1 2 9 0

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Practical Assignment #1

CA + OpenVPN

Mestrado em Segurança Informática
Ano letivo 2022/2023

Inês Martins Marçal **Nº: 2019215917**
João Carlos Borges Silva **Nº: 2019216753**

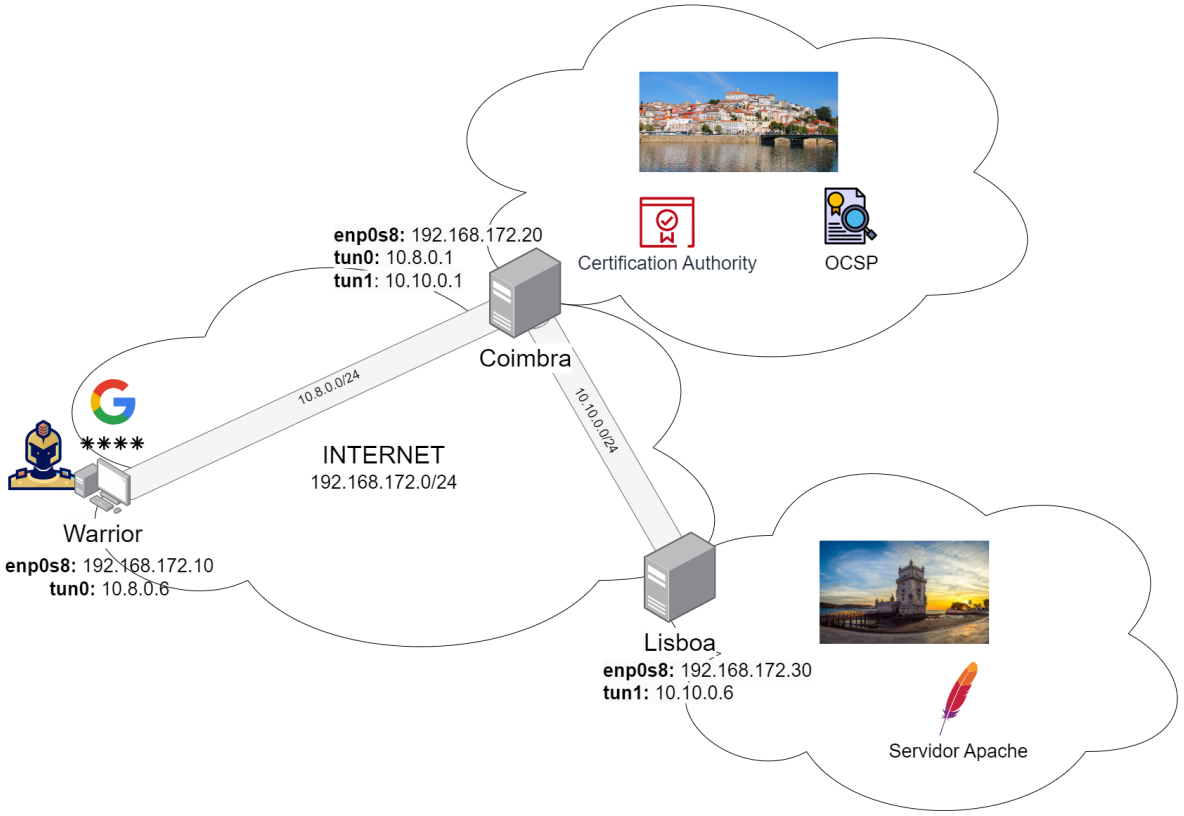
Índice

Introdução

Este trabalho foi realizado no âmbito da cadeira de Segurança em Tecnologias da Informação e visa a implementação de um cenário entre 3 Máquinas Virtuais utilizando diversos elementos estudados durante as aulas como certificados X.509 e túneis VPN. Os objetivos que se pretende alcançar através da realização deste trabalho são os seguintes:

- Configuração de 2 túneis VPN: um entre a rede de Coimbra e a rede de Lisboa e outro entre o Road Warrior e a rede Coimbra.
- Configuração de uma *CA* (*Certification Authority*) para criação e distribuição de certificados entre os diferentes pontos do cenário.
- Configuração de um servidor *OCSP* de modo a gerir os certificados criados pela CA: poder revogar certificados, assim como verificar a sua validade.
- Configuração de Google Authenticator para reforçar a autenticação do Road Warrior ao túnel estabelecido entre si e a rede de Coimbra.
- Configuração de um servidor Apache http (https) de modo que o mesmo possa ser acedido por quem contenha um certificado válido assinado pela CA criada, mais especificamente o Road Warrior.

O cenário anteriormente referido encontra-se ilustrado no seguinte esquema (tendo sido incluído no mesmo não só os IPs atribuídos às diferentes máquina virtuais, como também os IPs utilizados nos túneis VPN):



Tal como podemos observar temos 3 pontos de comunicação (3 Máquinas Virtuais), todos interligados através da rede Internet (192.168.172.0/24). A cada Máquina Virtual foi, portanto, atribuído um IP desta rede: Warrior - 192.168.172.10, Coimbra - 192.168.172.20 e

Lisboa - 192.168.172.30. É ainda importante relembrar que tanto Coimbra como Lisboa possuem as suas respectivas redes internas associadas a diferentes serviços: Coimbra contém a *CA* e o servidor *OCSP* e é a partir dos mesmos que toda a gestão dos certificados deverá ser efetuada, já Lisboa irá conter o servidor *Apache* que, por sua vez, deverá poder ser acessado por quem esteja ligado aos túneis *VPN*.

De modo a estabelecer uma comunicação segura entre os diferentes pontos do cenário, a mesma deverá ser efetuada por meio dos túneis *VPN* que interligam os mesmos: 1 túnel entre Warrior - Coimbra, com IP 10.8.0.1 do lado de Coimbra e 10.8.0.6 do lado do Warrior; 1 túnel entre Coimbra - Lisboa, com IP 10.10.0.1 do lado de Coimbra e 10.10.0.6 do lado de Lisboa.

Software utilizado

Durante a realização deste trabalho procedeu-se à instalação de *software* tanto a nível da virtualização das Máquinas Virtuais como de algum *software* necessário a integrar nas mesmas. Para virtualização das máquinas virtuais foi utilizado a *Oracle VM Box,* tendo-se optado pela distribuição *Lubuntu 22.04*, pelo que os comandos demonstrados ao longo do trabalho seguem esta mesma especificação.

Com vista a configurar o cenário proposto foi instalado o seguinte *software* em cada Máquina Virtual:

- openssl (já instalado por *default*): para geração da *CA* e certificados a serem utilizados, assim como para a implementação do servidor *OCSP*
- openvpn: para criação dos túneis *VPN* de modo a estabelecer uma conexão segura entre os diferentes pontos do cenário dentro da rede da *Internet*
- net-tools: utilizado para verificar algumas configurações relacionadas com os IPs configurados.

Para o efeito foi utilizado o seguinte comando:

```
/home/stipl1:~$ sudo apt install openssl openvpn
```

Algum do *software* instalado corresponde às especificações pretendidas para cada Máquina Virtual, pelo que poderá ser diferente:

- Coimbra
  - libpam-google-authenticator: servirá mais tarde para a implementação de *2FA* durante a autenticação do Road Warrior no túnel *VPN*
  - libqrencode3 (já instalado por default): novamente, utilizado durante a configuração de *2FA* para que seja possível gerar um *QRCode* para o utilizador *scanear* através da aplicação *Google Authenticator*

Para o efeito foi utilizado o seguinte comando:

```
/home/stipl1:~$ sudo apt install libpam-google-authenticator libqrencode3
```

- Lisboa
  - *Apache2*: o nome é bastante intuitivo, pelo que servirá para a implementação de um servidor *Apache*

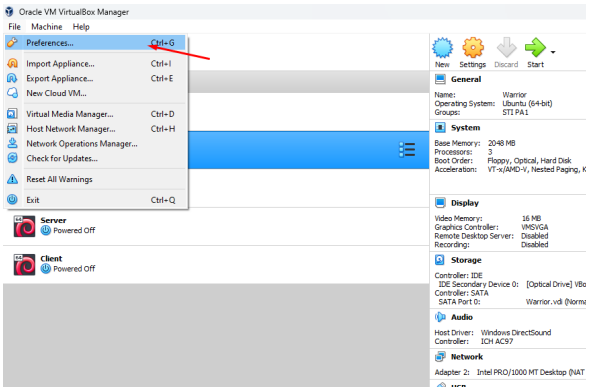
Para o efeito foi utilizado o seguinte comando:

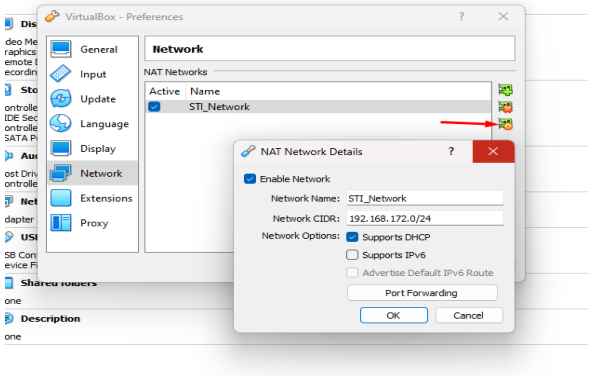
```
/home/stipl1:~$ sudo apt install apache2
```

Além do software instalado nas máquinas virtuais foi ainda utilizada a aplicação *Google Authenticator* no telemóvel.

