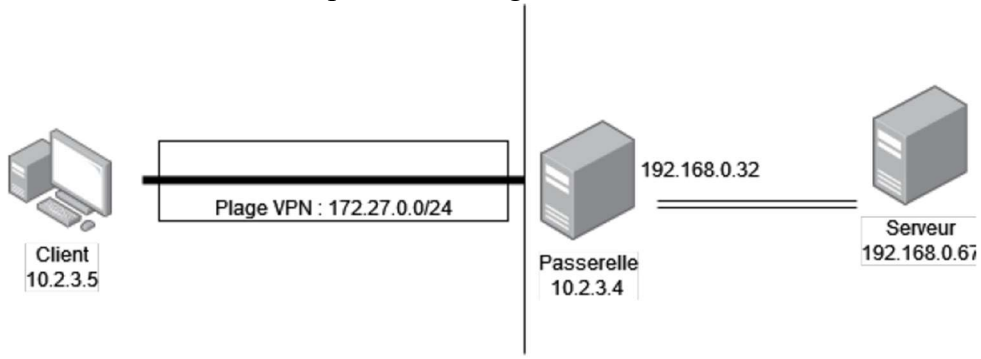
TP2: OpenVPN

Ifrim Vasile-Alexandru
M2SSI

The objective of this lab is to install and configure OpenVPN between a enterprise server (having an internal network) and its clients on remote PCs.

1. Environment and network setup

For our exercise, we will set up the following network:



The network consists of:

- an OpenVPN server and a OpenVPN client who are on a public network – 10.2.3.0/24
- a VPN zone, with a virtual address pool 172.27.0.0/24
- an internal zone, uniquely accessible only from the OpenVPN server, on 192.168.0.0/24

We will be using LXD for creating containers and managing networks. To create the networks:

```
$ lxc network create public0 ipv4.address=10.2.3.1/24
  ipv4.nat=true ipv6.address=none
$ lxc network create internal0 ipv4.address=192.168.0.1/24
  ipv4.nat=false ipv6.address=none
```

```
root Dec 31 13:17 .../alex/SystemsSecurity/TP2 > lxc network create public0 ipv4.address=10.2.3.1
/24 ipv4.nat=true ipv6.address=none
Network public0 created
root Dec 31 13:18 .../alex/SystemsSecurity/TP2 > lxc network create internal0 ipv4.address=192.16
8.0.1/24 ipv4.nat=false ipv6.address=none
Network internal0 created
```

Then, we create the containers, attach the corresponding network interfaces and set the given IP addresses:

```
$ lxc launch images:alpine/3.21 openvpn-server
$ lxc launch images:alpine/3.21 openvpn-client
$ lxc launch images:alpine/3.21 openvpn-intserver
```

```
$ lxc network attach public0 openvpn-client eth0
$ lxc config device set openvpn-client eth0 ipv4.address
  10.2.3.5
```

```
$ lxc network attach internal0 openvpn-intserver eth0
$ lxc config device set openvpn-intserver eth0 ipv4.address
  192.168.0.67
```

```
$ lxc network attach public0 openvpn-server eth0
$ lxc config device set openvpn-server eth0 ipv4.address
  10.2.3.4
$ lxc network attach internal0 openvpn-server eth1
$ lxc config device set openvpn-server eth1 ipv4.address
  192.168.0.32
```

Finally, we must restart the containers and wait a bit for the changes to take place.

```
$ lxc restart openvpn-client openvpn-server openvpn-intserver
```

```
root Jan 1 14:44 ~ > lxc ls
+-------------------+---------+---------------------+------+-----------+-----------+
|       NAME        |  STATE  |        IPV4         | IPV6 |   TYPE    | SNAPSHOTS |
+-------------------+---------+---------------------+------+-----------+-----------+
| mgr1              | STOPPED |                     |      | CONTAINER | 0         |
+-------------------+---------+---------------------+------+-----------+-----------+
| mgr2              | STOPPED |                     |      | CONTAINER | 0         |
+-------------------+---------+---------------------+------+-----------+-----------+
| mgr3              | STOPPED |                     |      | CONTAINER | 0         |
+-------------------+---------+---------------------+------+-----------+-----------+
| openvpn-client    | RUNNING | 10.2.3.5 (eth0)     |      | CONTAINER | 0         |
+-------------------+---------+---------------------+------+-----------+-----------+
| openvpn-intserver | RUNNING | 192.168.0.67 (eth0) |      | CONTAINER | 0         |
+-------------------+---------+---------------------+------+-----------+-----------+
| openvpn-server    | RUNNING | 192.168.0.32 (eth1) |      | CONTAINER | 0         |
|                   |         | 10.2.3.4 (eth0)     |      |           |           |
+-------------------+---------+---------------------+------+-----------+-----------+
```

