Firewall Evasion and VPN Usage Lab

(M.EEC/SSR,M.EIC/SR@FEUP)

1 Overview

VPNs (Virtual Private Networks) are an important topic in network security as they are often used to securely extend a private network into an insecure one such as the Internet. VPNs are commonly associated with access control, confidentiality by means of cryptography, and traffic tunneling. They are often seen as a tool that allows one to "mask" their source IP addresses.

In this lab students will explore VPNs as a tool, without focusing on their implementation, to:

- Access private resources from an outside location.
- Securely access plaintext content that would otherwise be easily eavesdropped.
- Change their source IP address when browsing external resources.
- Ultimately, be aware of the security consequences of using VPNs.

2 Task 0: Lab Setup

We will conduct multiple experiments with a computer (VPN client) accessing resources on a private network through a VPN server. To test multiple scenarios and configurations, we use two routers and an external web server. Like in previous Labs, these are containers that you launch using `dcup` and log into using `docksh`.

All machines and networks are depicted in Figure 1. Take a moment to familiarize yourself with the setup. You will set up NAT in both routers and you will block access to both routers' private networks using `iptables` rules. Afterwards, you will be accessing resources on the `internal_net` from the external host

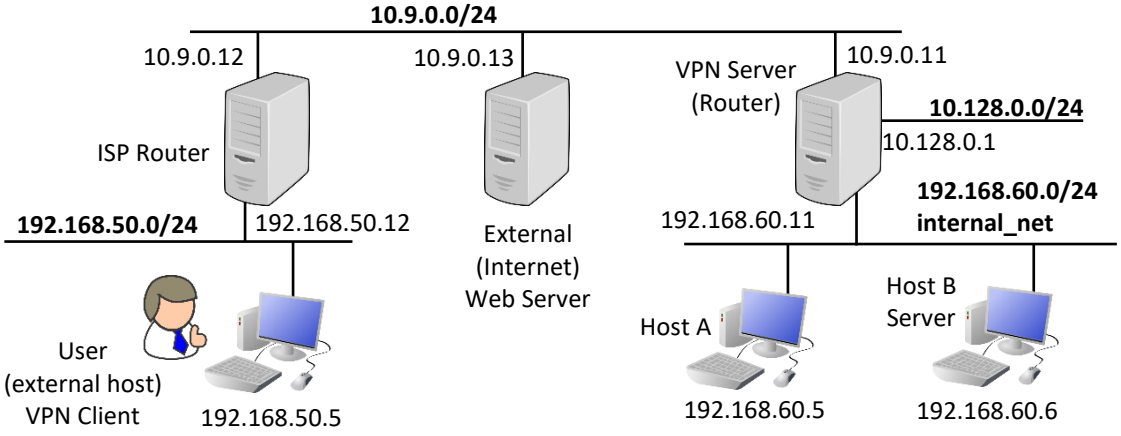

Figure 1: Lab Environment Setup

[1] https://seedsecuritylabs.org/Labs_20.04/Networking/VPN_Tunnel/
[2] https://seedsecuritylabs.org/Labs_20.04/Networking/Firewall_Evasion/
[3] https://github.com/mfthomps/Labtainers/tree/master/labs/vpnlab2

using a VPN. The initial network configuration is partially done (routes on all devices and services running, but no NAT and no firewall).

Lab task. Make sure you can access all devices from the user/external machine. Use curl to access the Web server on the External Web Server and on the internal Host B. They both serve a basic web page that reports the HTTP client's IP address (as perceived by them), like the many "What Is My IP" websites you find online. With `curl`'s `-i` option, you can set the source interface when there are multiple available (you don't need it for now, but you'll use it later):

```
# curl http://10.9.0.13 --interface eth0
...Your IP is ...
```

What IP do the servers report before setting up NAT and firewall rules? What about afterwards? (you find the instructions to set up NAT and firewall rules below). To make sure everything is working as intended try the following requests before NAT and firewall:

- From User to External Web Server
- From User to Host B
- From External Web Server to Host B

And after setting up NAT and firewall:

- From User to External Web Server
- From User to Host B
- From Host B to External Web Server (and the other way round)

Compare the response you get from the External Web Server while doing the request from Host B and from the User. You can also perform traffic sniffing on the ISP Router using `tcpdump`. What do you observe?

```
# tcpdump -n -i eth0 -X
```

