1 2  9 0 UNIVERSIDADE D COIMBRA

Practical Assignment 1

Information Tecnology Security

Alexy de Almeida	2019192123	adenis@student.dei.uc.pt
Rodrigo Ferreira	2019220060	rferreira@student.dei.uc.pt

PL3

March 2023

Contents

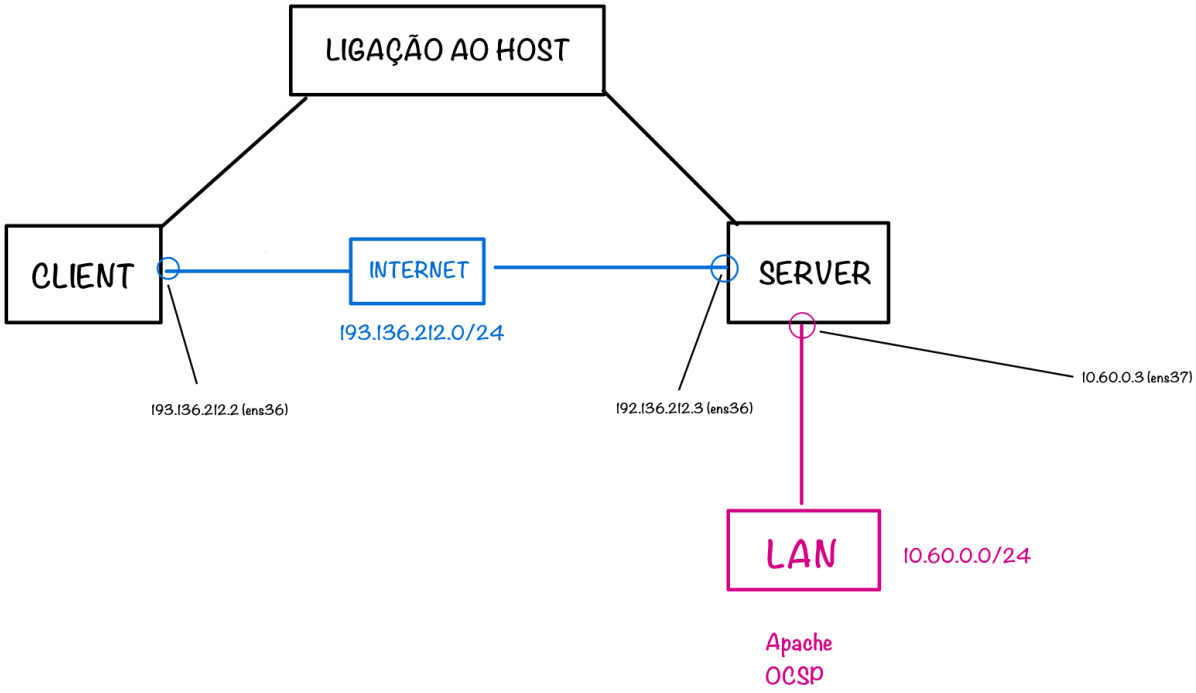
1 Introduction

The project aims to configure a VPN tunnel between a remote client and a VPN gateway using OpenVPN, as well as to enable a two-factor user authentication with OpenVPN and Apache services resorting to Google Authenticator. For that, there's also the need to create and manage certification authorities, create X.509 certificates, deal with revocation and configure OCSP for the certificates.

2 Experimental scenario

In order to be able to test our scenario, two CentOS virtual machines were used: one for the road warrior (remote client), which we will refer as **Client**; and one for the VPN gateway, which we will refer as **Server**.

2.1 Network architecture



The IP addresses were configured as follows:

- Internet: 192.136.212.0 /24
- LAN: 10.60.0.0 /24
- interface ens36 (**Server**): 193.136.212.3/24
- interface ens37 (**Server**): 10.60.0.3/24
- interface ens36 (**Client**): 193.136.212.2/24

3 Configurations

The **Client** must possess a valid X.509 certificate in order to establish a VPN tunnel with the VPN gateway and the **Server** must verify the validity of the X.509 certificate presented by the **Client** using OCSP and refuse connection if the certificate is revocated.

