Security Insider Lab I - Report 4

by

Subbulakshmi Thillairajan, Fabian Göttl, Mohamed Belkhechine

Practical Part

Exercise 1:

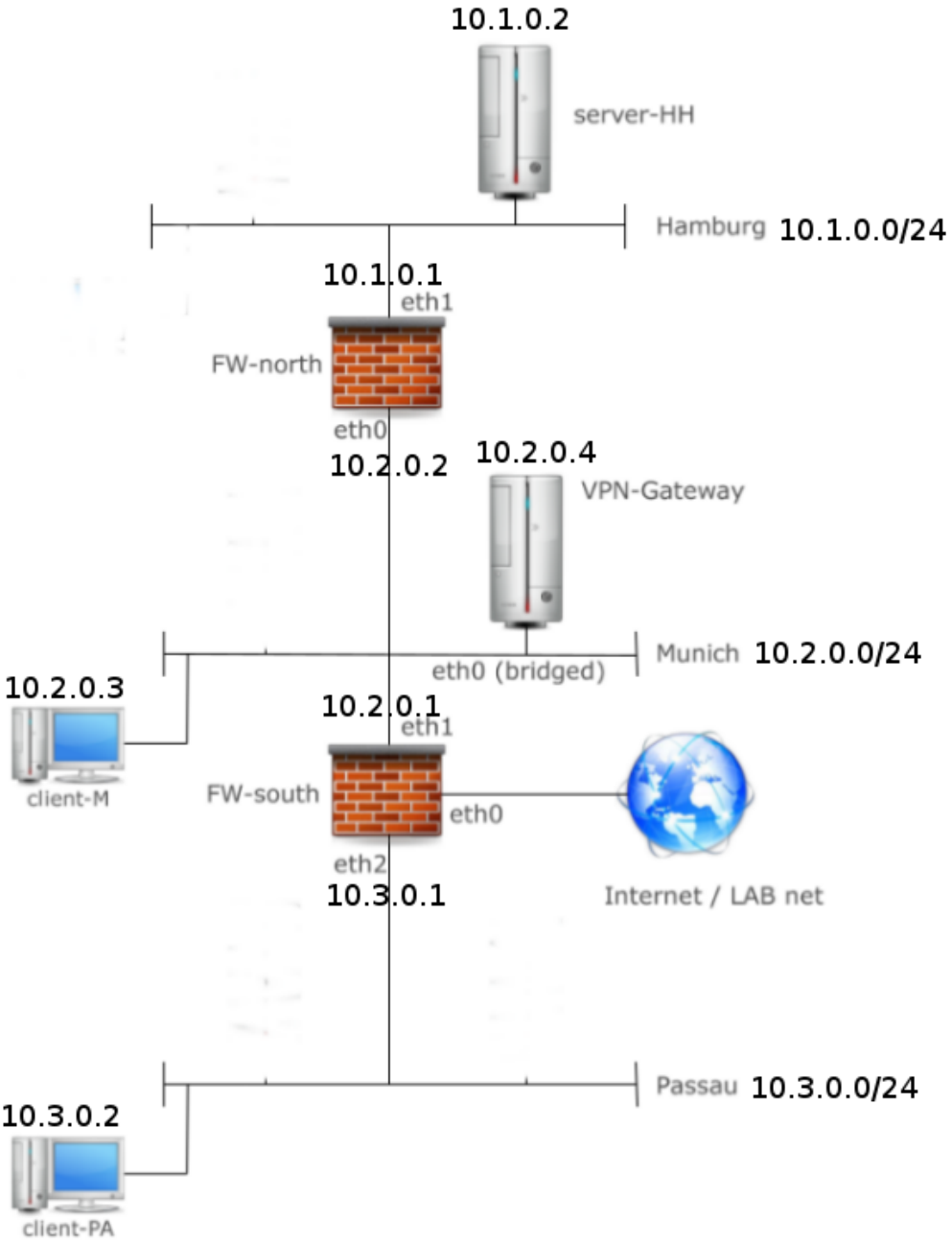

Figure 1: Network overview

Install openvpn package at VPN-gateway and host machine:
sudo apt-get install openvpn

Tun interface

First, we use the simplest setup of creating a VPN connection between Host machine and VPN-gateway. Here, we create a tun interface and use static symmetric keys.

