.pem
server 10.10.0.0 255.255.255.0
ifconfig-pool-persist /var/log/openvpn/ipp.txt
push "route 10.8.0.0 255.255.255.0"
keepalive 10 120
tls-auth ta-lisboa.key 0
cipher AES-256-CBC
comp-lzo
user nobody
group nogroup
verb 3
explicit-exit-notify 1
plugin /usr/lib/openvpn/openvpn-plugin-auth-pam.so openvpn
script-security 2
tls-verify /etc/openvpn/ocsp.sh
```

Por último, para testar o estabelecimento de ligação recorrendo ao túnel VPN entre o servidor (Coimbra) e cliente (Lisboa neste caso), sendo semelhante para o caso do *road warrior*, corremos o seguinte comando:

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

Estando tudo funcional, procedemos à criação de serviços que são geridos pelo gestor de serviços “systemd” e que permitem aquando o login na máquina o início automático de ambas as instâncias de servidor VPN.

```
sudo systemctl enable openvpn-server@server-lisboa.service
sudo systemctl enable openvpn-server@server-warrior.service
sudo systemctl start openvpn-server@server-lisboa.service
sudo systemctl start openvpn-server@server-warrior.service
```

NOTA: Para inserir todas as passwords necessárias para iniciar estes serviços pode ser feita chamando o seguinte comando a partir da shell ou tty.

```
sudo systemd-tty-ask-password-agent
```

4.2.5. Autenticação TOTP (por Google Authenticator)

O serviço de TOTP fornecido pelo Google Authenticator vai ter associado a um utilizador responsável por gerir toda a autenticação de utilizadores (clientes) junto dos serviços da Google. Como tal, criamos um utilizador (e um grupo) chamado “*gauth”* para essa gestão. Assim, executamos os seguintes comandos para esse mesmo fim.

```
sudo addgroup gauth
sudo useradd -g gauth gauth
sudo mkdir /etc/openvpn/google-authenticator
sudo chown gauth:gauth /etc/openvpn/google-authenticator
sudo chmod 0700 /etc/openvpn/google-authenticator
```

A fim de nos autenticarmos com OTP, criámos o seguinte utilizador que depois usámos no *road warrior*, com *username* “`warrior`” e palavra-passe “`sti2022`”.

```
sudo useradd -d /home/warrior -s /bin/false warrior
sudo passwd warrior sti2022
```

Para permitir o funcionamento do plugin de TOPT, na pasta “`/etc/pam.d`” criou-se um ficheiro chamado “`openvpn`” com a seguinte configuração:

```
auth requisite /lib/x86_64-linux-gnu/security/pam_google_authenticator.so
secret=/etc/openvpn/google-authenticator/${USER} user=gauth forward_pass
```

Por último, a seguinte linha de comando gera um *QR code* que vai ser associado ao utilizador criado anteriormente denominado “`warrior`”, que obriga a um scan com o telemóvel com a aplicação do Google Authenticator.

```
su -c "google-authenticator -t -d -r3 -R30 -f -l 'OpenVPN Server'
-s/etc/openvpn/google-authenticator/warrior" - gauth
```

4.2.6. Configurações adicionais

De forma a não dar erro quando se usa o comando para configurar o OpenVPN, os seguintes comandos são necessários para criar o *serial number* e o ficheiro "`index.txt`" onde serão armazenados todos os *CSR*s.

```
sudo touch /etc/pki/CA/index.txt /etc/pki/CA/serial /etc/pki/CA/crlnumber
sudo sh -c 'echo 01 > /etc/pki/CA/serial'
sudo sh -c 'echo 01 > /etc/pki/CA/crlnumber'
```

Além disso, no ficheiro "`/etc/hosts`", foram adicionadas as seguintes linhas:

```
10.10.0.6     www.apache.lisboa.pt
192.168.172.128 ocsp.coimbra.pt
```

Também foi necessário permitir o reencaminhamento de IPs, para permitir o redirecionamento de pacotes do servidor de Lisboa para o *road warrior* e vice-versa. Como tal, no ficheiro "`sysctl.conf`", o parâmetro "`net.ipv4.ip_forward`" foi posto a "`1`" para permitir essa funcionalidade.