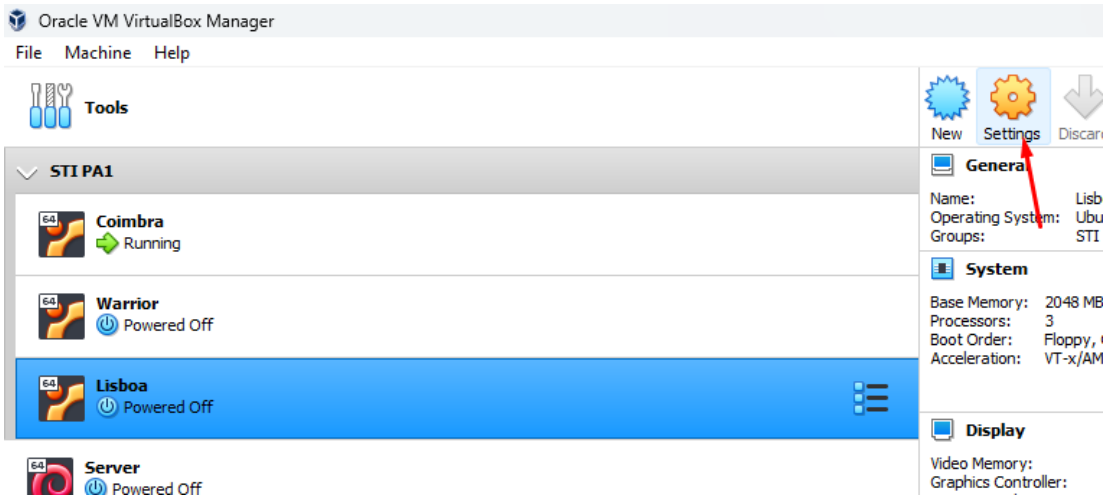
Configuração de *IPs* das Máquinas Virtuais

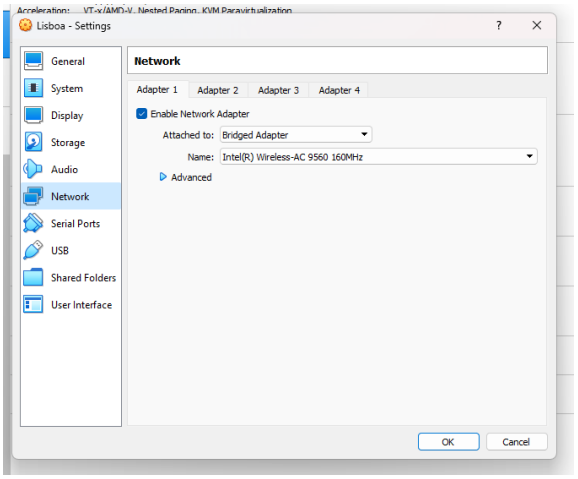
De modo a assegurar que todas as Máquinas Virtuais se encontrassem dentro da rede de Internet foram necessárias algumas configurações ao nível de adaptadores de rede. Para tal, em primeiro lugar foi necessário criar adaptadores de rede de dois tipos: *Bridged Adapter* (com vista a estabelecer ligação ao *WiFi* e conseguir instalar o *software* necessário) e *NAT Network* (para configurar a rede da Internet). Antes de configurar o 2º adaptador foi ainda necessário criar uma rede *NAT* seguindo os seguintes passos:

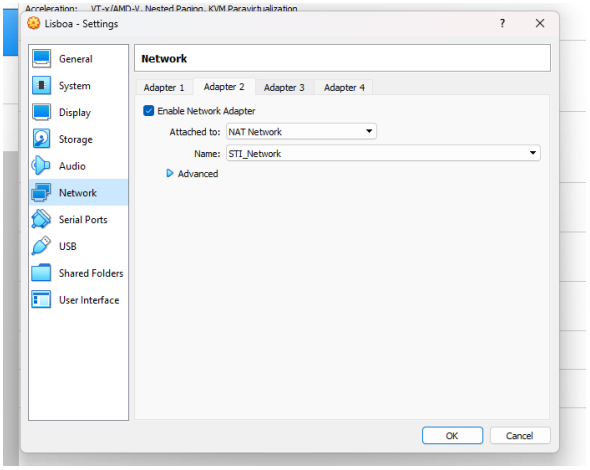




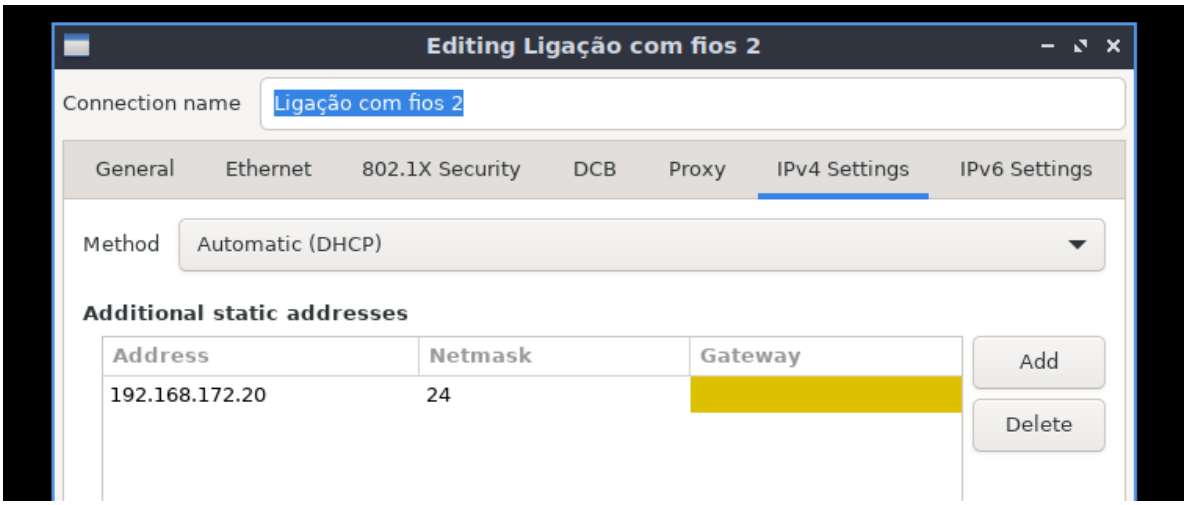
De seguida define-se os adaptadores das Máquinas Virtuais da seguinte maneira:







Finalmente já dentro de cada Máquina Virtual configura-se o *IP* desejado:



Criação da Certification Authority

A criação de uma *CA* capaz de gerar corretamente certificados *X.509* que, por sua vez, autentiquem todas as partes envolvidas durante a comunicação das mesmas constituiu um ponto fulcral neste trabalho. Além da criação dos certificados, esta *CA* deverá ser capaz de poder revogar certificados, anulando, assim, a sua validade e confiança para com os restantes certificados. No final de contas, os certificados criados irão ser utilizados em diversos pontos do nosso cenário: estabelecimento de túneis *VPN* e acesso ao servidor *Apache*.

Os passos para a criação de uma *CA* são os seguintes: emissão de *Certificate Signing Request (CSR)* e posterior emissão de um certificado *self-signed* pela *CA* a criar, é igualmente necessário criar uma chave para que seja possível executar estes dois primeiros. Contudo, antes de qualquer uma destas etapas será preciso proceder a algumas configurações a nível do ficheiro "/etc/ssl/openssl.conf":

- É necessária a definição de uma diretoria onde a nossa *CA* irá operar e indicação da mesma no ficheiro de configurações do *openssl*, no nosso caso o nome escolhido foi "/etc/pki/CA". Além desta, foram definidas algumas pastas adicionais dentro desta diretoria: private (para guardar as chaves criadas), csrs (para guardar os *CSRs* criados) e certs (para guardar os certificados criados). Relativamente a esta parte inicial, o nosso ficheiro de configurações apresenta os seguintes parâmetros:

```
[ CA_default ]
dir             = /etc/pki/CA           # Alterado
certs           = $dir/certs
crl_dir         = $dir/crl
database        = $dir/index.txt
new_certs_dir   = $dir/newcerts
certificate     = $dir/certs/ca.crt     # Alterado
serial          = $dir/serial
crlnumber       = $dir/crlnumber
crl             = $dir/crl.pem
private_key     = $dir/private/ca.key   # Alterado
```

```
x509_extensions = usr_cert
name_opt        = ca_default
cert_opt        = ca_default

default_days    = 365
default_crl_days= 30
default_md      = default
preserve        = no

policy          = policy_match
```

Seguem-se os comandos necessários para criar as pastas e ficheiros para a estrutura inicial da *CA*:

```
/home/stipl1:~$ sudo mkdir /etc/pki
/home/stipl1:~$ sudo mkdir /etc/pki/CA
/home/stipl1:~$ sudo mkdir /etc/pki/CA/private
/home/stipl1:~$ sudo mkdir /etc/pki/CA/csrs
/home/stipl1:~$ sudo mkdir /etc/pki/CA/certs
/home/stipl1:~$ sudo mkdir /etc/pki/CA/newcerts
/home/stipl1:~$ sudo mkdir /etc/pki/CA/crl
/home/stipl1:~$ sudo touch /etc/pki/CA/index.txt
/home/stipl1:~$ sudo bash -c "echo 01 > /etc/pki/CA/serial"
/home/stipl1:~$ sudo bash -c "echo 01 > /etc/pki/CA/crlnumber"
```

- Foram também feitas configurações adicionais de modo a que os certificados criados passassem a suportar *OCSP*. Para o efeito foram efetuados ajustes na *stanza* “usr_cert” e adicionando uma nova chamada “v3_OCSP”:

```
[ usr_cert ]
basicConstraints                = CA:FALSE
subjectKeyIdentifier            = hash
authorityKeyIdentifier          = keyid,issuer
authorityInfoAccess             = OCSP;URI:http://ocsp_stipl1.pt:8081

[ v3_OCSP ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = OCSPSigning
```

Enquanto que a *stanza* “usr_cert” se refere às configurações de um certificado comum (onde se passou a ter informação do respectivo servidor OCSP), a *stanza* “v3_OCSP” permitirá adicionar ao certificado do servidor OCSP uma extensão que o identifique como tal.

- Finalmente e de modo a que os certificados usados na criação dos túneis estejam válidos foram adicionadas mais 2 *stanzas* a este ficheiros de configurações: “server” e “client”. A primeira é necessária para certificados que venham a ser utilizados

como servidor na ligação do túnel *VPN*, enquanto que a segunda é usada pelo cliente desse túnel, permitindo assim que o mesmo reconheça a identidade de cada parte (se é servidor ou cliente). Segue-se a configuração acrescentada:

```
[ server ]
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
extendedKeyUsage = serverAuth
keyUsage = digitalSignature, keyEncipherment
authorityInfoAccess = OCSP;URI:http://ocsp_stipl1.pt:8081

[ client ]
basicConstraints = CA:FALSE
nsCertType = client
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid, issuer:always
extendedKeyUsage = clientAuth
keyUsage = digitalSignature, keyEncipherment
authorityInfoAccess = OCSP;URI:http://ocsp_stipl1.pt:8081
```

Após se ter estas configurações definidas no ficheiro de "/etc/ssl/openssl.conf" estará tudo pronto para criação da *CA* em Coimbra. Em primeiro lugar gera-se a chave privada ("ca.key") responsável pelo *CSR* e certificado da *CA*, para tal utiliza-se o algoritmo *Triple DES* definindo o tamanho em 2048 bits, sendo de seguida necessária inserção de uma *passphrase*:

```
/home/stipl1:~$ cd /etc/pki/CA
/etc/pki/CA:~$ sudo openssl genrsa -out private/ca.key -des3 2048
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

Posteriormente, é necessário gerar um CSR ("ca.csr") para a CA a ser criada, utilizando a chave privada que acabou de ser criada. Para tal, após a execução do seguinte comando é necessária a introdução da *passphrase* anteriormente definida, bem como informação adicional acerca deste certificado como: País, Estado, Localidade, Organização, Unidade da Organização, Common Name e Email.