3.1 Packages and installations

Some packages and updates are necessary to run everything as expected.

```
# Necessary to install OpenVPN
[server & client] yum update
[server & client] yum install epel-release -y

# To install Apache and compile its modules
[server] yum install httpd mod_ssl
[server] yum install httpd-devel

# To install OpenVPN
[server & client] yum install openvpn

# To install Google Authenticator
[server] yum install google-authenticator

# Re-update packages installed
[server & client] yum update
```

The OpenVPN **Server** and **Client** configuration files are necessary in order to run OpenVPN on both of the machines. For that, we need to fetch them and put them in the right directories.

```
[server] cp /usr/share/doc/openvpn-2.4.12/sample/sample-config-files/
   server.conf /etc/openvpn
[client] cp /usr/share/doc/openvpn-2.4.12/sample/sample-config-files/
   client.conf /etc/openvpn
```

3.2 OCSP

On the **Server** side, the Apache configuration file allows us to include information about the OCSP server address and port.

```
nano /etc/pki/tls/openssl.cnf

                [ usr_cert ]
                authorityInfoAccess = OCSP;URI:http://ocsp.dei.uc.pt:81

                # uncomment
                keyUsage = digitalSignature, keyEncipherment
                extendedKeyUsage = clientAuth,serverAuth
```

We then proceed to configure Apache to support OCSP and the certificates revocation. The Apache server will use the **Client**'s certificate OCSP address and port and, in case that information isn't part of the certificate, it redirects to the address and port indicated on "SSLOCSPDefaultResponder". Apache also needs to know the location of the CRL generated, which we indicate on "SSLCARevocationFile". The "SSLCARevocationCheck" also enables to check that the revocation status not only applies to one certificate but to every one that was created.

```
nano /etc/httpd/conf.d/ssl.conf

                <VirtualHost _default_:443>
                SSLOCSPEnable on
                SSLOCSPDefaultResponder http://ocsp.dei.uc.pt:81
                SSLOCSPOverrideResponder off
                SSLCARevocationFile /etc/pki/CA/cacrl.pem
```

```
                SSLCARevocationCheck chain

                # uncomment
                SSLVerifyClient require
                SSLVerifyDepth 10
```

It is also necessary to create the **crlnumber** file that will contain the series number of the next CRL. For instance, the first certificate created by the AC will have the number 01 and so on for the next certificates to be created.

```
cd /etc/pki/CA
echo 01 > crlnumber
```

3.3 Certificates

We're now able to create the certificates that will contain the OCSP information. The first certificate to create is the CA, which will be self-signed. This certificate will be needed when creating others certificates, since we'll use it to sign them. Every one of the certificates needs to be created on the **Server** side.

```
cd /etc/pki/CA

# CA
openssl genrsa -des3 -out private/cakey.pem
openssl req -new -key private/cakey.pem -out ca.crt

                PT
                Coimbra
                Coimbra
                UC
                DEI
                STI CA
                alexx.da95@gmail.com

openssl x509 -req -days 360 -in ca.crt -out cacert.pem -signkey private/
   cakey.pem
```

An **index.txt** file is necessary to contain information about the created certificates by the AC. As for the **serial** file, it will contain the series number of the certificate. As explained previously, the first certificate will contain the number 01.

```
touch index.txt
echo 01 > serial
```

It is also important to create the **CRL** file containing the information of the revoked certificates from the AC.

```
openssl ca -gencrl -keyfile private/cakey.pem -cert cacert.pem -out cacrl.
   pem
```

The first certificate to create will be specifically for the **Server** using OpenVPN.

```
# Server with OpenVPN
openssl genrsa -des3 -out private/server.key
openssl req -new -key private/server.key -out server.csr

                PT
                Coimbra
                Coimbra
```

```
                UC
                DEI
                gw-vpn.dei.uc.pt
                alexx.da95@gmail.com

openssl ca -in server.csr -cert cacert.pem -keyfile private/cakey.pem -out
    certs/server.crt
```

The creation of the **Client** certificate is necessary both for the OpenVPN part as well as to have permission to access the Apache site, which we will address below. For that, we need to create two certificates for the **Client**and both of them will be converted to the p12 format (format needed to be able to load the certificates on the browser), but only one of them will be revoked later on.

```
# Client Apache 1
openssl genrsa -des3 -out private/client_apache.key
openssl req -new -key private/client_apache.key -out client_apache.csr

                PT
                Coimbra
                Coimbra
                UC
                DEI
                client_apache
                alexx.da95us@gmail.com

openssl ca -in client_apache.csr -cert cacert.pem -keyfile private/cakey.
   pem -out certs/client_apache.crt
openssl pkcs12 -export -out client_apache.p12 -inkey private/client_apache
   .key -in certs/client_apache.crt -certfile cacert.pem


# Client Apache 2
openssl genrsa -des3 -out private/clientp12.key
openssl req -new -key private/clientp12.key -out clientp12.csr

                PT
                Coimbra
                Coimbra
                UC
                DEI
                clientp12
                alexx.da95@gmail.com

openssl ca -in clientp12.csr -cert cacert.pem -keyfile private/cakey.pem -
   out certs/clientp12.crt

openssl pkcs12 -export -out clientp12.p12
-inkey private/clientp12.key -in certs/clientp12.crt -certfile cacert.pem
```

