

Systems Security

OpenVPN

Miguel Frade

Polytechnic Institute of Leiria

School year 2017–2018



1 Introduction

- What is OpenVPN
- Install required software

2 Configure

- Client to site

3 Testing

- Linux
- Other OS

Introduction

What is OpenVPN?

- is an open-source software application that implements Virtual Private Network (VPN);
- creates secure **client-to-site** or **site-to-site** connections in routed or bridged configurations and remote access facilities;
- it uses a custom security protocol that utilizes SSL/TLS for key exchange;
- it is capable of traversing network address translators (NATs) and firewalls (if properly configured);
- it is available for all major Operating Systems: Windows, MacOS, Linux, Android, iOS;

Introduction

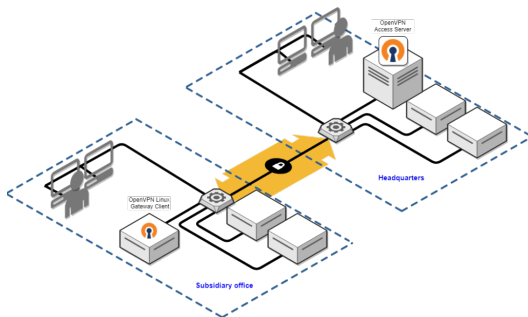
What is OpenVPN?

- is an open-source software application that implements Virtual Private Network (VPN);
- creates secure **client-to-site** or **site-to-site** connections in routed or bridged configurations and remote access facilities;
- it uses a custom security protocol that utilizes SSL/TLS for key exchange;
- it is capable of traversing network address translators (NATs) and firewalls (if properly configured);
- it is available for all major Operating Systems: Windows, MacOS, Linux, Android, iOS;

Main disadvantage:

- lack of good GUI, most of the configuration must be done in the command line and configuration files;
 - but there are compatible alternatives (out of scope for this class): [SoftEther](#)

Authentication Methods



OpenVPN has several ways to authenticate peers with each other:

- **pre-shared keys** – the easiest setup, but also the less secure;
- **username / password** – can be enabled, both with or without certificates;
- **digital certificates** – the most robust solution and feature-rich;

Setup test scenario

Recommended setup to test OpenVPN with the instructions available on the next slides:

- ❶ **server** – setup a virtual machine with Ubuntu 16.04: server
 - with 2 network interfaces, one “NAT” and the other as “Host Only”;
- ❷ **client** – setup a virtual machine with (k)ubuntu 16.04 desktop;
 - with 2 network interfaces, one “NAT” and the other as “Host Only”;
 - later the “NAT” interface must be shutdown:
 - with only the “Host Only” interface this setup will prevent the virtual to have access to the Internet, but should be able to access it through the server after the VPN is properly setup;
 - shutdown “NAT” interface: `sudo ifdown enp3s0` *# replace enp3s0 with the name of your NAT interface*
 - activate “NAT” interface: `sudo ifup enp3s0`
- ❸ configure OpenVPN on both server and client virtual machines accordingly to these slides;

Server and Client – Install required software

Software requirements for Linux:

- **server and client: OpenSSL** – already installed in most Linux distributions;
- **server and client: OpenVPN** – install the ▶ latest stable version on both client and server. The instructions are for Ubuntu 16.04

```
sudo -i      # enter root mode

# download and install GnuPG repository key
wget -O - https://swupdate.openvpn.net/repos/repo-public.gpg | apt-key add -

# add repository to the source lists
echo "deb http://build.openvpn.net/debian/openvpn/release/2.4 xenial main" > /etc/apt/sources.list.d/openvpn-aptrepo.list

# install OpenVPN
apt update && apt install openvpn

exit        # exit root mode
```

- **note:** on the **client** activate the client interface “NAT” temporarily to install the required software;

Generate digital certificates and secret key

Easy-RSA

- **server: Easy-RSA** – script to help setup the digital certificates, available [here](#). No installation required just uncompress the downloaded file: `unzip EasyRSA-3.0.3.zip`

- create a copy of the sample configuration file

```
cd EasyRSA-3.0.3      # change directory
cp vars.example vars  # create a copy of the sample configuration file
```