```
/etc/pki/CA:~$ sudo openssl req -new -key private/ca.key -out csrs/ca.csr
Enter pass phrase for ca.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
```

```
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]: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) []:[Coimbra]:CA
Email Address []:coimbra@mail.com

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

Finalmente é gerado um certificado "self-signed" ("ca.crt") com uma duração de 5 anos (1825 dias) através da chave privada e CSR anteriormente criados, sendo necessário, novamente, introduzir a *passphrase* definida:

```
/etc/pki/CA:~$ sudo openssl x509 -req -days 1825 -in csrs/ca.csr -out certs/ca.crt -signkey private/ca.key
Enter pass phrase for ca.key:
Certificate request self-signature ok
subject=C = PT, ST = Coimbra, L = Coimbra, O = UC, OU = DEI, CN = [Coimbra]:CA, emailAddress = coimbra@mail.com
```

Após todos estes passos obtém-se uma CA pronta para começar a gerar os restantes certificados e efetuar a sua gestão.

Criação do servidor *OCSP*

Após as alterações efetuadas no ficheiro de configurações "/etc/ssl/openssl.conf", no sentido de se criar o servidor OCSP será necessário gerar um certificado próprio para esse efeito, seguindo, assim, os mesmos passos a quando a criação do certificado para a CA. Neste caso, o comando final terá uma notação diferente, já que o certificado da CA era *self-signed* e os certificados que iremos passar a gerar serão assinados pela CA:

```
/etc/pki/CA:~$ sudo openssl ca -in csrs/ocsp.csr -cert certs/ca.crt -keyfile private/ca.key -out certs/ocsp.crt -extensions v3_OCSP
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for private/ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 2 (0x2)
        Validity
            Not Before: Mar  8 17:04:01 2023 GMT
            Not After : Mar  7 17:04:01 2024 GMT
        Subject:
            countryName               = PT
            stateOrProvinceName       = Coimbra
            organizationName          = UC
            organizationalUnitName    = DEI
            commonName                = [Coimbra]:OCSP
            emailAddress              = coimbra@mail.com
```

```
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Extended Key Usage:
                OCSP Signing
Certificate is to be certified until Mar  7 17:04:01 2024 GMT (365 days)
Sign the certificate? [y/n]: y

1 out of 1 certificate requests certified, commit? [y/n] y
Write out database with 1 new entries
Data Base Updated
```

É de reparar que foi utilizada a extensão anteriormente definida na *stanza* "v3_OCSP" no ficheiro de configurações do *openssl* e se repararmos essa mesma informação é refletida no parâmetro *x509v3 Extended Key Usage,* onde aparecerá escrito *OCSP Signing*. Posto isto, teremos um certificado capaz de suportar um servidor OCSP. Este servidor poderá ser iniciado através do seguinte comando, sendo necessário introduzir de seguida a *passphrase* definida:

```
/etc/pki/CA:~$ sudo openssl ocsp -index index.txt -port 8081 -rsigner certs/ocsp.crt -rkey private/ocsp.key -CA certs/ca.crt
-text -out log.txt
ACCEPT 0.0.0.0:8081 PID=3063
Enter pass phrase for private/ocsp.key:
ocsp: waiting for OCSP client connections...
```

Com isto, teremos o servidor OCSP a correr segundo as configurações anteriormente definidas, isto é, no endereço "http://ocsp_stipl1.pt" no porto 8081. A partir de agora, sempre que seja preciso efetuar algum tipo de verificação relativamente ao estado de um certificado deverão ser indicados estes endereços e portos.

Para validação de certificados através deste servidor durante o estabelecimento de túneis foi utilizado o seguinte *script*:

```
#!/bin/sh
ocsp_url="http://ocsp_stipl1.pt:8081/"
issuer="/etc/pki/CA/certs/ca.crt"
nonce="-nonce"
verify="/etc/pki/CA/certs/ca.crt"

check_depth=0
cur_depth=$1     # this is the *CURRENT* depth
common_name=$2   # CN in case you need it

err=0
if [ -z "$issuer" ] || [ ! -e "$issuer" ]; then
  echo "Error: issuer certificate undefined or not found!" >&2
  err=1
fi

if [ -z "$verify" ] || [ ! -e "$verify" ]; then
  echo "Error: verification certificate undefined or not found!" >&2
  err=1
fi

```

```
if [ -z "$ocsp_url" ]; then
  echo "Error: OCSP server URL not defined!" >&2
  err=1
fi

if [ $err -eq 1 ]; then
  echo "Did you forget to customize the variables in the script?" >&2
  exit 1
fi

if [ $check_depth -eq -1 ] || [ $cur_depth -eq $check_depth ]; then
  eval serial="\$tls_serial_${cur_depth}"
  if [ -n "$serial" ]; then
    status=$(openssl ocsp -issuer "$issuer" \
                     "$nonce" \
                     -CAfile "$verify" \
                     -url "$ocsp_url" \
                     -serial "${serial}" 2>&1)

    if [ $? -eq 0 ]; then
      if echo "$status" | grep -Eq "(error|fail)"; then
          exit 1
      fi
      if echo "$status" | grep -Eq "^${serial}: good"; then
        if echo "$status" | grep -Eq "^Response verify OK"; then
            exit 0
        fi
      fi
    fi
  fi
  exit 1
fi
```

O mesmo permitirá verificar se o certificado apresente 1 de 3 estados diferentes:

- *good*: pertence à mesma CA e não foi revogado
- *unknown*: não apresenta a mesma CA e portanto é desconhecida qual informação acerca do mesmo
- *revoked*: pertence à mesma CA e já foi revogado

Emissão de certificados

Até ao momento, como podemos observar, foram criados 2 certificados, 1 para CA e outro para o servidor OCSP, mas no final de contas serão necessários 7 certificados e por conseguinte 7 chaves privadas e 7 CSRs. Deste modo, a distribuição de certificados deverá ser a seguinte:

- Coimbra:
  - ca.crt: certificado gerado para suportar CA, usado durante a criação dos restantes certificados.
  - ocsp.crt: certificado gerado para suportar o servidor OCSP.

    - coimbra.crt: certificado a ser usado na configuração dos túneis entre Warrior-Coimbra e Coimbra-Lisboa. Será um certificado do tipo servidor nos 2 túneis.
- Warrior:
    - warrior.crt: certificado a ser usado na configuração do túnel entre Warrior-Coimbra. Será um certificado do tipo cliente no túnel em que se encontra inserido.
    - apacheuser.crt: certificado a ser usado como autenticação no acesso ao servidor Apache localizado em Lisboa.
- Lisboa:
    - lisboa.crt: certificado a ser usado na configuração do túnel entre Coimbra-Lisboa. Será um certificado do tipo cliente no túnel em que se encontra inserido.
    - apache.crt: certificado utilizado para suportar o protocolo HTTPS no servidor Apache da rede de Lisboa.

E, obviamente, com tantos certificados claramente haverá diferenças nos mesmos, nomeadamente nos parâmetros de identificação utilizados (País, Estado, Localidade, ..) no CSR. Mas a maior diferença na geração dos diferentes certificados surge nas diferentes extensões utilizadas, relembrando que durante a configuração do ficheiro "/etc/ssl/openssl.conf" foram criadas duas *stanzas* para definir duas novas extensões (além de v3_OCSP): *server* e *client*.

É, portanto, preciso entender que entidades serão o servidor e quais serão o cliente nos seus respectivos túneis VPN para que se possa efetuar a atribuição das diferentes extensões. Para o cenário proposto foi definido que Coimbra seria servidor dos dois túneis, enquanto que Warrior e Lisboa seriam clientes dos seus respectivos túneis. Seguem-se os comandos utilizados para a criação destes mesmos certificados:

- Coimbra.crt:

```
/etc/pki/CA:~$ sudo openssl ca -in csrs/coimbra.csr -cert certs/ca.crt -keyfile private/ca.key -out certs/coimbra.crt -extensions server
                                                                    . . .
X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Cert Type:
                SSL Server
            Netscape Comment:
                OpenSSL Generated Server Certificate
            X509v3 Subject Key Identifier:
                50:29:54:51:23:2B:25:AE:22:DF:AD:8C:EB:45:69:75:2C:1D:61:DA
            X509v3 Authority Key Identifier:
                DirName:/C=PT/ST=Coimbra/L=Coimbra/O=UC/OU=DEI/CN=[Coimbra]:CA/emailAddress=coimbra@mail.com
                serial:10:44:A6:A5:78:0B:0E:90:36:9B:8C:42:34:01:AE:DD:09:82:CA:C2
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Key Usage:
                Digital Signature, Key Encipherment
            Authority Information Access:
                OCSP - URI:http://ocsp_stipl1.pt:8081
```

- Warrior.crt:

```
/etc/pki/CA:~$ sudo openssl ca -in csrs/warrior.csr -cert certs/ca.crt -keyfile private/ca.key -out certs/warrior.crt -extensions client
                                                                    . . .
X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Cert Type:
                SSL Client
            Netscape Comment:
                OpenSSL Generated Client Certificate
            X509v3 Subject Key Identifier:
                58:93:5B:03:75:1A:17:51:75:94:49:C8:27:18:71:59:85:F3:0D:C7
            X509v3 Authority Key Identifier:
                DirName:/C=PT/ST=Coimbra/L=Coimbra/O=UC/OU=DEI/CN=[Coimbra]:CA/emailAddress=coimbra@mail.com
                serial:42:7F:83:35:10:91:26:3A:2D:E5:82:72:49:F3:E9:C3:FF:40:27:A4
            X509v3 Extended Key Usage:
                TLS Web Client Authentication
            X509v3 Key Usage:
                Digital Signature, Key Encipherment
            Authority Information Access:
                OCSP - URI:http://ocsp_stipl1.pt:8081
```

- Lisboa.crt:

```
/etc/pki/CA:~$ sudo openssl ca -in csrs/lisboa.csr -cert certs/ca.crt -keyfile private/ca.key -out certs/lisboa.crt -extensions client
                                                                    . . .
X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Cert Type:
                SSL Client
            Netscape Comment:
                OpenSSL Generated Client Certificate
            X509v3 Subject Key Identifier:
                84:FD:B1:5F:35:99:4B:A0:CB:91:B4:6F:3C:D0:FF:4C:02:6E:54:B7
            X509v3 Authority Key Identifier:
                DirName:/C=PT/ST=Coimbra/L=Coimbra/O=UC/OU=DEI/CN=[Coimbra]:CA/emailAddress=coimbra@mail.com
                serial:42:7F:83:35:10:91:26:3A:2D:E5:82:72:49:F3:E9:C3:FF:40:27:A4
            X509v3 Extended Key Usage:
                TLS Web Client Authentication
            X509v3 Key Usage:
                Digital Signature, Key Encipherment
            Authority Information Access:
                OCSP - URI:http://ocsp_stipl1.pt:8081
```

Estes três certificados serão, portanto, utilizados durante a autenticação entre diferentes pontos do túnel, sendo de notar as extensões usadas: *server* e *client*. O uso das extensões poderá ser refletido em diversos pontos do certificado nomeadamente nos parâmetros: “Netscape Cert Type” e “X509v3 Extended Key Usage”. A partir do uso destas configurações, quando forem estabelecidos os túneis VPN, cada ponto poderá se autenticar como servidor ou como cliente, dependendo da entidade.