Finally, we need to create the Apache certificate, whose common name must be the address the **Client** will try to access later on. In order to maintain consistency, the domain dei.uc.pt will be the same as OCSP.

```
openssl genrsa -des3 -out private/apache_ocsp.key
openssl req -new -key private/apache_ocsp.key -out apache_ocsp.csr

                PT
```

```
                Coimbra
                Coimbra
                UC
                DEI
                web.dei.uc.pt
                alexx.da95@gmail.com

openssl ca -in apache_ocsp.csr -cert cacert.pem -keyfile private/cakey.pem
    -out certs/apache_ocsp.crt
```

3.4 Apache

To conclude the configurations needed for Apache, we need to add OCSP and Apache server names to the hosts file on the **Server** side.

```
nano /etc/hosts

                10.60.0.3 web.dei.uc.pt
                10.60.0.3 ocsp.dei.uc.pt
```

The same must be done for the **Client**, but we only add the Apache server name.

```
nano /etc/hosts

                10.60.0.3 web.dei.uc.pt
```

Since we created the Apache certificate, we need to edit its configuration file.

```
nano /etc/httpd/conf.d/ssl.conf

                SSLCertificateFile /etc/pki/CA/certs/apache_ocsp.crt
                SSLCertificateKeyFile /etc/pki/CA/private/apache_ocsp.key
                SSLCACertificateFile /etc/pki/CA/cacert.pem
```

3.5 OpenVPN

The next step is to configure OpenVPN both on the **Server** and **Client** sides. Before editing the OpenVPN configuration file to accept a **Client** connection in a road warrior scenario authenticated with a certificate, two files need to be generated: a file containing the parameters for the Diffie-Hellman algorithm and a secret key file. The secret key helps block DoS attacks and UDP port flooding, so both the **Server** and the **Client** muss possess the same copy.

```
cd /etc/openvpn
openssl dhparam -out dh2048.pem 2048
openvpn --genkey --secret ta.key
```

In order for OCSP to validate the authenticity of the **Client**'s certificate that's trying to establish a VPN connection to the server, a script needs to be modified:

```
cp /usr/share/doc/openvpn-2.4.12/contrib/OCSP_check/OCSP_check.sh /etc/
   openvpn
nano /etc/openvpn/OCSP_check.sh

                ocsp_url="http://ocsp.dei.uc.pt:81"
                issuer="/etc/pki/CA/cacert.pem"
                verify="/etc/pki/CA/cacert.pem"

                # add at the end
```

```
                echo "OCSP status: $status"

chmod 755 /etc/openvpn/OCSP_check.sh
```

We can then start to edit the configuration file as follows:

```
nano /etc/openvpn/server.conf

                script-security2
                local 193.136.212.3
                port 1194
                proto udp
                dev tun
                ca /etc/pki/CA/cacert.pem
                cert /etc/pki/CA/certs/server.crt
                key /etc/pki/CA/server.key
                dh dh2048.pem
                server 10.8.0.0 255.255.255.0
                push "route 10.60.0.0 255.255.255.0"
                tls-verify /etc/openvpn/OCSP_check.sh
                tls-auth ta.key 0
                comp-lzo
```

We first indicate that we need to run the OCSP script, which we verify at the bottom with "tls-verify". 193.136.212.3, port 1194 are the IPv4 address and port on which the server will be accepting new connections. We can see that the previously created file containing the values for the Diffie-Hellman algorithm is needed, as well as the secret key with value 0. 10.8.0.0 was the selected IP address range attributed to the VPN clients, knowing that the server will assign the first address, 10.8.0.1, to the "tun0" interface.

But before running OpenVPN on the **Client** side, some files need to be transferred over there in a secured way. The CA certificate, the OpenVPN **Client** key (client_apache.key), the OpenVPN **Client** certificate (client_apache.crt), the secret key (ta.key) and the dh2048 file (dh2048.pem), which all need to be included in the configuration file.