Observation:

In our environment there were issues with having 2 interfaces on `openvpn-server`: after restarting, `eth1` wouldn't receive the configuration from the `dnsmasq` process handling the `internal0` interface: for this we had to manually add to /etc/network/interfaces the following line:

```
  iface eth1 inet dhcp
```

and put it up by:

```
$ ifup eth1
```

```
openvpn-server:~# cat /etc/network/interfaces
auto eth0
iface eth1 inet dhcp
iface eth0 inet dhcp
hostname $(hostname)
openvpn-server:~#
```

This will also modify `/etc/resolv.conf` by setting the nameserver to `internal0` – we presume it relies in the end just like `public0` on the nameserver of the host (if we want to be more correct, we would take down both interfaces, and put them back up in order, `eth1`, then `eth0`, so it uses `public0` as the nameserver).

We will continue by installing OpenVPN on the client and server:

```
$ apk add openvpn
```

```
openvpn-server:~# openvpn --version
OpenVPN 2.6.11 x86_64-alpine-linux-musl [SSL (OpenSSL)] [LZO
library versions: OpenSSL 3.1.7 3 Sep 2024, LZO 2.10
Originally developed by James Yonan
Copyright (C) 2002-2024 OpenVPN Inc <sales@openvpn.net>
```

2. Cryptography

We need to set up a Certificate Authority and generate TLS certificates for the OpenVPN server and client(s), as well as ECDH parameters for the key exchanges. On the `openvpn-server` and `openvpn-client`: we will initialize a new PKI:

```
$ apk add easy-rsa
$ mkdir -p /etc/openvpn/easy-rsa
$ ln -s /usr/share/easy-rsa/easyrsa /etc/openvpn/easy-
  rsa/easyrsa
$ cd /etc/openvpn/easy-rsa && ./easyrsa init-pki
```

```
openvpn-server:/etc/openvpn/easy-rsa# ./easyrsa init-pki

Notice
------
'init-pki' complete; you may now create a CA or requests.

Your newly created PKI dir is:
* /etc/openvpn/easy-rsa/pki

* Using Easy-RSA configuration:

* IMPORTANT: Easy-RSA 'vars' template file has been created in your new PKI.
             Edit this 'vars' file to customise the settings for your PKI.
             To use a global vars file, use global option --vars=<YOUR_VARS>

* Using x509-types directory: /usr/share/easy-rsa/x509-types
```

Normally, the secured CA environment must be on a separate system, but we will instantiate it here with OpenVPN server. We create the CA certificate (a password for encrypting it should be used in a real environment!):

```
$ ./easyrsa build-ca nopass
```

```
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.
-----
Common Name (eg: your user, host, or server name) [Easy-RSA CA]:m2ssi-iva

Notice
------
CA creation complete and you may now import and sign cert requests.
Your new CA certificate file for publishing is at:
/etc/openvpn/easy-rsa/pki/ca.crt

openvpn-server:/etc/openvpn/easy-rsa# ls -l pki/
total 23
-rw-------    1 root     root          1196 Jan  1 14:20 ca.crt
drwx------    2 root     root             2 Jan  1 14:20 certs_by_serial
-rw-------    1 root     root             0 Jan  1 14:20 index.txt
-rw-------    1 root     root             0 Jan  1 14:20 index.txt.attr
drwx------    2 root     root             2 Jan  1 14:20 inline
drwx------    2 root     root             2 Jan  1 14:20 issued
-rw-------    1 root     root          5043 Jan  1 14:13 openssl-easyrsa.cnf
drwx------    2 root     root             3 Jan  1 14:20 private
drwx------    2 root     root             2 Jan  1 14:13 reqs
drwx------    5 root     root             5 Jan  1 14:20 revoked
-rw-------    1 root     root             3 Jan  1 14:20 serial
-rw-------    1 root     root          9014 Jan  1 14:13 vars
-rw-------    1 root     root          9014 Jan  1 14:13 vars.example
```