Finalmente, são ainda necessários mais dois certificados relacionados com o servidor *Apache* a ser criado, um para a ativação de HTTPS (“apache.crt”) e outro para a autenticação do cliente ao site (“apacheuser.crt”). Desta vez não será usada qualquer extensão durante a criação destes certificados, sendo os mesmos criados com as configurações *default* impostas pelo OpenSSL. De seguida encontram-se os comandos utilizados:

- Apache.crt:

```
/etc/pki/CA:~$ sudo openssl ca -in csrs/apache.csr -cert certs/ca.crt -keyfile private/ca.key -out certs/apache.crt
                                                                    . . .
X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Key Identifier:
                CE:3E:B9:AE:9F:68:AE:B1:80:F5:5D:F8:C2:8C:29:65:4D:AB:9A:BA
            X509v3 Authority Key Identifier:
                DirName:/C=PT/ST=Coimbra/L=Coimbra/O=UC/OU=DEI/CN=[Coimbra]:CA/emailAddress=coimbra@mail.com
                serial:42:7F:83:35:10:91:26:3A:2D:E5:82:72:49:F3:E9:C3:FF:40:27:A4
            Authority Information Access:
                OCSP - URI:http://ocsp_stipl1.pt:8081
```

- Apacheuser.crt:

```
/etc/pki/CA:~$ sudo openssl ca -in csrs/apacheuser.csr -cert certs/ca.crt -keyfile private/ca.key -out certs/apacheuser.crt
                                                                    . . .
X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Key Identifier:
                9A:96:38:AE:F2:84:0B:9A:D0:0B:20:60:F1:95:10:0A:CD:5D:E5:97
            X509v3 Authority Key Identifier:
                DirName:/C=PT/ST=Coimbra/L=Coimbra/O=UC/OU=DEI/CN=[Coimbra]:CA/emailAddress=coimbra@mail.com
                serial:42:7F:83:35:10:91:26:3A:2D:E5:82:72:49:F3:E9:C3:FF:40:27:A4
            Authority Information Access:
                OCSP - URI:http://ocsp_stipl1.pt:8081
```

Como podemos observar, estes certificados não apresentam qualquer extensão na sua configuração sendo apenas certificados comuns a utilizar para outros serviços, neste caso o *Apache*. Algo que podemos igualmente reparar nestes certificados será a presença de informação relacionada com o servidor OCSP que, por sua vez, surge devido às configurações inicialmente adicionadas no ficheiro "/etc/ssl/openssl.conf".

De modo a simplificar a emissão de certificados, todos os certificados foram criados na Máquina Virtual de Coimbra e por conseguinte todas as chaves e CSRs foram criados igualmente no mesmo sítio, mas num cenário real o mesmo não deveria acontecer. Num contexto real, tanto as chaves como CSRs deveriam ser criados pelas próprias máquinas virtuais, serem enviados para a CA e finalmente a CA responderia como o certificado correspondente.

Configuração dos túneis VPN (ainda sem 2FA)

Após se ter todos os certificados gerados será possível finalmente criar e configurar os túneis VPN entre as diferentes entidades do cenário considerado. Estes túneis permitirão uma comunicação segura, através da rede da Internet, entre as diferentes redes internas, apoiando-se na autenticidade e robustez dos mecanismos criptográficos utilizados (assinatura digital, HMAC, certificados X.509, …). O primeiro ponto para iniciar esta configuração foi a criação de pastas, onde se procedeu aos armazenamento dos devidos ficheiros:

```
/home/stipl1:~$ sudo mkdir /etc/pki/Openvpn
/home/stipl1:~$ sudo mkdir /etc/pki/Openvpn/configs
/home/stipl1:~$ sudo mkdir /etc/pki/Openvpn/private
```

As diretorias criadas serão apenas temporárias, posteriormente cada ficheiro irá para as suas máquinas/diretorias específicas. A pasta "/etc/pki/Openvpn/configs" servirá para criar os ficheiros de configuração dos túneis a criar (tanto ficheiros de servidor como cliente), enquanto que a pasta "/etc/pki/Openvpn/private" servirá para criar chaves necessárias na configuração dos túneis, nomeadamente *tls-auth keys* e parâmetros *Diffie Hellman*. Para criar estas últimas chaves foram utilizados os seguintes comandos:

```
/home/stipl1:~$ cd /etc/pki/Openvpn/private
/etc/pki/Openvpn/private:-$ sudo openssl dhparam -out dh2048_c_w.pem
/etc/pki/Openvpn/private:-$ sudo openssl dhparam -out dh2048_c_l.pem
/etc/pki/Openvpn/private:-$ sudo openvpn --genkey tls-auth ta_c_w.key
/etc/pki/Openvpn/private:-$ sudo openvpn --genkey tls-auth ta_c_l.key
```

De seguida, irá proceder-se à criação dos ficheiros de configuração para o estabelecimento dos túneis VPN, onde iremos ter ao todo 4 ficheiros de configuração: 2 do tipo servidor (*server-coimbra-warrior.conf* e *server-coimbra-lisboa.conf*) e 2 do tipo cliente (*client-warrior.conf* e *client-lisboa.conf)*. Como já referido, Coimbra será o servidor nos dois túneis VPN a serem criados e como tal as configurações nos dois ficheiros será praticamente igual, diferindo apenas no porto a utilizar, ficheiro de parâmetros de *Diffie Hellman,* ip do *server*, *tls-auth key* e rotas a qual se dá "push".

- server-coimbra-warrior.conf:

```
local 192.168.172.20
port 1194
proto udp
dev tun

ca ca.crt
cert coimbra.crt
key coimbra.key

dh dh2048_c_w.pem
server 10.8.0.0 255.255.255.0
ifconfig-pool-persist /var/log/openvpn/ipp.txt
push "route 10.10.0.0 255.255.255.0"
keepalive 10 120
tls-auth ta_c_w.key 0 # This file is secret
cipher AES-256-CBC
persist-key
persist-tun
status /var/log/openvpn/openvpn-status.log
verb 3
explicit-exit-notify 1

script-security 2
tls-verify /etc/pki/Openvpn/configs/OCSP_check.sh
```

- server-coimbra-lisboa.conf:

```
local 192.168.172.20
port 1195
proto udp
dev tun

ca ca.crt
cert coimbra.crt
key coimbra.key  # This file should be kept secret

dh dh2048_c_l.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_c_l.key 0 # This file is secret
cipher AES-256-CBC
persist-key
persist-tun
status /var/log/openvpn/openvpn-status.log
verb 3
explicit-exit-notify 1

script-security 2
tls-verify /etc/pki/Openvpn/configs/OCSP_check.sh
```

Olhando para ambos os ficheiros de configurações pode-se verificar que a utilização de alguns dos IPs já apresentados no cenário inicial, por exemplo na variável *local* e *server*, na primeira é indicado o IP local da máquina virtual (neste caso Coimbra), enquanto que na segunda é indicada a rede a qual o túnel irá pertencer. Quanto ao *push* das rotas efetuado, o mesmo será bastante intuitivo: no primeiro túnel (que corre na rede 10.8.0.0) damos a conhecer todas as rotas do outro túnel (rede 10.10.0.0), de modo similar o mesmo ocorrerá no segundo túnel, dando a conhecer as rotas da rede 10.8.0.0. No final de cada ficheiro foram também acrescentadas 2 linhas que permitem a autenticação dos certificados utilizados nos túneis a partir do servidor OCSP anteriormente definido. Esta verificação é efetuada através do *script OCSP_check.sh* (descrita na criação do servidor OCSP).

Passando para os ficheiros de configuração dos clientes de cada túnel (Coimbra e Lisboa) percebemos novamente que ambos são bastante semelhantes, diferindo em campos como porto a utilizar para estabelecer ligação ao túnel, certificados utilizados e *tls-auth key* usada.

- client-warrior.conf:

```
client
dev tun
proto udp
remote 192.168.172.20 1194
resolv-retry infinite
nobind
persist-key
persist-tun
```

```
ca ca.crt
cert warrior.crt
key warrior.key

remote-cert-tls server
tls-auth ta_c_w.key 1
cipher AES-256-CBC
verb 3
```

- client-lisboa.conf:

```
client
dev tun
proto udp
remote 192.168.172.20 1195
resolv-retry infinite
nobind
persist-key
persist-tun

ca ca.crt
cert lisboa.crt
key lisboa.key

remote-cert-tls server
tls-auth ta_c_l.key 1
cipher AES-256-CBC
verb 3
```

Nestas configurações, duas linhas para as quais se deverá prestar atenção são a primeira linha dos dois ficheiros e a linha que contém *remote-cert-tls server*. A primeira identifica as entidades que usarem estes ficheiros de configuração como cliente no seu túnel VPN, já a segunda permitirá verificar se o certificado da outra parte do túnel (servidor) realmente tem o *extended key usage* correto, sendo um passo importante na precaução contra potenciais ataques descritos em: http://openvpn.net/howto.html#mitm.

Configuração de Two-Factor Authentication

Sendo que o tráfego da comunicação por parte do *Road Warrior* vem de uma rede completamente desconhecida pelo cenário, é fulcral que se tenha atenção também ao mesmo, nomeadamente à sua autenticação. Posto isto, quando se efetuar a autenticação do *Road Warrior* no túnel VPN além da *passphrase* anteriormente definida, deverá ser imposto um mecanismo de 2FA no qual será pedido um *username* e *password*.