Router configuration: setting up NAT. The following `iptables` command sets up a NAT on the router for traffic going out from its `eth0` interface. With this rule, for packets going out to the "Internet", their source IP address will be replaced by the router's IP address, `10.9.0.XX`, assuming that eth0 is the name of the interface connecting the router to the 10.9.0.0/24 network (which is not always the case).

```
iptables -t nat -A POSTROUTING -j MASQUERADE -o eth0
```

You can find the correct interface names with the following command:

```
# ip -br address
lo               UNKNOWN         127.0.0.1/8
eth1@if1907      UP              192.168.50.12/24
eth0@if1909      UP              10.9.0.12/24
```

Router configuration: Firewall rules. We will be blocking all ingress traffic on both routers, as is typically done on many residential and enterprise routers. Hosts on private networks must reach the Internet, but not the other way round. The following command makes the router accept forwarding traffic for new connections from the inside. Then, we also accept traffic belonging to already ESTABLISHED or RELATED connections. Finally, we drop all traffic that doesn't fit either of those rules:

```
iptables -A FORWARD -o eth0 -m conntrack --ctstate NEW -j ACCEPT
iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -P FORWARD DROP
```

Cleanup. If needed, you can restore the filter table to its original state with the following commands (go back to SEED's "Firewall Exploration Lab" if you need any extra reminding):

```
iptables -F
iptables -P OUTPUT ACCEPT
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
```

3 Task 1: SSH for Simple Tunneling

SSH can easily be used for tunneling: if the host you are login into has access to a service running on a certain port (e.g., you are host A login into host B; host B has access to a service running on host T, port Y), you can make an SSH tunnel between a port on your machine and that target port. Using SSH for tunneling or redirecting ports is very common practice and has the added advantage of providing encryption:

```
$ ssh -4NT -L <A's IP>:<A port X>:<T's IP>:<T's port Y> <user>@<B IP>
// -4: use IPv4 only, or we will see some error message.
// -N: do not execute a remote command.
// -T: disable pseudo-terminal allocation (save resources).
```

Regarding A's IP, it is the IP address that our port forwarding will listen/bind to – we can use `0.0.0.0` to listen to all of A's interfaces (on port X), or we can limit the connection to a specific interface by using that interface's IP address (e.g., using `127.0.0.1:<port>` will limit the connection to the loopback interface, which is the default). When data is received by host A on the specified IP and port, that data is sent to B, which will then forward it to host T, on port Y.

Use the previous command to ssh into the VPN Server Router and connect to the web server at Host B. What is the IP reported by Host B? Perform traffic sniffing on the ISP router. Can you observe the traffic?

Can you also use a similar command to telnet into Host A? Demonstrate. Remember that telnet is an insecure protocol (all information, including passwords, is sent in plaintext). What is the consequence of using an SSH tunnel to telnet into Host A regarding the ISP Router's ability to eavesdrop on our traffic? What about the VPN Server's ability to capture traffic?

3.1 Task 1.1: Advanced SSH Firewall Bypass / VPN

SEED's Firewall Evasion Lab (Task 3) showcases a more advanced use of SSH that creates a virtual interface (`tun0`) on both VPN client and server machines, and then connect these two TUN interfaces using an encrypted TCP connection, closely resembling the implementation of a typical VPN.

This option requires the privilege of creating TUN interfaces on both machines, which usually means having root privilege on both server and client and having root login enabled on the server, which is uncommon and considered to be a bad practice. Those changes would be done on `/etc/ssh/sshd_config`:

```
PermitRootLogin  yes
PermitTunnel     yes
```

Then, the tunnel could be created with the command below, followed by network configuration commands (IP addressing and routing):

```
# ssh -w 0:0 root@<VPN Server's IP>
```

We won't explore this option. We'll use OpenVPN instead.

4 Task 2: OpenVPN

An OpenVPN server is already present and running on the VPN Server Router machine; OpenVPN also provides encryption and authentication by default. (Although nowadays uncommon, you can also have tunneling without encryption, or even client authentication. Can you think of use cases for such a setup?)