On the server, we generate a keypair and request, left unencrypted – we will rely on protecting them just with access and file permissions. On the PKI for the openvpn-server we generate the DH parameters used during the TLS handshake with connecting clients.

```
$ ./easyrsa gen-req <unique_server_short_name> nopass
$ ./easyrsa gen-dh
```

```
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.
-----
Common Name (eg: your user, host, or server name) [vpnserver]:

Notice
------
Keypair and certificate request completed. Your files are:
req: /etc/openvpn/easy-rsa/pki/reqs/vpnserver.req
key: /etc/openvpn/easy-rsa/pki/private/vpnserver.key
```

```
.............++*++*++*++*++*++*++*++*++
++*++*++*++*++*++*++*++*++*++*++*++*++*
+*++*++*++*
DH parameters appear to be ok.

Notice
------

DH parameters of size 2048 created
at: /etc/openvpn/easy-rsa/pki/dh.pem
```

On the client, a keypair and request will be generated, this time with a password; the name selected must be unique across the PKI.

```
$ ./easy-rsa gen-req <unique_client_short_name>
```

```
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
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.
-----
Common Name (eg: your user, host, or server name) [vpnclient1]:

Notice
------
Keypair and certificate request completed. Your files are:
req: /etc/openvpn/easy-rsa/pki/reqs/vpnclient1.req
key: /etc/openvpn/easy-rsa/pki/private/vpnclient1.key

openvpn-client:/etc/openvpn/easy-rsa#
```

The request files of clients need to be sent to the CA system. This is not a security sensitive, though it is wise to verify the received file matches the sender's copy if the transport is untrusted. In our environment, we will copy the file from `openvpn-client`'s filesystem to `openvpn-server`'s, through the host.

```
$ lxc file pull <instance>/<path> <target path>
$ lxc file push <target path> <instance>/<path>
```

```
root Jan 1 19:04 .../alex/SystemsSecurity/TP2 > lxc file pull openvpn-client/etc/openvpn/easy-rsa
/pki/reqs/vpnclient1.req .
root Jan 1 19:04 .../alex/SystemsSecurity/TP2 > ls -l
total 12K
drwxr-xr-x 2 root root 4.0K Jan  1 19:04 ./
drwxrwxr-x 5 alex alex 4.0K Dec 30 19:14 ../
-rw------- 1 root root  891 Jan  1 19:04 vpnclient1.req
root Jan 1 19:04 .../alex/SystemsSecurity/TP2 > lxc file push vpnclient1.req openvpn-server/etc/o
penvpn/easy-rsa/pki/reqs/
root Jan 1 19:05 .../alex/SystemsSecurity/TP2 >
```

```
openvpn-server:/etc/openvpn/easy-rsa# ls -l pki/reqs/
total 3
-rw-------    1 root     root           891 Jan  1 18:05 vpnclient1.req
-rw-------    1 root     root           891 Jan  1 14:37 vpnserver.req
Show Apps ver:/etc/openvpn/easy-rsa#
```

Now, on the CA (`openvpn-server`), we can also import the entity's request file, giving it an arbitrary „short name", basically copying the request from wherever it is on the system into `pki/reqs/`:

```
$ ./easyrsa import-req <path_to_req> <unique_short_file_name>
```

We review each request's details, then sign them as one of the 2 types: server or client. The created certificates are located in `pki/issued/`.

```
$ ./easyrsa show-req <unique_short_file_name>
$ ./easyrsa sign-req client vpnclient1
$ ./easyrsa sign-req server vpnserver
```

```
openvpn-server:/etc/openvpn/easy-rsa# ./easyrsa sign-req server vpnserver

* Using SSL: openssl OpenSSL 3.1.7 3 Sep 2024 (Library: OpenSSL 3.1.7 3 Sep 2024)

* Using Easy-RSA configuration: /etc/openvpn/easy-rsa/pki/vars

You are about to sign the following certificate.
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a server certificate for 825 days:

subject=
    commonName                = vpnserver


Type the word 'yes' to continue, or any other input to abort.
  Confirm request details: yes

Using configuration from /etc/openvpn/easy-rsa/pki/b109ff4a/temp.b6d8e996
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'vpnserver'
Certificate is to be certified until Apr  6 18:24:24 2027 GMT (825 days)

Write out database with 1 new entries
Database updated

Notice
------
Certificate created at:
* /etc/openvpn/easy-rsa/pki/issued/vpnserver.crt
```

If we want to verify that the certificates indeed have the right key usages, keyEncipherment, digitalSignature and server authentication:

```
$ openssl x509 -in /etc/openvpn/server.crt -text -noout | grep
  -A1 "X509v3"
```

```
openvpn-server:/etc/openvpn/easy-rsa# openssl x509 -in pki/issued/vpnserver.crt -text -noout | gr
ep -A1 "X509v3"
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Key Identifier:
                10:5B:76:03:47:99:93:FD:D2:C4:BB:FB:53:B5:B6:F2:20:DE:42:D0
            X509v3 Authority Key Identifier:
                keyid:C3:61:5C:6C:7B:A8:D1:35:E5:43:F5:D7:C0:4D:DE:82:86:D6:F1:FA
--
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Key Usage:
                Digital Signature, Key Encipherment
            X509v3 Subject Alternative Name:
                DNS:vpnserver
```

Final step is returning the signed certificates produced, including the CA certificate, `ca.crt` (in our case we have to return just the files for the client). They are found in `pki/issued/`.