At the same time, we can also transfer the **Client**'s Apache key (clientp12.key) and Apache certificate (clientp12.crt) for testing purposes later on. Last but not least, we also need to send the Apache certificates (clientp12.p12 and client_apache.crt) in order to load the certificate on the browser and provide access for the **Client** to the Apache server. In order to test everything correctly, the certificate clientp12.p12 will be revoked later and we shall test the access to the Apache server with both the revoked and non-revoked certificates.

All of this can be safely done via SSH:

```
scp /etc/pki/CA/cacert.pem 192.168.6.136:/etc/pki/CA
scp /etc/pki/CA/private/client_apache.key 192.168.6.136:/etc/pki/CA/
   private
scp /etc/pki/CA/certs/client_apache.crt 192.168.6.136:/etc/pki/CA/certs
scp /etc/pki/CA/client_apache.p12 192.168.6.136:/etc/pki/CA
scp /etc/pki/CA/certs/clientp12.crt 192.168.6.136:/etc/pki/CA/certs
scp /etc/pki/CA/clientp12.p12 192.168.6.136:/etc/pki/CA
scp /etc/pki/CA/private/clientp12.key 192.168.6.136:/etc/pki/CA/private
scp /etc/openvpn/ta.key 192.168.6.136:/etc/openvpn
scp /etc/openvpn/dh2048.pem 192.168.6.136:/etc/openvpn
```

As for the OpenVPN **Client** configuration file:

```
nano /etc/openvpn/client.conf
```

```
                client
                dev tun
                proto udp
                remote gw-vpn.dei.uc.pt 1194
                user nobody
                group nobody
                ca /etc/pki/CA/cacert.pem
                cert /etc/pki/CA/certs/client.crt
                key /etc/pki/CA/private/client.key
                tls-auth ta.key 1
```

The common name we chose for the **Server** when creating its certificate was "server-vpn", which we need to indicate after "remote" and before the port to be able to establish the tunnel connection using VPN. Another important step is to add the **Server** name of the on the **Client**'s hosts.

```
nano /etc/hosts
                193.136.212.3 gw-vpn.dei.uc.pt
```

We can finally configure OpenVPN for the **Client**:

```
nano /etc/openvpn/client.conf

                client
                dev tun
                proto udp
                remote gw-vpn.dei.uc.pt 1194
                user nobody
                group nobody
                ca /etc/pki/CA/cacert.pem
                cert /etc/pki/CA/certs/client.crt
                key /etc/pki/CA/private/client.key
                tls-auth ta.key 1
```

3.6 Two-Factor Authentication

OpenVPN

The first step before configuring the two-factor authentication is to add the auth-pam plugin at the end of the **Server** OpenVPN configuration file.

```
nano /etc/openvpn/server.conf
                plugin /usr/lib64/openvpn/plugins/openvpn-plugin-auth-pam.
                   so openvpn
```

Next, we need to activate IP forwarding:

```
nano /etc/sysctl.conf
                net.ipv4.ip_forward = 1
                net.ipv6.conf.all.forwarding = 1
```

We also need to give permissions to Google Authenticator:

```
useradd gauth
mkdir /etc/openvpn/google-auth
chown gauth:gauth /etc/openvpn/google-auth && chmod 700 /etc/openvpn/
   google-auth
semanage fcontext -a -t openvpn_etc_rw_t -ff '/etc/openvpn/google-auth
   (/.*)?'
```

A Google Authenticator user can finally be created:

```
useradd -s /sbin/nologin alexy-authenticator
passwd alexy-authenticator
                alexy1995
```

We can now use the configurations and the newly created user to generate the QR code, which we need to scan on the Google Authenticator mobile application:

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

Finally, we need to create a file with the following content:

```
nano /etc/pam.d/openvpn

                auth [user_unknown=ignore success=ok ignore=ignore default
                   =bad] pam_securetty.so
                auth required pam_google_authenticator.so secret=/etc/
                   openvpn/google-auth/${USER} user=gauth forward_pass
                auth include system-auth
                account include system-auth
                password include system-auth
```

Apache

The first step to configure a two-factor authentication for Apach is to get the source code and decompress the file needed to install the module:

```
# Get source code and decompress file
cd
wget https://storage.googleapis.com/google-code-archive-downloads/v2/code.
   google.com/google-authenticator-apache-module/GoogleAuthApache.src.r10.
   bz2
tar xvf GoogleAuthApache.src.r10.bz2

# Compile the module
cd google-authenticator-apache-module
make install
```