Inicialmente, será criado um utilizador responsável pela gestão deste mecanismo, o qual terá como nome “gauth” pertencendo ao grupo “gauth”, para tal são utilizados os seguintes comandos:

```
/home/stipl1:~$ sudo addgroup gauth
/home/stipl1:~$ sudo useradd -g gauth gauth
/home/stipl1:~$ sudo mkdir /etc/openvpn/google-authenticator
```

```
/home/stipl1:~$ sudo chown gauth:gauth /etc/openvpn/google-authenticator
/home/stipl1:~$ sudo chmod 0700 /etc/openvpn/google-authenticator
/home/stipl1:~$ sudo passwd gauth
```

O próximo passo passará pela criação de um ficheiro de configurações PAM para o OpenVPN, este ficheiro servirá para colocar em funcionamento o *plugin* do *Google Authenticator.*

```
/home/stipl1:~$ sudo touch /etc/pam.d/openvpn
```

- /etc/pam.d/openvpn:

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

Como referido anteriormente, para ativar o mecanismo de 2FA iremos ter credenciais como um *username* e uma *password*, portanto será criado um utilizador para esse efeito usando o seguinte comando:

```
/home/stipl1:~$ useradd -d /home/warrior -s /bin/false warrior
/home/stipl1:~$ passwd warrior
```

Este utilizador servirá apenas para o *Road Warrior* efetuar autenticação no túnel VPN e, como tal, o mesmo foi criado de modo a não ser possível efetuar autenticação na máquina virtual de Coimbra.

Com toda esta configuração posta em prática é hora de gerar o *QRCode* para o *user Road Warrior*. Através do próximo comando iremos gerar o tal *QRCode*, esta ação será feita pelo user *gauth,* anteriormente criado, e será necessária a introdução da *password* definida para o mesmo:

```
/home/stipl1:~$ su -c "google-authenticator -t -d -r3 -R30 -f -l 'OpenVPN Server' -s /etc/openvpn/goog
le-authenticator/testuser" - gauth
                      ...         [Imagem QR Code]         ...
Your new secret key is: 2PSUD2B46OWNERXAXOMXAYVHLE
Enter code from app (-1 to skip): 322532
Your emergency scratch codes are:
  93931235
  25208430
  98620000
  46589452
  32524599

By default, a new token is generated every 30 seconds by the mobile app.
In order to compensate for possible time-skew between the client and the server,
we allow an extra token before and after the current time. This allows for a
time skew of up to 30 seconds between authentication server and client. If you
experience problems with poor time synchronization, you can increase the window
from its default size of 3 permitted codes (one previous code, the current
code, the next code) to 17 permitted codes (the 8 previous codes, the current
code, and the 8 next codes). This will permit for a time skew of up to 4 minutes
between client and server.
Do you want to do so? (y/n) y
```

Após a execução do comando apresentado aparecerá um *QRCode,* o qual teremos de *scanear* utilizando a aplicação *Google Authenticator* instalada no telemóvel. Posteriormente, é necessário inserir o código presente na aplicação no terminal e, finalmente, responder com "y" à última questão.

O último passo para ter autenticação (com *2FA*) 100% funcional no túnel VPN será a inclusão de algumas linhas no final dos ficheiros de configuração do túnel em questão (*client-warrior.conf* e *server-coimbra-warrior.conf*):

- server-coimbra-warrior.conf:

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

- client-warrior.conf:

```
. . .
auth-user-pass
```

A partir deste momento, ambos os túneis VPN estarão devidamente configurados como requisitado, possuindo, assim, os mecanismos necessários para que seja possível oferecer uma maior segurança durante a comunicação entre as diferentes entidades.

Configuração do servidor Apache

Por fim, será efetuada a configuração do servidor *Apache* presente na máquina virtual de Lisboa. Como ponto de partida foram desabilitados os ficheiros de configuração *default* do Apache que se encontram na pasta /etc/apache2/sites-available: *000-default.conf* e *default-ssl.conf*.

```
/home/stipl1:~$ cd /etc/apache2/sites-available
/etc/apache2/sites-available:~$ sudo a2disside 000-default.conf
/etc/apache2/sites-available:~$ sudo a2disside default-ssl.conf
```

De seguida, irá ser criada, novamente, temporariamente uma pasta para armazenar o ficheiro de configuração *Apache* na máquina virtual de Coimbra (assim como aconteceu para os ficheiros de configuração do OpenVPN). Dentro dessa pasta irá proceder-se à criação do ficheiro *config.conf*.

```
/home/stipl1:~$ sudo mkdir /etc/pki/Apache
/home/stipl1:~$ sudo mkdir /etc/pki/Apache/configs
/home/stipl1:~$ cd /etc/pki/Apache/configs
/etc/pki/Apache/configs:~$ sudo touch config.conf
```

- config.conf:

```
<IfModule mod_ssl.c>
    <VirtualHost _default_:80>
        ServerName apache_stipl1.pt
        ServerAdmin admin@apache_stipl1.pt

        Redirect / https://apache_stipl1.pt
    </VirtualHost>
```

```
    <VirtualHost _default_:443>
        ServerName apache_stipl1.pt
        ServerAdmin admin@apache_stipl1.pt

        DocumentRoot /var/www/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
        SSLEngine on

        SSLCertificateFile  /etc/apache2/apache.crt
        SSLCertificateKeyFile /etc/apache2/apache.key
        SSLCACertificateFile /etc/apache2/ca.crt

        SSLVerifyClient require
        SSLVerifyDepth  10

        <FilesMatch "\.(cgi|shtml|phtml|php)$">
                SSLOptions +StdEnvVars
        </FilesMatch>
        <Directory /usr/lib/cgi-bin>
                SSLOptions +StdEnvVars
        </Directory>

        SSLOCSPEnable on
        SSLOCSPDefaultResponder http://ocsp_stipl1.pt:8081
        SSLOCSPOverrideResponder on
    </VirtualHost>
</IfModule>
```

À primeira vista podemos observar que temos duas secções neste ficheiro de configurações, uma para a porta 80 e outra para a porta 443. A primeira refere-se a ligações HTTP, estando apenas indicado que as mesmas serão redirecionadas para ligações HTTPS (não é pedido pelo enunciado, mas achamos ser a maneira mais correta deste *website* funcionar, pois obriga a que haja uma ligação segura ao mesmo).

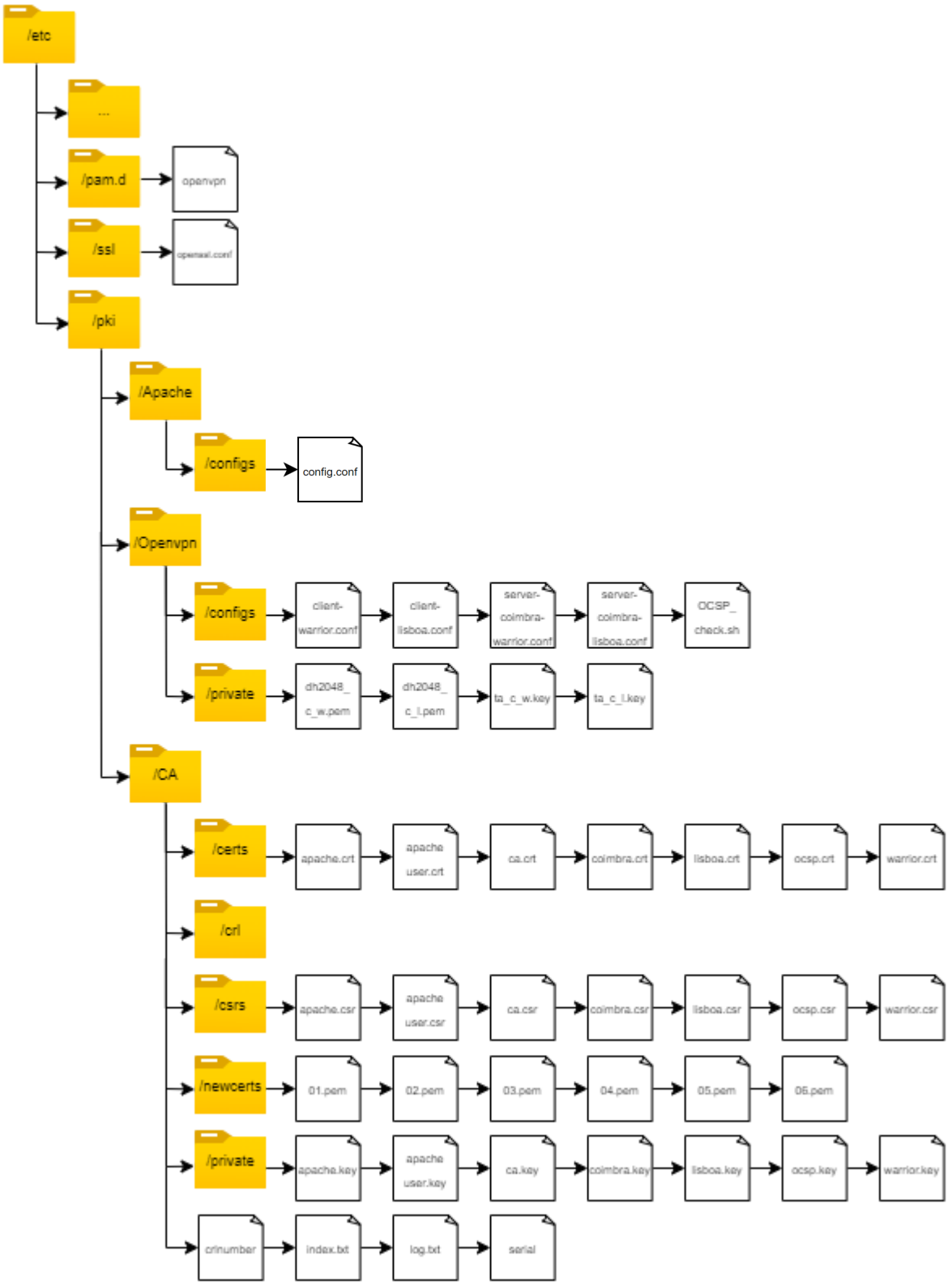
Na configuração HTTPS foram indicados certificados anteriormente criados, como é o caso do certificado próprio para Apache (e sua respetiva chave) e do certificado da CA inicialmente criado. As linhas “SSLVerifyClient require” e “SSLVerify Depth 10” servirão para autenticar qualquer pessoa que visite o servidor *Apache*, obrigando, assim, a posse de um certificado o qual, posteriormente, terá de ser importado no browser. As últimas 3 linhas desta secção referem-se à interligação do servidor *Apache* com o servidor OCSP presente em Coimbra, de modo a verificar a autenticidade dos certificados usados por quem acede ao *website*.