OpenVPN is convenient and configurable, allowing for many different scenarios and use cases. You can find more information on their manuals. https://openvpn.net/community-resources/how-to/

4.1 Task 2.1: Internal and external resources through the VPN

The User machine already has OpenVPN installed, and all the files required to establish a connection to the VPN Server, namely a configuration file where the server IP is specified as well as some parameters, and files allowing for client and server authentication. Connect to the VPN Server using the provided configuration, checking the available network interfaces before and after:

```
# ip -br address
lo               UNKNOWN          127.0.0.1/8
eth0@if1907      UP               192.168.60.5/24
# cd /vpnfiles/
# openvpn --config client.conf   (this is blocking, open another terminal)
# ip -br address
```

As suggested, check your client's network interfaces before and after connecting to the VPN. What differences do you notice?

Accessing internal/private resources: Verify if you can reach internal resources using the VPN. What IP is reported by Host B? Can the ISP Router see this traffic?

Accessing external resources: Can you also reach the External Web Server through the VPN? What network configuration is required to access all resources (internal and external) through the VPN? (make sure you can always reach the VPN Server from outside the tunnel – in fact, that should be the only traffic not going through the tunnel). What IP do you expect the External Web Server to report? Confirm if that's the case.

Security and privacy. VPNs are often advertised (especially by commercial VPN providers) as a way of hiding your identity online (by changing your public IP address) and protecting your information from eavesdroppers when accessing insecure resources (by tunneling traffic through an encrypted channel). Is that the case of our VPN? You can use `tcpdump` on the ISP Router, and on the VPN Server Router to draw your conclusions.

Momentarily losing your VPN connection (e.g., in between HTTP requests) may "leak" your real public IP address, and the same could happen with improper network configurations even if the VPN connection is active. Proper network configuration and error handling are crucial (but also complex) when using VPNs for privacy protection. You can search the web for "VPN WebRTC leaks" to find out more about a vulnerability in browsers that may expose users' IP addresses even when using VPNs.

4.2 Task 2.2: Split-Tunneling

Sometimes a user wants to access private/internal resources using a VPN but, at the same time, preserve their public IP address when accessing external addresses. This is typical in many enterprise settings, where users want to access internal resources without their whole Internet traffic going through the company's routers. In fact, there are scenarios where access to the Internet through a VPN is discouraged or even forbidden. (Can you think of reasons why, considering performance or legal aspects?)

This approach is commonly known as split-tunneling, and it can be implemented in our User Host by specifying routes. To properly test it, issue `iptables` rules at the VPN Server Router blocking all machines

connected to the VPN (`10.128.0.0/24` network) from accessing external resources ("the Internet"). Make sure that's the case:

- Hosts on `10.128.0.0/24` shouldn't be able to reach `10.9.0.0/24`.
- Hosts on `192.168.60.0/24` should be able to reach `10.9.0.0/24`.
- Hosts on `192.168.60.0/24` and hosts on `10.128.0.0/24` should reach each other.

While connected to the VPN as in the previous task, try to reach the internal and external resources at the same time. What IP and content does each server (Host B and External Web Server) report? What traffic can be eavesdropped, and by which router?

4.3 Task 2.3: Problematic tunneling

Consider that a network administrator has blocked Internet access for certain machines since they are running critical services that are known to have vulnerabilities. Access to those services is granted only to specific users by means of a VPN. However, if a machine is connected to the VPN from the outside, then it must be connected to the Internet at the same time, creating an opportunity for outside communication, particularly if the network configuration is done poorly.

Make sure that you're still in a split-tunneling configuration, with your User machine having access to internal and external resources through different network interfaces (Host B and External Web Server report different IP addresses). Now, place new firewall rules to make sure that internal machines (`192.168.60.0/24`) do NOT have access to external resources (External Web Server).

On an internal machine, establish a tunnel to the External Web Server using SSH via the VPN Client – here, you'll use SSH as you did on Task 1 (you don't need the commands from Task 1.1). This time you will use SSH from an internal machine (SSH client), to the User/VPN Client machine (SSH server) and you will set the External Web Server as target (remember to specify the user – `seed`, with password `dees`). Through this connection, a previously compromised machine could, for example, upload/exfiltrate information to outside servers.

Can you implement rules on the VPN Server Router to prevent this traffic? Would this external access be possible if the client was not using a split-tunnel configuration (and instead used a configuration like the one in Task 2.1)?