The next step is to add the load of the module and add Google-Authentication configurations to the Apache configuration file:

```
nano /etc/httpd/conf/httpd.conf

# Add after "Include conf.modules.d/*.conf"
Loadmodule authn_google_module modules/mod_authn_google.so

# Alter <Directory />
<Directory />
     Options FollowSymLinks ExecCGI
     AllowOverride All
     Order deny,allow
     Allow from all
     AuthType Basic
     AuthName "Alexy and Rodrigo's authentication"
     AuthBasicProvider "google_authenticator"
     Require valid-user
     GoogleAuthUserPath google_auth_users
     GoogleAuthCookieLife 3600
```

```
    GoogleAuthEntryWindow 2
</Directory>
```

An important step in the configuration file is to comment everything that is inside the tags "Directory "/var/www"" and "Directory "/var/www/html"". After that, we can copy the Google-Authenticator user data to Apache:

```
# Create directory for Google-Authenticator users
cd /etc/httpd
mkdir google_auth_users

# Copy data and put inside the google-authenticator users folder
cp /etc/openvpn/google-auth/alexy-authenticator /etc/httpd/
   google_auth_users

# Give permission only to Apache to access the user's data
chown apache:apache /etc/httpd/google_auth_users/alexy-authenticator
```

4 Testing

4.1 Certificate revocation

Two **Client** certificates are going to be used for testing purposes: clientp12.crt and client_apache.crt. The certificate to be revoked is the first one, which means that this certificate cannot access the Apache url and, if it's chosen as the one configured in the OpenVPN **Client** configuration file, it cannot establish a VPN connection as well.

```
openssl ca -revoke certs/clientp12.crt -keyfile private/cakey.pem -cert
   cacert.pem
openssl ca -gencrl -keyfile private/cakey.pem -cert cacert.pem -out cacrl.
   pem
```

To the CRL content and be certain of the revoked certificates, we use the following command:

```
openssl crl -in cacrl.pem -noout -text
```

We obtain the following result:

```
Certificate Revocation List (CRL):
        Version 2 (0x1)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: /C=PT/ST=Coimbra/L=Coimbra/O=UC/OU=DEI/CN=STI CA/emailAddress=alexx.da95@gmail.com
        Last Update: Mar 10 11:20:51 2023 GMT
        Next Update: Apr  9 11:20:51 2023 GMT
        CRL extensions:
            X509v3 CRL Number:
                2
Revoked Certificates:
    Serial Number: 03
        Revocation Date: Mar  8 18:49:31 2023 GMT
    Signature Algorithm: sha256WithRSAEncryption
         7f:72:24:d3:83:bb:fb:8e:44:d8:33:b6:58:9d:22:27:05:8f:
         39:bc:37:7a:ee:53:fb:e1:a8:c8:d4:57:9c:3c:31:50:52:9f:
         1c:f6:4d:18:9b:cf:ec:1a:25:5e:ce:11:a3:6c:72:5d:63:d5:
         a9:d6:18:91:aa:16:89:b8:fd:34:da:8f:4d:70:8d:ad:20:9d:
         2c:ef:b3:50:4b:9d:71:3f:ef:17:7c:59:76:6b:c6:f3:5d:e8:
         33:0e:79:10:61:24:28:40:85:45:0f:ff:53:e2:77:4c:b9:5a:
         50:77:36:59:f7:9c:91:2f:39:1f:62:34:94:a7:60:2a:ec:81:
         62:87:4d:2a:a8:22:c6:53:89:82:70:fa:80:a3:5d:86:37:b2:
         0f:65:44:d0:10:32:ed:93:3e:7e:9d:3e:ea:01:cc:6a:b0:24:
         b7:e0:86:a8:a4:a3:ff:44:4a:78:77:f5:ac:f7:8a:70:5f:eb:
         42:42:5e:da:0b:1f:0b:b0:3b:fe:79:99:84:8a:8a:3c:4a:f3:
         1c:b8:da:34:11:3f:d3:2d:5a:a4:26:5a:45:45:2f:02:12:cb:
         5f:33:06:92:ec:cc:a2:d7:db:22:27:7d:df:9f:99:7b:56:02:
         04:a7:77:d1:0f:f4:14:08:3b:7f:27:ba:05:3a:03:2b:10:0c:
         71:ff:dc:34
```