3. Simple configuration files

For easier access to neccesary files, we will copy the following from `easy-rsa/pki/` to `/etc/openvpn/` :

```
openvpn-server:/etc/openvpn# cp easy-rsa/pki/ca.crt easy-rsa/pki/dh.pem easy-rsa/pki/private/vpns
erver.key easy-rsa/pki/issued/vpnserver.crt .
openvpn-server:/etc/openvpn# ls -l
total 14
-rw-------    1 root     root          1196 Jan  1 20:32 ca.crt
-rw-------    1 root     root           428 Jan  1 20:32 dh.pem
-rwxr-xr-x    1 root     root           960 Jun 29  2024 down.sh
drwxr-xr-x    3 root     root             4 Jan  1 14:13 easy-rsa
-rw-r--r--    1 root     root         10816 Jan  1 20:31 server.conf
-rwxr-xr-x    1 root     root          2612 Jun 29  2024 up.sh
-rw-------    1 root     root          4615 Jan  1 20:32 vpnserver.crt
-rw-------    1 root     root          1704 Jan  1 20:32 vpnserver.key
```

The full server configuration, with specifications, is attached in this repo, but the essential parts are:

```
proto tcp
dev tun
ca /etc/openvpn/ca.crt
cert /etc/openvpn/vpnserver.crt
key /etc/openvpn/vpnserver.key
dh /etc/openvpn/dh.pem
verb 5
server 172.27.0.0 255.255.255.0
push "route 192.168.0.0 255.255.255.0"
```

Some short details on the `server.conf` provided:

- `proto tcp` - specifies the TCP protocol for the VPN connection
- `dev tun` - uses a TUN device for routing (IP-based)
- `ca` - path to the CA certificate
- `cert` - server certificate
- `key` - server private key

- dh: Diffie-Hellman parameters for secure key exchange
- server <vpn_net> <submask> - VPN subnet, clients will use the range for their virtual IPs
- push "route <internal_net> <submask>" - allows clients to reach the internal subnet
- client-to-client - enables direct communication between VPN clients

For the client configuration, client.conf, attached in this repo, the essential is:

```
client
dev tun
proto tcp
remote 10.2.3.4 1194
ca /etc/openvpn/ca.crt
cert /etc/openvpn/client1.crt
key /etc/openvpn/client1.key
persist-key
persist-tun
```

4. Lancement du VPN

To prepare autostart of OpenVPN on the openvpn-server container(alpine distro):

```
$ rc-update add openvpn default
```

Because the internal network does not have a direct route back to the VPN subnet, the server must masquerade traffic through iptables. Also, IP forwarding must be enabled:

```
$ apk add iptables
$ iptables -t nat -A POSTROUTING -s 172.27.0.0/24 -d
  192.168.0.0/24 -j MASQUERADE
$ echo "net.ipv4.ip_forward = 1" >> /etc/sysctl.d/ipv4.conf
$ sysctl -p /etc/sysctl.d/ipv4.conf
```

Finally, we can launch openvpn on both containers:

```
$ openvpn --config /path/to/config
```

And from the client we should be able to ping the server on the internal network:

```
openvpn-client:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500
fault qlen 500
    link/none
    inet 172.27.0.2/24 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::a2c5:e078:1b4e:7998/64 scope link stable-p
       valid_lft forever preferred_lft forever
48: eth0@if49: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
en 1000
    link/ether 00:16:3e:4c:0f:4a brd ff:ff:ff:ff:ff:ff lin
    inet 10.2.3.5/24 brd 10.2.3.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::216:3eff:fe4c:f4a/64 scope link
       valid_lft forever preferred_lft forever
openvpn-client:~# ping 192.168.0.67
PING 192.168.0.67 (192.168.0.67): 56 data bytes
64 bytes from 192.168.0.67: seq=0 ttl=63 time=8.025 ms
64 bytes from 192.168.0.67: seq=1 ttl=63 time=10.632 ms
64 bytes from 192.168.0.67: seq=2 ttl=63 time=40.329 ms
```