Generate static symmetric key:
openvpn --genkey --secret /etc/openvpn/static.key

Transfer key to client with scp (secure SSH connection):
scp osboxes@10.2.4:/etc/openvpn/static.key /etc/openvpn/

Copy server default config files (at VPN-gateway):
sudo cp /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz /etc/openvpn/
sudo gunzip /etc/openvpn/server.conf.gz

Copy client default config files (at Host machine):
sudo cp /usr/share/doc/openvpn/examples/sample-config-files/client.conf /etc/openvpn/

Adjust file /etc/openvpn/server.conf (at VPN-gateway):
dev tun
ifconfig 10.8.0.1 10.8.0.2
secret static.key
push "route 10.2.0.0 255.255.255.0"
proto tcp-server
port 443

Adjust file /etc/openvpn/client.conf (at Host machine):
dev tun
remote 10.2.0.4
ifconfig 10.8.0.2 10.8.0.1
secret static.key
route 10.2.0.0 255.255.255.0
proto tcp-client
port 443

Add firewall rules (at FW-south):
In order to enable port forwarding for firewall south we did the following three steps:
1- Adding the forwarding rule:
iptables -t nat -A PREROUTING -p tcp --dport 443 -j DNAT --to-destination 10.2.0.4:443
with **10.2.0.4:443** the physical ip address of the vpn-gateway machine.
2- Masquerading iptables:
iptables -t nat -A POSTROUTING -j - MASQUERADE
3- Enabling ip_v4 packet forwarding:
sysctl -w net . ipv4 . ip_forward =1

A VPN connection between host machine and VPN-gateway was successful. We connected by using the external (eth0) IP of FW-south. This firewall is redirecting the in-going traffic at eth0 to the internal IP 10.2.0.4 of the VPN gateway. The tun interface created a own VPN subnet 10.8.0.0/24, where 10.8.0.1 was the gateway and 10.8.0.2 the client. We could successfully ping between the machines.

Tap interface

In order to enable VPN access to the whole 10.2.0.0/24 Munich network, we create a tap interface. This method will route Layer 2 traffic between the server and connected VPN sites. The clients are allowed to use a IP within the Munich subnet.

Create and run bridge-start file (at VPN-gateway):

```
#!/bin/bash

# Define Bridge Interface
br="br0"

# Define list of TAP interfaces to be bridged,
# for example tap="tap0 tap1 tap2".
tap="tap0"

# Define physical ethernet interface to be bridged
# with TAP interface(s) above.
eth="enp0s3"
eth_ip="10.2.0.4"
eth_netmask="255.255.255.0"
eth_broadcast="10.2.0.255"

for t in $tap; do
        openvpn --mktun --dev $t
done

brctl addbr $br
brctl addif $br $eth

for t in $tap; do
        brctl addif $br $t
done

for t in $tap; do
        ifconfig $t 0.0.0.0 promisc up
done

ifconfig $eth 0.0.0.0 promisc up

ifconfig $br $eth_ip netmask $eth_netmask broadcast $eth_broadcast
```

Create bridge-stop file (at VPN-gateway):

This is only required if the interface should be deleted.

```
#!/bin/bash

# Define Bridge Interface
br="br0"

# Define list of TAP interfaces to be bridged together
tap="tap0"
```

ifconfig $br down
brctl delbr $br

for t in $tap; do
  openvpn --rmtun --dev $t
done

After creating the bridge interface, the old TCP/IP settings of the eth0 interface are not used anymore. Hence, the internet connection was lost and we could not connect to the VPN server.

We restore the connection by adding the default gateway:
sudo route add default gw 10.2.0.1

Further, a restart of the networking service was required:
sudo /etc/init.d/networking restart

Add firewall rules in VPN-gateway:
iptables -A INPUT -i tap0 -j ACCEPT
iptables -A INPUT -i br0 -j ACCEPT
iptables -A FORWARD -i br0 -j ACCEPT

File /etc/openvpn/server.conf:
dev tap0
port 443
proto tcp-server
server-bridge 10.2.0.4 255.255.255.0 10.2.0.50 10.2.0.100
push "route 10.1.0.0 255.255.255.0"
push "route 10.3.0.0 255.255.255.0"
secret static.key

The server-bridge creates a bridge between tap0 and the remote tap interface. The line starts with the gateway IP address and the subnet. We reserve a client IP space from 10.2.0.50 to 10.2.0.100.