To verify that the certificate was indeed revocated, we use the following command:

```
openssl ocsp -CAfile cacert.pem -issuer cacert.pem -cert certs/clientp12.
   crt -url http://ocsp.dei.uc.pt:81 -resp_text
```

We can verify the ID (03) of the revoked certificate, also mentioned in the crl file:

```
OCSP Response Data:
    OCSP Response Status: successful (0x0)
    Response Type: Basic OCSP Response
    Version: 1 (0x0)
    Responder Id: C = PT, ST = Coimbra, L = Coimbra, O = UC, OU = DEI, CN = STI CA, emailAddress = ale
xx.da95@gmail.com
    Produced At: Mar 10 19:32:20 2023 GMT
    Responses:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: EF9827B0020B4F6A297C2DEC5C4E94895362F6F0
      Issuer Key Hash: 39A8007B564701FBF40123FBE36EACE52900461B
      Serial Number: 03
    Cert Status: revoked
    Revocation Time: Mar  8 18:49:31 2023 GMT
    This Update: Mar 10 19:32:20 2023 GMT
```

4.2 Connectivity

Client	Server OpenVPN	Client OpenVPN	Certificate	Results
193.136.212.3 (Server VPN)	indifferent	indifferent	indifferent	gets response
10.8.0.1 (Server VPN) 10.60.0.1 (LAN)	enabled	disabled	indifferent	no response
10.8.0.1 (Server VPN) 10.60.0.1 (LAN)	disabled	enabled	indifferent	no response
10.8.0.1 (Server VPN) 10.60.0.1 (LAN)	enabled	enabled	revoked	no response
10.8.0.1 (Server VPN) 10.60.0.1 (LAN)	enabled	enabled	good	gets response

4.3 OpenVPN

Client	Server OpenVPN	Client OpenVPN	Certificate	Results
Server OpenVPN	disabled	enabled	indifferent	connection refused
Server OpenVPN	enabled	enabled	good	access granted
Server OpenVPN	enabled	enabled	revoked	access denied

Server	Server OpenVPN	Client OpenVPN	Certificate	Results
Server OpenVPN	disabled	enabled	indifferent	connection refused
Server OpenVPN	enabled	enabled	good	access granted
Server OpenVPN	enabled	enabled	revoked	access denied

4.4 Apache

Client	Server OpenVPN	Client OpenVPN	Certificate	Results
web.dei.uc.pt	enabled	disabled	good	cannot find address
web.dei.uc.pt	enabled	enabled	good	can access
web.dei.uc.pt	enabled	enabled	revoked	access is revoked

4.5 Google Authenticator

Client	Server OpenVPN	Client OpenVPN	Certificate	Credentials	Results
OpenVPN	enabled	enabled	good	invalid	access denied
OpenVPN	enabled	enabled	good	valid	access granted
Apache	enabled	enabled	good	invalid	access denied
Apache	enabled	enabled	good	valid	access granted

5 Conclusion

From the tests that were performed and the results we obtained, we can safely conclude that the goals for this project were reached. We successfully implemented a VPN tunnel between a client and a server, while at the same time using OCSP to validate the certificate of the client establishing the VPN connection and accessing the Apache server. We also implemented a Two-Factor Authentication using Google Authentication, both for OpenVPN as well as Apache, which gave several layers of security to the system. That is, the client would need a valid certificate, valid Google Authenticator credentials and permited access to the server network in order to be able to enter the Apache server.

The hardest part of the project was understanding the right commands to use and how to solve minor problems related to that, which could be a question of minutes or hours. Some steps might also be missing in certain documents and without those we wouldn't be able to safely configure everything.

Bibliography

[1] Urs, "Setup an openvpn server with certificate and two-factor authentication on centos 7." [Online]. Available: https://nethack.ch/2016/12/08/setup-an-openvpn-server-with-certificate-and-two-factor-authentication-on-centos-7/

[2] B. Simon, "Getting google authenticator and apache to cooperate on centos." [Online]. Available: https://www.blogbyben.com/2012/02/getting-google-authenticator-and-apache.html

[3] C. Duckett, "Pairing apache and google authenticator." [Online]. Available: https://www.techrepublic.com/article/pairing-apache-and-google-authenticator/

[4] G. Ficara, "Openvpn with google 2-factor authentication on centos 7." [Online]. Available: https://velenux.wordpress.com/2019/03/12/openvpn-with-google-2-factor-authentication-on-centos-7/

[5] J. Granjal, *Segurança Prática em Sistemas e Redes com Linux*. FCA, 2017.