É de relembrar que este ficheiro de configurações ainda não se encontra na máquina virtual de Lisboa, mas sim na de Coimbra. Para que seja possível, finalmente, activar-se as configurações com HTTPS terá de ser executado os seguintes comandos (na máquina virtual de Lisboa):

```
/home/stipl1:~$ sudo a2enmod ssl
/home/stipl1:~$ sudo a2ensite config.conf
```

Distribuição final dos ficheiros criados

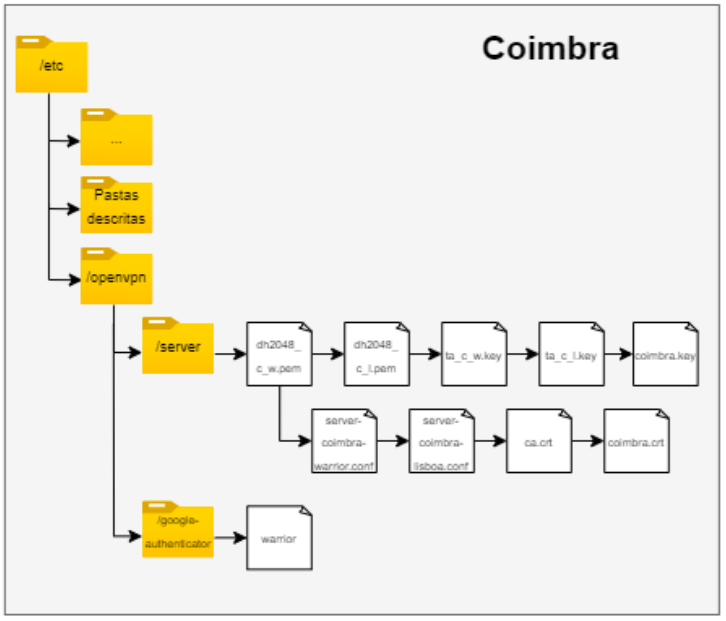
Nesta altura todos os ficheiros necessários à realização deste projeto já estarão prontos, mas, como já referido várias vezes e por questões de simplicidade, foi tudo criado na máquina virtual de Coimbra. Como tal, é necessário distribuir os ficheiros por todas as máquinas correspondentes. Nesta altura, a máquina virtual de Coimbra terá os ficheiros organizados da seguinte forma:

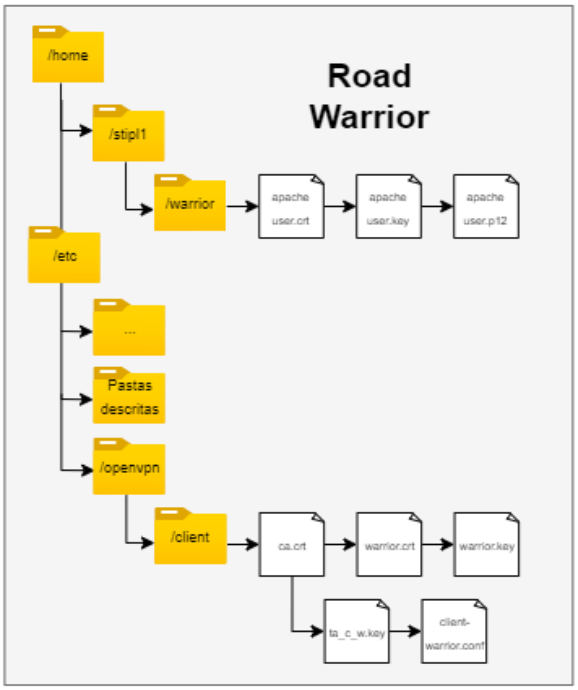


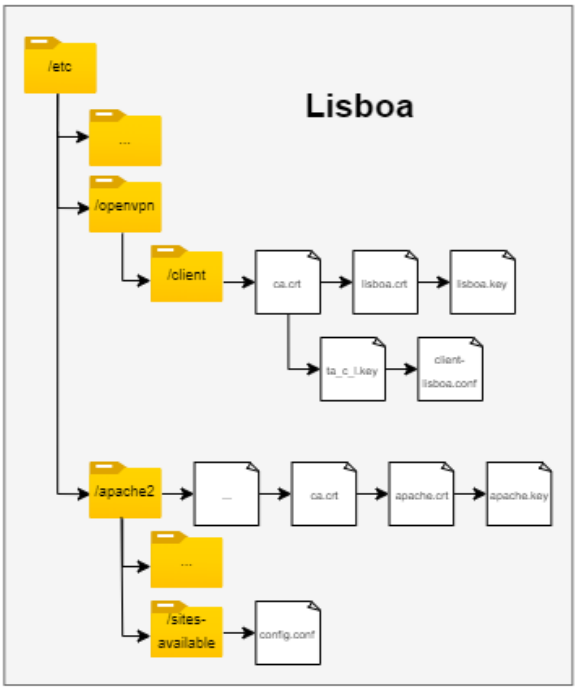
A maioria do ficheiros/pastas representadas já são conhecidas, mas seguem-se algumas notas relativamente àqueles que não foram tão mencionados:

- /etc/pki/CA/logs.txt (ficheiro): guarda a informação retornada de quando alguém verifica o estado de algum certificado usando o servidor OCSP
- /etc/pki/CA/newcerts (pasta): os ficheiros contidos nesta pasta surgiram quando foram criados certificados pela CA, deverão existir 7 ficheiros dentro desta mesma pasta.
- /etc/pki/CA/serial (ficheiro): deverá neste momento apresentar "07" como conteúdo
- /etc/pki/CA/index.txt (ficheiro): deverá neste momento conter 6 linhas, cada uma com a informação dos certificados (estado e *subject*)

Segue-se um esquema ilustrativo de como estes ficheiros deverão estar organizados em cada máquina virtual:







Dos ficheiros exibidos no esquema, o único que ainda não foi mencionado foi o "/home/stipl1/warrior/apacheuser.p12" presente na máquina virtual do *Road Warrior*. Este ficheiro será utilizado para a autenticação no servidor *Apache* ao aceder ao mesmo no *browser*. Para gerar este ficheiro *pkcs12* utiliza-se o seguinte comando, sendo necessária não só a *passphrase* anteriormente definida para o mesmo, como também uma *password* para dar *import* no *browser*:

```
/home/stipl1:~$ openssl pkcs12 -export -in apacheuser.crt -inkey apacheuser.key -out apacheuser.p12
Enter pass phrase for apacheuser.key:
Enter Export Password:
Verifying - Enter Export Password:
```

Configurações extra

De modo a ser mais percetível, foram alteradas as configurações dos ficheiros *hosts* de todas as máquinas virtuais, atribuindo, assim, *hostnames* específicos a alguns IPs. São apresentados de seguida os ficheiros "/etc/hosts" de cada máquina virtual:

- Coimbra:

```
192.168.172.20 ocsp_stipl1.pt
10.10.0.6 apache_stipl1.pt
```

- Road Warrior:

```
192.168.172.20 ocsp_stipl1.pt
10.10.0.6 apache_stipl1.pt
```

- Lisboa:

```
192.168.172.20 ocsp_stipl1.pt
10.10.0.6 apache_stipl1.pt
```

Na máquina de Coimbra foi ainda alterada uma linha no ficheiro "/etc/sysctl.conf", de modo a permitir o encaminhamento de pacotes entre *Warrior* e Lisboa, ativando, assim, o IP *forwarding*.

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

Tendo todos os ficheiros organizados nas suas respectivas diretorias e máquinas virtuais e todas as configurações efetuadas poderá executar-se, finalmente, todo o projeto. No capítulo seguinte será demonstrado como estabelecer as ligações entre os túneis, ligar o servidor OCSP e ligar o servidor *Apache*.

Execução e funcionamento

Após terem sido efetuadas todas as configurações necessárias será possível criar o cenário inicialmente descrito. Como passo inicial será preciso iniciar o servidor OCSP, sem este ligado não seria possível iniciar os restantes serviços (túneis VPN e servidor *Apache*), já que os mesmos necessitam das verificações efetuadas por este servidor.

Para iniciar o servidor OCSP deverá utilizar-se o comando já referido no capítulo em que se procedeu à sua criação.

De seguida, serão estabelecidas as ligações entre as diferentes máquinas virtuais, utilizando os túneis VPN:

- Coimbra:

```
/home/stipl1:~$ cd /etc/openvpn/server
/etc/openvpn/server:~$ sudo openvpn --config server-coimbra-warrior.conf
                                    . . .
🔐 Enter Private Key Password: ******
```

```
/home/stipl1:~$ cd /etc/openvpn/server
/etc/openvpn/server:~$ sudo openvpn --config server-coimbra-lisboa.conf
                                    . . .
🔐 Enter Private Key Password: ******
```

- Warrior:

```
/home/stipl1:~$ cd /etc/openvpn/client
/etc/openvpn/client:~$ sudo openvpn --config client-warrior.conf
                                    . . .
Enter Auth Username: warrior
🔐 Enter Auth Password: ************
🔐 Enter Private Key Password: ******
```

- Lisboa:

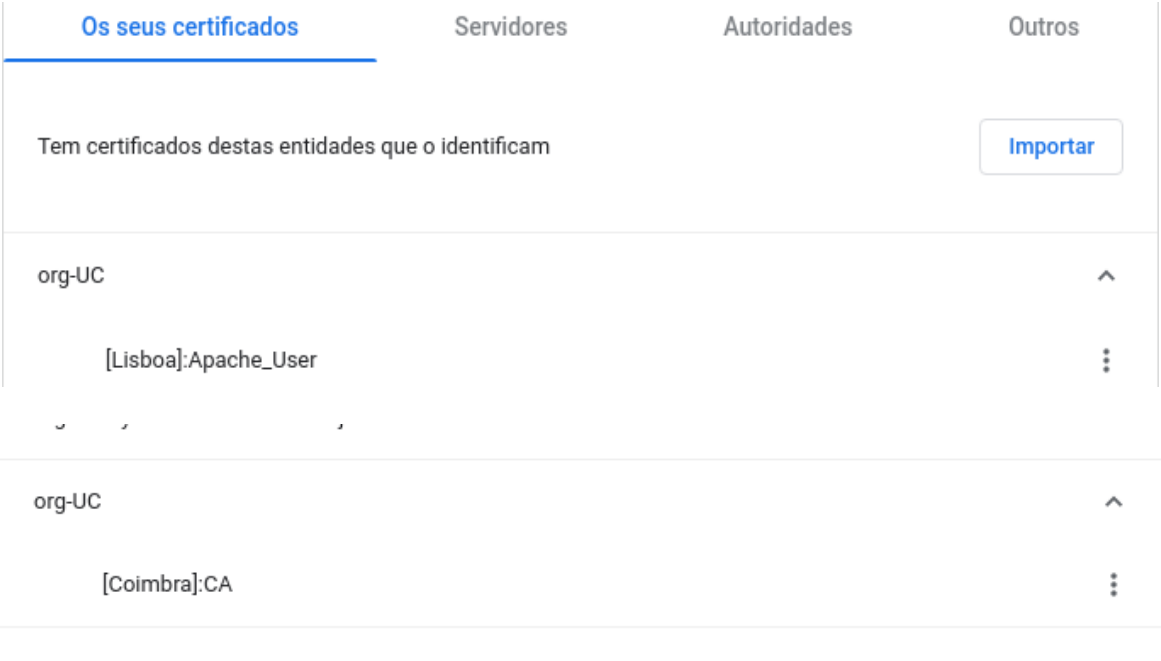
```
/home/stipl1:~$ cd /etc/openvpn/client
/etc/openvpn/client:~$ sudo openvpn --config client-lisboa.conf
                                    . . .
🔐 Enter Private Key Password: ******
```

É de notar que enquanto em Coimbra e Lisboa apenas foi pedida a *passphrase* do certificado, no *Road Warrior* foi adicionalmente pedido o *username* e *password* (*password* definida para o *user* + código do *google-authenticator*).