File /etc/openvpn/client.conf:
client
dev tap
proto tcp-client
remote FW-south 443
secret static.key

A VPN connection between host machine and VPN-gateway was successful. We connected by using the external (eth0) IP of FW-south. We could successfully ping between the machines.

Exercise 2:

We use easy-rsa to create the required SSL and CA certificates:
sudo apt-get install easy-rsa

Copy default configs:
sudo cp -r /usr/share/easy-rsa /etc/openvpn/easy-rsa2

Editing **/etc/openvpn/easy-rsa2/vars.**

Symlink from version config to default config name:
sudo ln -s openssl-0.9.6.cnf openssl.cnf

Add vars to environments variables:
source ./vars

Build ca cert:
sudo -E ./clean-all
sudo -E ./build-ca

Build server cert:
sudo -E ./build-key-server VPN-Gateway.group6
We give no challenge password, otherwise a connection would be possible with this password. Further, we press 2x Y.

Build client cert:
sudo -E ./build-key-pass mclient
Common Name (CN): mclient
Other fields like Organization, Contact, Country were created with example data.
We set password as *test123*. We give no challenge password, otherwise a connection would be possible with this password. Further, we press 2x Y.

Pack the required files for a client:
tar -cf mclient.tar mclient.key mclient.crt ca.crt

server.conf:
dev tap0
port 443
proto tcp-server
server-bridge 10.2.0.4 255.255.255.0 10.2.0.50 10.2.0.100
push "route 10.1.0.0 255.255.255.0"
push "route 10.3.0.0 255.255.255.0"
ca ./easy-rsa2/keys/ca.crt
cert ./easy-rsa2/keys/VPN-gateway.group6.crt
key ./easy-rsa2/keys/VPN-gateway.group6.key
dh ./easy-rsa2/keys/dh2048.pem

Client.conf:
client
dev tap

proto tcp-client
remote FW-south 443
ca ca.crt
cert mclient.crt
key mclient.key

Reload configs and restart openVPN:
sudo systemctl daemon-reload
sudo service openvpn restart

After copying the openvpn config files from the other team, we ran this command: **openvpn client.conf** from the other team host machine as depicted in the following figure:

```
root@bmpc:/etc/openvpn# openvpn client.conf
Wed Nov 30 15:20:54 2016 OpenVPN 2.3.10 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO
] [EPOLL] [PKCS11] [MH] [IPv6] built on Feb  2 2016
Wed Nov 30 15:20:54 2016 library versions: OpenSSL 1.0.2g  1 Mar 2016, LZO 2.08
Enter Private Key Password: *******
```

Figure 2: Openvpn client run command.

After successfully being connected to the VPN, we test the reachability of server-hh.

```
mbelkhechine@bmpc:~$ ping 10.1.2
PING 10.1.2 (10.1.0.2) 56(84) bytes of data.
From 10.2.0.5: icmp_seq=1 Redirect Host(New nexthop: 10.2.0.1)
64 bytes from 10.1.0.2: icmp_seq=1 ttl=63 time=82.1 ms
From 10.2.0.5: icmp_seq=2 Redirect Host(New nexthop: 10.2.0.1)
From 10.2.0.1: icmp_seq=2 Redirect Host(New nexthop: 10.2.0.2)
64 bytes from 10.1.0.2: icmp_seq=2 ttl=63 time=8.95 ms
From 10.2.0.1: icmp_seq=9 Redirect Host(New nexthop: 10.2.0.2)
From 10.2.0.5: icmp_seq=9 Redirect Host(New nexthop: 10.2.0.1)
64 bytes from 10.1.0.2: icmp_seq=9 ttl=63 time=65.9 ms
```

Figure 3: Ping 10.1.2

We have been able to connect to the web server in the machine server-hh.

Figure 4: Access to web server.

And finally we tested ssh connection:

```
mbelkhechine@bmpc:~$ ssh osboxes@10.1.2
The authenticity of host '10.1.2 (10.1.0.2)' can't be established.
ECDSA key fingerprint is SHA256:Tiifum9mV+8uMUNUxRQxKbD9fYxaizzvvTwEgljltis.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.1.2,10.1.0.2' (ECDSA) to the list of known hosts.
osboxes@10.1.2's password:
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 4.4.0-47-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

2 packages can be updated.
0 updates are security updates.

Last login: Fri Nov 18 08:46:51 2016 from 10.3.0.2
osboxes@server-HH:~$ exit
logout
Connection to 10.1.2 closed.
```

Figure 5: SSH connection test.