- edit configuration file

```
nano vars              # you may use your favorite editor (i.e. kate, gedit, vim)
```

```
# Change the following parameters in the configuration file:
```

```
set_var EASYRSA_REQ_COUNTRY    "PT"
set_var EASYRSA_REQ_PROVINCE   "Leiria"
set_var EASYRSA_REQ_CITY       "Leiria"
set_var EASYRSA_REQ_ORG        "IPLeiria"
set_var EASYRSA_REQ_EMAIL      "xxxx@ipleiria.pt"  # choose your email address
set_var EASYRSA_REQ_OU         "Systems Security"
```

```
set_var EASYRSA_ALGO           ec                  # change to Elliptic Curve algorithm. Warning: not supported by all clients
set_var EASYRSA_CURVE          secp521r1          # setup the curve name, to list all supported curves: openssl ecparam -list_curves
set_var EASYRSA_DIGEST         "sha512"           # setup hash algorithm
```


Generate digital certificates and secret key

Easy-RSA

- generate digital certificates

```
./easysrsa init-pki                # create PKI directories
./easysrsa build-ca               # create CA key pair and self-signed certificate

# repeat for each server:
./easysrsa gen-req SERVER_x nopass # create a key pair for server SERVER_x
./easysrsa sign-req server SERVER_x # sign public key of SERVER_x and create the digital certificate
./easysrsa gen-dh                 # generate DH parameters for key exchange

# repeat for each client:
./easysrsa gen-req CLIENT_x nopass # create a key pair for client CLIENT_x
./easysrsa sign-req client CLIENT_x # sign public key of CLIENT_x and create the digital certificate
```

OpenVPN

- generate secret key (optional, but recommended)

```
openvpn --genkey --secret ta.key # create a static key
```

Filenames and their purpose

Important files generated by Easy-RSA (running in the CA computer):

Filename	Needed by	Description	Secret
pki/private/ca.key	only CA computer	root CA private key	YES
pki/ca.crt	servers and clients	root CA certificate	no
pki/ta.key	servers and clients	HMAC authentication key	YES
pki/dh_x.pem	only server x	Diffie Hellman parameters	no
pki/issued/SERVER_x.crt	only server x	server x certificate	no
pki/private/SERVER_x.key	only server x	server x private key	YES
pki/issued/CLIENT_x.crt	only client x	client x certificate	no
pki/private/CLIENT_x.key	only client x	client x private key	YES

Use `cp`, `scp` or `WinSCP` to copy files to servers and clients

Where to place the files

Create destination folders inside each server and client:

Server x	Client x
<code>sudo mkdir -p /etc/openvpn/keys</code>	<code>mkdir -p ~/.openvpn/keys</code>
<code>sudo chmod 700 /etc/openvpn/keys</code>	<code>chmod 700 ~/.openvpn/keys</code>

Copy the files created by Easy-RSA to the following folders:

Server x	Client x
<code>/etc/openvpn/ca.crt</code>	<code>~/.openvpn/ca.crt</code>
<code>/etc/openvpn/keys/ta.key</code>	<code>~/.openvpn/keys/ta.key</code>
<code>/etc/openvpn/SERVER_x.crt</code>	<code>~/.openvpn/CLIENT_x.crt</code>
<code>/etc/openvpn/keys/SERVER_x.key</code>	<code>~/.openvpn/keys/CLIENT_x.key</code>
<code>/etc/openvpn/dh_x.pem</code>	

Create the SERVER configuration file

For each Linux server:

```
sudo -i                # enter root mode
cd /etc/openvpn
# copy sample configuration file and uncompress it
cp /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz .
gunzip -c server.conf.gz > server_x.conf
nano server_x.conf     # edit server.conf and check the following parameters (add them if needed):
# ...
ca ca.crt
cert SERVER_x.crt
key keys/SERVER_x.key
dh dh_x.pem
# ...
server 10.8.0.0 255.255.255.0      # DHCP pool for clients, cannot conflict with existing subnets
# ...
push "redirect-gateway def1 bypass-dhcp"  # to avoid DNS leaking
# ...
push "dhcp-option DNS 9.9.9.9"         # instruct clients to use QUAD9 DNS service (or other service you wish)
# ...
tls-auth keys/ta.key 0              # HMAC authentication key, 0 for servers
auth SHA512                         # HMAC algorithm
# ...
cipher AES-256-CBC                  # encryption algorithm for the communications
# ...
user nobody                         # reduce OpenVPN privileges
group nogroup
# ..
exit                                # exit root mode
```