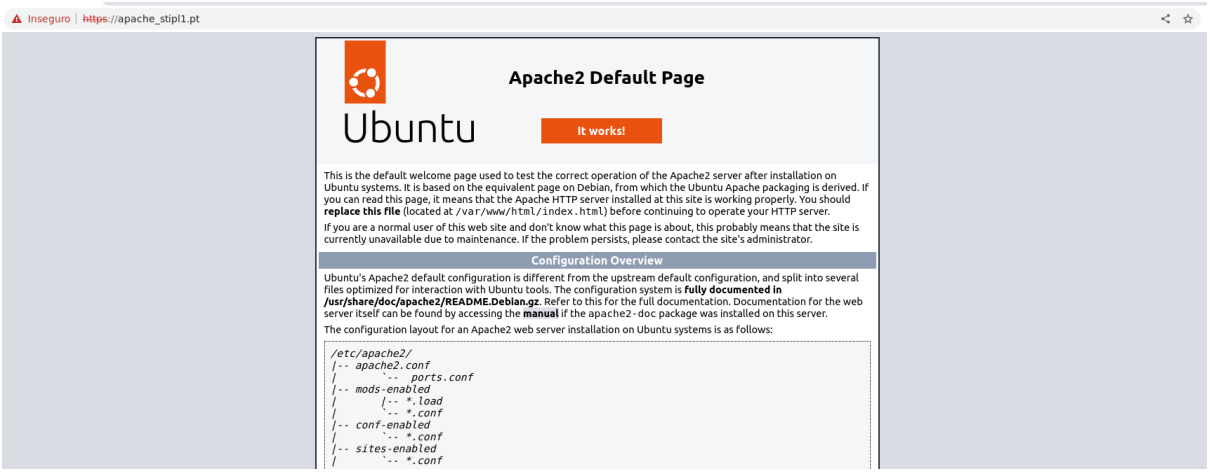
Por fim, é efetuado o *reload* e *restart* do servidor *apache* para que as configurações novas entrem em vigor, sendo pedido, de seguida, que seja introduzida a *passphrase* relativa ao certificado "apache.crt":

```
/etc/pki/CA:~$ sudo openssl ca -keyfile private/ca.key -cert certs/ca.crt -revoke certs/apacherevo.crt
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for private/ca.key:
Revoking Certificate 07.
Data Base Updated
```

Para que o *Warrior* consiga aceder ao servidor *Apache* poderá efetuá-lo através do seguinte *link*: "https://apache_stipl1.pt/". Ao entrar no mesmo irá deparar-se com uma mensagem de "erro". Com vista a resolver este "erro" será necessário fazer *import* do certificado da CA e do certificado "apacheuser.crt" (após este ter sido convertido para o formato *pkcs12*). Neste caso, o *browser* utilizado foi o *Google Chrome* e, portanto, o certificado do *Warrior* será importado na aba "Os seus certificados", enquanto que o certificado da CA será importado na aba "Autoridades".



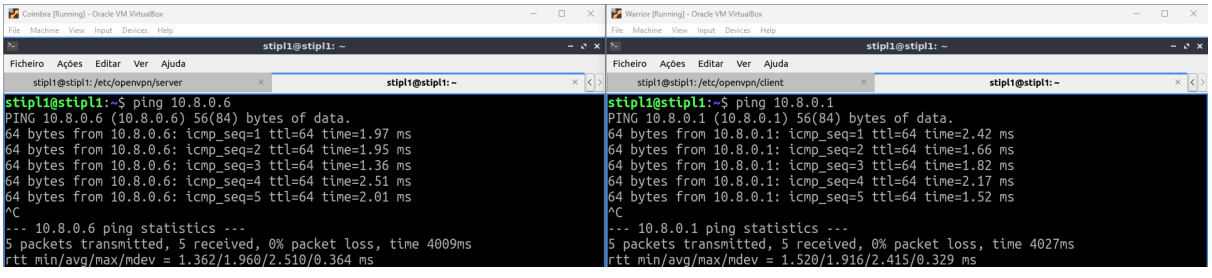
Temos, assim, finalmente o cenário completamente implementado como previsto, sendo agora possível ao *Warrior* aceder ao servidor *Apache*.

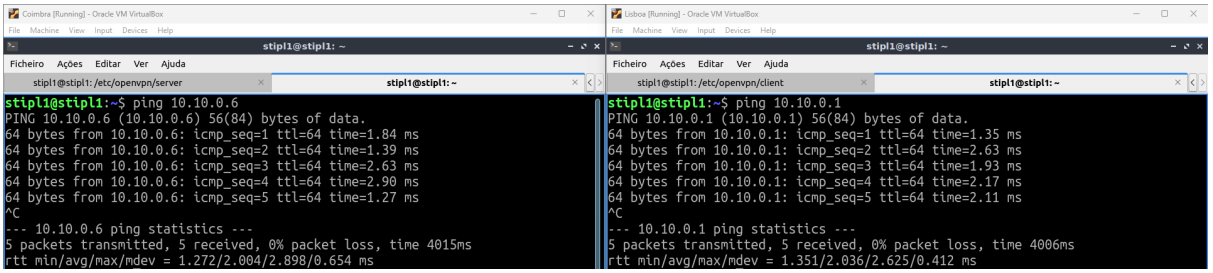


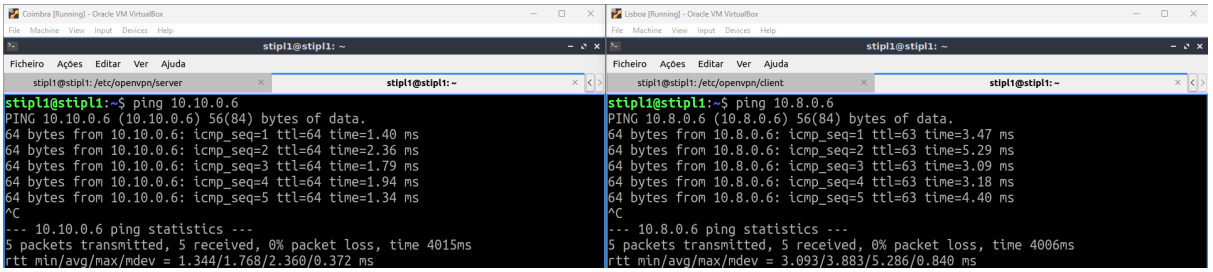
Testes

Após a implementação do cenário inicialmente proposto, foram efetuados alguns testes de modo a perceber se toda a comunicação entre as diferentes entidades e serviços estava a ser efetuada como esperado.

- ping: com vista a verificar se realmente a comunicação entre os diferentes túneis VPN está a ser efetuada, começando-se por fazer *ping* em cada máquina para as diferentes interfaces.