Theoretical part

Exercise 1

Exercise 1.1:

IPsec:

- IPsec works on Network Layer. It secures all data that flows between two end points without an association to any specific application. Once connected, the person will be virtually connected to the entire network.
- It defines how to provide data integrity, authenticity and confidentiality over insecure network like Internet.
- It completes its goal through tunneling, Encryption and Authentication.

  1. It is complex because the two entities which will communicate via IPSEC have to agree on same security policies which must be configured on the both end of the devices.

  2. A Single IPSec tunnel secures all the communication between the devices regardless of traffic type. It can be TCP, UDP, ICMP etc or any application like e-mail, client-server, database.

SSL:

- SSL works on Application Layer.
- It is a protocol used for secure web-based communication over the Internet(by using encryption and authentication).
- SSL also provides flexibility by providing level of security.

  1. SSL helps to secure one application at a time.

  2. It requires upgrading which is very cost consuming.
- Above problem can be resolved by purchasing SSL VPN gateway which is deployed at the edge of the corporate network and serve as a proxy to LAN.

  The browser thinks it is directly communicating with the application and application thinks it is directly communicating with browser thus making it transparent.

OpenVPN implements SSL/TLS encryption.

Exercise 1.2:

The reason to choose **Port 443** are,

- Port 443 is the default port for SSL-encrypted HTTP traffic and is allowed on most firewalls.
- It enables us to route traffic through firewalls or SSL proxies.
- It is used for security purposes and provides simplicity.

Exercise 1.3:

TUN (network TUNnel)

- TUN simulates a network layer device and it operates with layer 3 packets like IP packets.
- TUN is used with routing.
- A typical use for a TUN device is establishing VPN connections since it gives the VPN software a chance to encrypt the data before it gets put on the wire.
- Since a TUN device works at layer 3 it can only accept IP packets and in some cases only IPv4.

TAP (network tap)

- TAP simulates a link layer device and it operates with layer 2 packets like Ethernet frames.
- TAP is used for creating a network bridge of a subnet.
- TAP devices are commonly used in virtualization systems to provide virtual network adaptors to multiple guest machines.

- Since they are running at layer 2 they can transport any layer three protocol and aren't limited to point-to-point connections.
- An advantage is that broadcasts are redirected to the VPN sites.

Exercise 1.4:
Virtual network device can be viewed as a simple Point-to-Point or Ethernet device, which instead of receiving packets from a physical media,it receives them from user space program and instead of sending packets via physical media sends them to the user space program.The kernel handles the packet like it's from real physical device.

Exercise 1.5:
VPN-Endpoints :
- Endpoints are the devices that are connected via tunnel.
- The traffic that passes between the two endpoints is encrypted. Instead of a vpn client passing through a router and connecting to a machine(windows,linux), the connection goes from one VPN endpoint device to another.
- Thus, allowing to tunnel between 2 separate networks that are connected to the firewall as rather than just tunneling into a specific computer or server.

The endpoints are VPN Gateway and host (tap0 interface).

Exercise 1.6:
In case of using TAP :
In order to route traffic that is not addressed directly to the VPN-gateway, but intended for the VPN clients, the gateway has the see all packets. Hence, the physical network interface state is in promiscuous mode.

In case of using TUN:
The state of a physical network interface shows if the device is connected to a cable or if it is up/down. The state has to be connected for the VPN to work.

Exercise 2

Exercise 2.1:
The advantages:
- Allows computers to authenticate each other without any prior contact.
- Public key infrastructures (PKIs) helps to be certain of the identity of different people, devices, and services.
- Limiting access to certain hostnames, time spans
- Managing access can be directed to a third-party Certificate Authority (CA)

The disadvantages:
- Effort in managing the client certificates.
- Thorough understanding of asymmetric encryption principles required.
- Asymmetric encryption is slow and time consuming.

Exercise 2.2:
For connectivity between two sites we would suggest to use TAP as it maps the complete Munich network to the VPN without setting up manual routes. Here, the VPN config enable us to specify a IP address pool in Munich subnet reserved for the connecting clients. Further, broadcasts within the Munich network get redirected to the VPN. A disadvantage is that firewalling within the network is more difficult.