```
net.ipv4.ip_forward=1
```

4.3. Servidor de Lisboa

4.3.1. Configuração do servidor Apache

Os ficheiros "`default-ssl.conf`" e "`000-default.conf`", presentes na pasta cuja diretoria é "`/etc/apache2/sites-available`" são *templates* de suporte para a configuração do Apache. Estes foram *disabled* e os criados ("`apache-ssl.conf`" e "`apache.conf`") foram *enabled*, isto na mesma pasta mencionada anteriormente. Este passo garantiu que os ficheiros de configuração criados prevaleciam sobre os *default*. Os comandos usados foram os seguintes:

```
sudo a2dissite default-ssl
sudo a2dissite 000-default
sudo a2ensite apache-ssl
sudo a2ensite apache
```

As configurações presentes nos ficheiros referidos anteriormente são:

- No ficheiro "`apache.conf`":

```
<VirtualHost *:80>
        ServerName www.apache.lisboa.pt
        Redirect / https://www.apache.lisboa.pt
        ServerAdmin apache@dei.ist.pt
</VirtualHost>
```

- No ficheiro "`apache-ssl.conf`":

```
<IfModule mod_ssl.c>
        <VirtualHost *:443>
                ServerAdmin apache@dei.ist.pt
                DocumentRoot /var/www/apache/public_html
                ServerName www.apache.lisboa.pt

                ErrorLog ${APACHE_LOG_DIR}/error.log
                CustomLog ${APACHE_LOG_DIR}/access.log combined

                SSLEngine on
                SSLCertificateFile    /etc/pki/CA/certs/apache.crt
                SSLCertificateKeyFile /etc/pki/CA/private/apache.key
                SSLCACertificateFile /etc/pki/CA/certs/cacert.pem

                SSLOCSPEnable on
                SSLOCSPDefaultResponder http://ocsp.coimbra.pt:81
                SSLOCSPOverrideResponder off

                SSLVerifyClient require
                SSLVerifyDepth 10
        </VirtualHost>
</IfModule>
```

NOTAS:

1. Cada um destes ficheiros contêm as configurações para quando um utilizador se liga ao servidor Apache através do protocolo *http* ou usando o *https* que faz uso do módulo de SSL. No caso do utilizador de ligar ao servidor Apache através da configuração *http*, achamos mais correto forçar o utilizador que se liga através deste protocolo a usar SSL e assim, redirecionamos esse *user* para a mesma página mas usando *https.* Também para fins de teste mudamos a página apresentada ao cliente aquando a sua conexão relativamente à página *default* do Apache.
2. Para a configuração do Apache tivemos que referenciar os certificados necessários para este, emitidos junto da nossa CA privada (servidor de Coimbra) e que se encontram nos *paths* que são possíveis observar na nossa configuração.
3. Em termos do browser utilizado (Firefox), relembrar que temos de importar os certificados necessários para que seja possível estabelecer a ligação a este servidor *web*, nomeadamente certificado da CA, e certificado do utilizador no formato *"`PCKS#12`".*
4. Não é suposto o servidor de Apache ser acedido através da rede *"Internet"* estando apenas acessível através do túnel VPN. No entanto, não efetuamos nenhum mecanismo de *firewall* e controle de acesso quando um *host* tenta estabelecer ligação ao servidor através da interface "`ens33`". Temos consciência dessa situação e apenas não fizemos isso por uma questão de facilidade. Nos testes que fazemos assumimos que quando existe uma ligação ao Apache é utilizado o IP associado ao nome do servidor

"`www.apache.lisboa.pt`" instalado no DNS de cada uma das máquinas (ficheiro "`/etc/hosts`") e não o IP da interface "`ens33`".

4.3.2. Configuração de OpenVPN

Como foi dito anteriormente, o servidor de Lisboa foi tomado como cliente na ligação por túnel VPN ao servidor de Coimbra. Na diretoria fie "`/etc/openvpn/client`" foi criado um ficheiro chamado "`client.conf`" com a seguinte configuração:

```
client
dev tun1
proto udp
remote coimbra-vpn.dei.uc.pt 1193
resolv-retry infinite
nobind
user nobody
group nogroup
persist-key
persist-tun
ca ca.crt
cert lisboa-client.crt
key lisboa-client.key
tls-auth ta.key 1
cipher AES-256-CBC
comp-lzo
verb 3
```

Para estabelecer a ligação entre o servidor (Coimbra) e cliente (Lisboa), corre-se:

```
sudo openvpn --config /etc/openvpn/client/client.conf
```

Estando tudo funcional, procedemos à criação de serviços que são geridos pelo gestor de serviços "systemd" e que permitem aquando o login na máquina o início automático do cliente VPN.