Create the CLIENT configuration file

For each Linux client:

```
cd ~/.openvpn
# copy sample configuration file
cp /usr/share/doc/openvpn/examples/sample-config-files/client.conf ./client_x.conf
nano client_x.conf # edit client_x.conf and check the following parameters (add them if needed):
# ...
client                                     # VPN client mode
# ...
remote SERVER_x_ip_address 1194           # replace with the IP address of the server
# ...
ca ca.crt
cert CLIENT_x.crt
key keys/CLIENT_x.key
# ...
tls-auth keys/ta.key 1                    # HMAC authentication key, 1 for clients
auth SHA512                               # HMAC algorithm
# ...
cipher AES-256-CBC                        # encryption algorithm for the communications
# ...
user nobody                               # reduce OpenVPN privileges
group nogroup
# ..
```

Server – Enable packet forwarding on Linux

For each server:

- read the current state of IP forwarding

```
$ sysctl net.ipv4.ip_forward  
net.ipv4.ip_forward = 0
```

- enable forwarding

- temporarily (lost after a reboot)

```
$ sudo sysctl -w net.ipv4.ip_forward=1  
net.ipv4.ip_forward = 1
```

- persistently (remains after a reboot)

```
$ sudo nano /etc/sysctl.conf  
# change value to "1":  
net.ipv4.ip_forward = 1  
  
# if IPv6 is required change also:  
net.ipv6.conf.all.forwarding = 1
```

Server – setup IPTables to allow OpenVPN traffic

Add the following rules to your IPTables firewall:

```
IPT=/sbin/iptables
$IPT -F && $IPT -t nat -F    # delete any previous configurations
$IPT -P INPUT ACCEPT       # change to DROP after
$IPT -P OUTPUT ACCEPT      # successful test of your
$IPT -P FORWARD ACCEPT     # OpenVPN configuration
# (...)
DYN=1024:65535
echo "OpenVPN: server on port 1194 UDP"
$IPT -A INPUT -p udp -m state --state NEW --sport $DYN --dport 1194 -j ACCEPT

echo "OpenVPN: allow interface TUN"
$IPT -A INPUT -i tun+ -j ACCEPT

IFNAME=eth0                 # change to the name of your interface
echo "OpenVPN: route TUN <-> $IFNAME"
$IPT -A FORWARD -i tun+ -j ACCEPT
$IPT -A FORWARD -i tun+ -o $IFNAME -m state --state RELATED,ESTABLISHED -j ACCEPT
$IPT -A FORWARD -i $IFNAME -o tun+ -m state --state RELATED,ESTABLISHED -j ACCEPT

# This part is mandatory even if you don't want to use a firewall
echo "OpenVPN: ativar NAT"
SOURCE_NET=10.8.0.0/8      # source network from DHCP pool
$IPT -t nat -A POSTROUTING -s $SOURCE_NET -o $IFNAME -j MASQUERADE
```

Apply the IPTables rules and then perform the tests on the next slide.

First test – Linux OS on both client and server

1 client:

- make sure the “NAT” interface is shutdown;
- on the “Host only” interface configure the default gateway with the IP of the server;
- disable any firewall you might have configured;

2 server – stop OpenVPN service and execute OpenVPN on the command line:

```
$ sudo service openvpn stop      # for debugging purposes
$ cd /etc/openvpn
$ sudo openvpn SERVER_x.conf      # the terminal will stay locked after this command, press CTRL+C to terminate
# ...
Thu Nov 23 14:56:14 2017 Initialization Sequence Completed
```