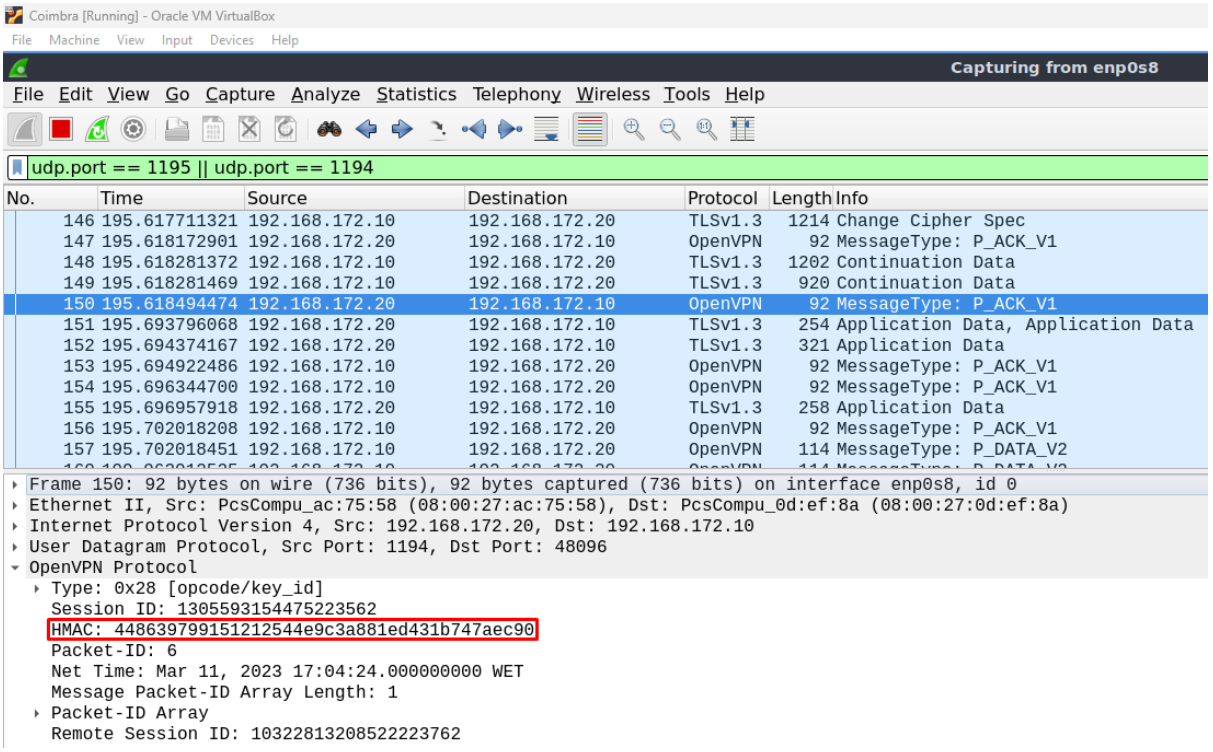




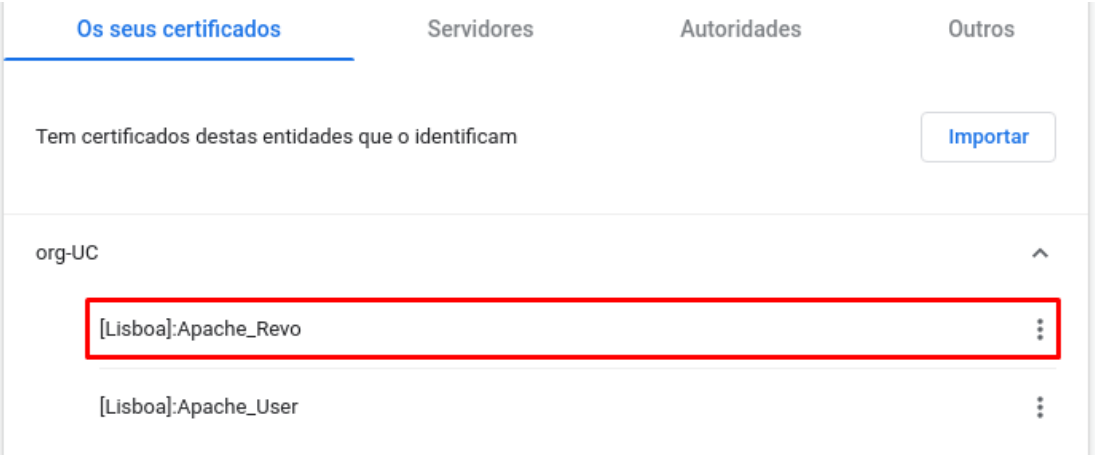


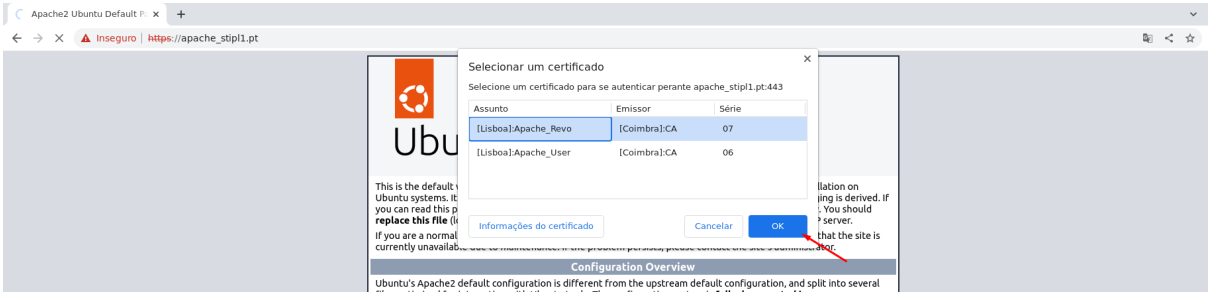
Como podemos observar, todos os *pings* são devolvidos, sendo portanto percetível que a comunicação entre túneis está funcional.

- wireshark: esta ferramenta pode ser utilizada igualmente para verificar que toda a comunicação entre túneis VPN está a funcionar, mas, neste caso, foi mais utilizada para testar se os mecanismos de segurança realmente estavam implementados (HMAC).



- revogar certificados: de modo a verificar que a revogação de certificados funciona como esperado foi criado um certificado de teste (“apacherevo.crt”). Se for utilizado este certificado na autenticação no servidor *Apache* e, mais tarde, o mesmo for revogado, o *user* deverá perder o acesso ao *website*.

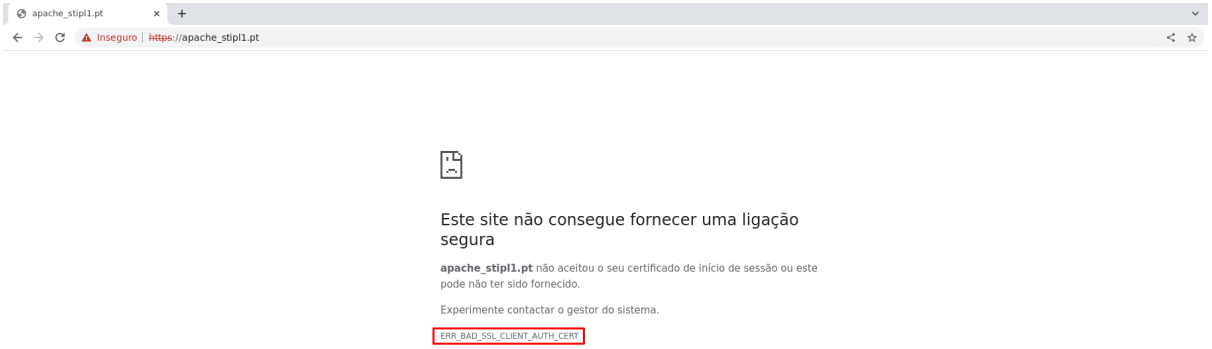




Em Coimbra é possível revogar o certificado através do seguinte comando:

```
sudo openssl ca -keyfile private/ca.key -cert certs/ca.crt -revoke certs/apacherevo.crt
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for private/ca.key:
Revoking Certificate 07.
Data Base Updated
```

Neste momento, o *Road Warrior* deverá estar impossibilitado de aceder ao servidor *Apache*:



Utilizando o servidor OCSP, pode-se confirmar que o mesmo se encontra revogado:

```
/etc/pki/CA:~$ sudo openssl ocsp -CAfile certs/ca.crt -issuer certs/ca.crt -cert certs/apacherevo.crt
-url http://ocsp_stipl1.pt:8081 -resp_text -noverify
                                        . . .
certs/apacherevo.crt: revoked
        This Update: Mar 11 17:23:59 2023 GMT
        Revocation Time: Mar 11 17:18:56 2023 GMT
```

Automatização de alguns processos

Sendo este um trabalho ainda extenso e que aborda vários pontos acerca da gestão de certificados e sua autenticação foi criado um *script* (ficheiro *bash*). Este *bash* permitirá automatizar vários processos como: criação de ficheiros (chaves, CSRs e certificados), distribuição de ficheiro pela máquinas virtuais e organização de todos ficheiros em todas as máquinas.

Para executar este *script* utiliza-se o seguinte comando:

```
/home/stipl1/:~$ bash deploy.sh <flag a utilizar>
```

As *flags* que podem ser utilizadas são as seguintes:

- -h / --help: mostrar todas as flags possíveis
- -ca / --certauth: gerar todas as chaves (ta.key, dh2048.pem, chaves de certificados), CSRs e certificados que vão ser usados
- -dist / --distribute: distribuir os ficheiros por todas as máquinas virtuais
- -c / --coimbra: organizar os ficheiros dentro da máquina virtual de Coimbra
- -l / --lisboa: organizar os ficheiros dentro da máquina virtual de Lisboa
- -w / --warrior: organizar os ficheiros dentro da máquina virtual de Warrior

É de frisar que este *script* não cria ficheiros de configuração nem faz quaisquer alterações aos mesmos, pelo que estes deverão ser criados do "zero".

De modo a não ter que executar sempre os túneis VPN, podem ser criados serviços para os mesmos (só se deverá criar estes serviços caso se tenha a certeza absoluta que as configurações do OpenVPN estão corretamente escritas e se o servidor OCSP esteja ligado).

- Coimbra:

```
/home/stipl1/:~$ sudo systemctl enable openvpn-server@server-coimbra-warrior
/home/stipl1/:~$ sudo systemctl start openvpn-server@server-coimbra-warrior
/home/stipl1/:~$ sudo systemctl enable openvpn-server@server-coimbra-lisboa
/home/stipl1/:~$ sudo systemctl start openvpn-server@server-coimbra-lisboa
```

- Warrior:

```
/home/stipl1/:~$ sudo systemctl enable openvpn-client@client-warrior
/home/stipl1/:~$ sudo systemctl start openvpn-client@client-warrior
🔐 Enter Auth Username: warrior
🔐 Enter Auth Password: ************
```

- Lisboa:

```
/home/stipl1/:~$ sudo systemctl enable openvpn-client@client-lisboa
/home/stipl1/:~$ sudo systemctl start openvpn-client@client-lisboa
```

Caso as *passphrases* dos certificados não sejam inicialmente pedidas será preciso usar o seguinte comando:

```
/home/stipl1/:~$ sudo systemd-tty-ask-password-agent
🔐 Enter Private Key Password: ******
```

Conclusão

Através deste trabalho tornou-se possível compreender melhor a importância de todos os mecanismos utilizados durante o mesmo, como certificados X.509, configuração de um servidor OCSP, utilização de túneis *VPN* e utilização *HTTPS* no servidor *Apache*. Todos estes permitiram no final de contas fornecer uma maior confiança entre as diferentes entidades do cenário proposto, onde a autenticidade dos certificados criados foi a base para a mesma. O presente trabalho permitiu ainda aprofundar os conhecimentos em diversas ferramentas como OpenSSL, OpenVPN, Apache e, no fundo, funcionamento de alguns comandos *Linux* (por exemplo: *systemctl*).

Por fim, conseguimos implementar o cenário proposto, ainda que tenham surgido vários problemas a nível das máquinas virtuais ou no *software* utilizado.

Referências

- Slides da disciplina de Segurança em Tecnologias da Informação
- https://www.openssl.org/docs/man1.1.1/man1/openssl-ca.html
- https://www.openssl.org/docs/man1.1.1/man1/ocsp.html
- https://www.openssl.org/docs/man1.1.1/man1/openssl-pkcs12.html
- https://www.openssl.org/docs/man1.1.1/man1/openssl-genrsa.html
- https://bhashineen.medium.com/create-your-own-ocsp-server-ffb212df8e63
- https://github.com/OpenVPN/openvpn/blob/master/contrib/OCSP_check/OCSP_check.sh
- https://ulimit.nl/wp-content/uploads/2019/08/Extending-a-Debian-OpenVPN-server-with-Multi-Factor-Authentication-via-Google-Authenticator.pdf
- https://linuxize.com/post/redirect-http-to-https-in-apache/
- https://access.redhat.com/documentation/en-us/red_hat_jboss_web_server/3/html/http_connectors_and_load_balancing_guide/configure_httpd_to_validate_ocsp_certificates
- https://ubuntu.com/server/docs/service-openvpn