```
sudo systemctl enable openvpn-client@client.service
sudo systemctl start openvpn-client@client.service
```

4.3.3. Configurações adicionais

A única configuração adicional foi feita no ficheiro "`/etc/hosts`", onde adicionamos a seguinte linha :

```
10.10.0.6    www.apache.lisboa.pt
```

4.4. Road Warrior

4.4.1. Configuração de OpenVPN

À semelhança do que foi feito para o servidor de Lisboa e como foi dito anteriormente, o *road warrior* foi tomado como cliente na ligação por túnel VPN ao servidor de Coimbra. A principal diferença entre o ficheiro de configuração do OpenVPN entre os dois clientes (servidor de Lisboa e *road warrior*) é a obrigatoriedade de requerer ao *road warrior* uma TOTP sendo isto visível na configuração através da linha "`auth-user-pass`" . Assim, na diretoria "`/etc/openvpn/client`" foi criado o ficheiro chamado "`client.conf`" com a seguinte configuração:

```
client
dev tun
proto udp
remote coimbra-vpn.dei.uc.pt 1194
resolv-retry infinite
nobind
user nobody
group nogroup
persist-key
persist-tun
ca ca.crt
cert warrior-client.crt
key warrior-client.key
tls-auth ta.key 1
cipher AES-256-CBC
comp-lzo
verb 3
auth-user-pass
```

Tal como foi feito anteriormente, para estabelecer a ligação entre o servidor (Coimbra) e cliente (Road Warrior), corre-se:

```
sudo openvpn --config /etc/openvpn/client/client.conf
```

Estando tudo funcional, procedemos à criação de serviços que são geridos pelo gestor de serviços "systemd" e que permitem, aquando o login na máquina, o início automático do cliente VPN.

```
sudo systemctl enable openvpn-client@client.service
sudo systemctl start openvpn-client@client.service
```

NOTA: Ao iniciar o serviço OpenVPN, é necessário o *road warrior* se identificar. Para tal, usaram-se os dados criados previamente, quando se configurou o servidor Apache. O *username* é o mesmo (no exemplo utilizado este é "`warrior`") e a palavra-passe será a criada anteriormente concatenada com o código de seis dígitos presentes no Google Authenticator (se esse código for 123456, a palavra-passe que se tem de utilizar é sti2022123456).

4.4.2. Configurações adicionais

À semelhança do que tinha sido feito anteriormente, no ficheiro "`/etc/hosts`", foram adicionadas as seguintes linhas:

```
192.168.172.128   coimbra-vpn.dei.uc.pt
192.168.172.128   ocsp.coimbra.pt
10.10.0.6         www.apache.lisboa.pt
```

5. Testes

Início	Destino	Estado	Ferramenta
Road Warrior (tun0)	Coimbra (tun0)	OK	Ping, Traceroute, Wireshark
Road Warrior (tun0)	Lisboa (tun1)	OK	Ping, Traceroute, Wireshark
Coimbra (tun0)	Road Warrior (tun0)	OK	Ping, Traceroute, Wireshark
Coimbra (tun1)	Lisboa (tun1)	OK	Ping, Traceroute, Wireshark
Lisboa (tun1)	Coimbra (tun1)	OK	Ping, Traceroute, Wireshark
Lisboa (tun1)	Road Warrior (tun0)	OK	Ping, Traceroute, Wireshark
Road Warrior (ens33)	Coimbra (ens33)	OK	Wireshark
Coimbra (ens33)	Lisboa (ens33)	OK	Wireshark
Road Warrior	Lisboa	OK	Firefox (acesso ao Apache)
Coimbra	Lisboa	OK	Firefox (acesso ao Apache)
Lisboa	Lisboa	OK	Firefox (acesso ao Apache)

6. Conclusões

Neste trabalho conseguimos reproduzir o cenário de rede que nos foi proposto estando todos os serviços funcionais. Tivemos a oportunidade de estudar processos que permitem a garantia de segurança entre sistemas e implementar estes mecanismos. Encontramos diversos obstáculos no processo de realização deste trabalho que pensamos ter sido dificultado pelo facto de a documentação disponível ser muitas vezes de difícil leitura para um utilizador que não tem muita experiência na área de redes. Por outro lado, a documentação de fácil compreensão é limitada e requer várias horas de pesquisa para encontrar.

Em termos de implementação do cenário prático, encontramos vários problemas de cariz técnico que não foram referidos nas aulas práticas e que apenas com o auxílio do docente conseguimos resolver. No entanto, conseguimos atingir os objetivos propostos.

7. Referências

- Linux Man Pages: openssl openvpn apache google-authenticator
- Segurança Prática em Sistemas e Redes com Linux, Jorge Granjal, editor: FCA, isbn: 9789727228652
- https://www.junmajinlong.com/files/openvpn-cookbook-2nd.pdf
- https://ulimit.nl/wp-content/uploads/2019/08/Extending-a-Debian-OpenVPN-server-with-Multi-Factor-Authentication-via-Google-Authenticator.pdf
- https://openvpn.net
- https://httpd.apache.org/docs/2.4/ssl/ssl_howto.html
- https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2&hl=en
- https://wiki.debian.org/Self-Signed_Certificate
- https://openvpn.net/community-resources/hardening-openvpn-security/