3 client – execute OpenVPN on the command line:

```
$ cd ~/.openvpn
$ sudo openvpn CLIENT_x.conf      # the terminal will stay locked after this command, press CTRL+C to terminate
# ...
Thu Nov 23 15:03:24 2017 Initialization Sequence Completed
```

4 client – check if VPN is working:

```
$ traceroute 9.9.9.9
1?: [LOCALHOST] pmtu 1500
1: 10.8.0.1 125.908ms # IP from OpenVPN DHCP pool
1: 10.8.0.1 139.983ms # this IP doesn't show up if the VPN is not connected
# ...
6: dns.quad9.net 138.106ms !H
```

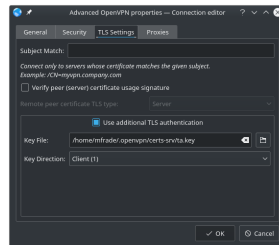
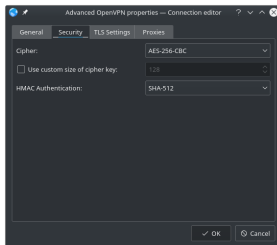
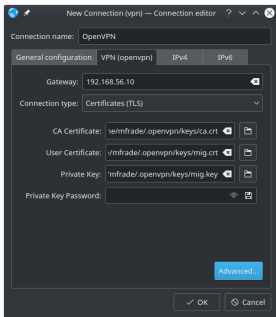

Second test – Linux OS on both client and server

After the previous test is successful:

- **client** – install OpenVPN integration with the desktop interface

```
$ sudo apt install network-manager-openvpn
```

- configure and test again;



- **server** – after successful test with the GUI, enable OpenVPN service:

```
$ sudo service openvpn start
```

OpenVPN for other OS

OpenVPN is available for many OS:

- **Windows** – check OpenVPN [downloads page](#)
- **MAC** – a popular OpenVPN client for MacOSX is [Tunnelblick](#)
- **Android** – there are many clients:
 - [OpenVPN Connect](#) – from the OpenVPN authors, last time I tried didn't work well with Elliptic Curve algorithms and contains ads;
 - [OpenVPN for Android](#) – from Arne Schwabe, works well with Elliptic Curve algorithms and doesn't have ads;
- **iOS** – [OpenVPN Connect](#) from the OpenVPN authors;
- **pfSense** – this firewall OS has built-in support for OpenVPN, check this [tutorial](#)

All these OpenVPN clients require a single .ovpn configuration file, with all keys embedded

How to build a .ovpn file

- the .ovpn file is very similar to the .conf file
- but the .ovpn file has all keys and certs embedded, including the private key
- the conversion from .conf to .ovpn can be automated
 - download the script `conf2ovpn.sh` from Moodle
 - or copy/past it from the slides:

```
cd ~/.openvpn          # go to the directory with the .conf file

touch conf2ovpn.sh     # create an empty script file

# give execution permissions
chmod +x conf2ovpn.sh

# edit the script file
nano conf2opvpn.sh
    # then copy/past the script on the next slide

# execute the script
./conf2ovpn.sh CLIENT_x.conf
```

- then copy the .ovpn to the expected location for your OpenVPN client software;

Script conf2ovpn.sh (available for download on Moodle)

```
1  #!/bin/bash
2  # based on https://goo.gl/DjkX39
3  src=$1
4  dst="$(basename $src .ovpn)_unified.ovpn"
5
6  gawk -f - $src > $dst << 'AWK'
7
8  BEGIN {
9      RS="\n|\r\n"
10 }
11
12 function readcert(file) {
13     while ((getline < file) > 0) {
14         contents = contents RT $0
15         if ($1 == "-----BEGIN")
16             contents = $0
17         if ($1 == "-----END")
18             break
19     }
20
21     close(file)
22     return contents
23 }
```

```
24
25 $1 ~ /^(ca|key|cert|tls-auth)$/ {
26     tag = $1
27     print "#" $0
28
29     if (tag == "tls-auth"){
30         print "# for servers: key-direction 0"
31         print "key-direction 1"
32     }
33
34     print "<" tag ">"
35     print readcert($2)
36     print "</" tag ">"
37
38     next
39 }
40
41 {
42     print
43 }
44
45 AWK
```