FIT9137 Workshop Week 11

Topics

- Firewall
- Virtual Private Networks (VPNs)

Covered Learning Outcomes:

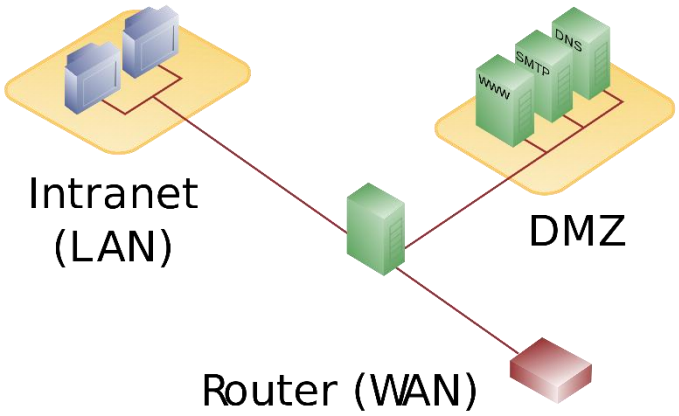
- identify and describe fundamental concepts of network security mechanisms against common threats and countermeasures.

Instructions

- One of the main purposes of an applied session is to build the learning community, create connections and include the learners. The other goal is to give and receive feedback from your peers and or your tutors.
- Form groups of 2 students (online) or more for face-to-face to work through the exercises. If met a problem, try to solve it by asking direct questions to your peer. If the issue was not solved within peers, ask your tutor. If did not get a chance to solve the problem during your applied session with your peer or tutor, jump into one of many consultation hours and ask any of the tutors to help you. Please visit the "Teaching Team and Unit Resources" tile in the FIT9137 Moodle site.

• ACTIVITY A: Firewall

In this exercise we will configure the firewall service on a gateway router in a sample core configuration file. Download the file **`FIT9137_Week-11_Activity_Firewall.imn`** from Moodle and save it in the shared folder on the host machine (your laptop or PC). Open core and from the file menu open the downloaded core configuration from the shared folder in the VM (under `/media` folder starting with `sf_`).



The node `phoenix` is the gateway router with its `eth0` connected to the Internal network of the organization, the `eth1` interface to the Demilitarized Zone (DMZ), and the `eth2` connected to the Internet. The firewall will protect the two networks: Internal, and DMZ against unauthorized traffic by inspecting the packets that pass through the router (or firewall).

The DMZ network is part of the organization's network where servers that provide services to the Internet will be located. The DMZ network is separated from the Internal network (or intranet) since it is possible for these servers to be compromised by an attacker and then be used to attack the rest of the network hence the firewall protects the Internal network against both of the Internet and DMZ. Check out the https://en.wikipedia.org/wiki/DMZ_(computing) for more information. The diagram of a typical three-legged network model employing a DMZ using a single firewall is used in our workshop activity.

Firewall Policy

Open the Services configuration for the firewall router (node `phoenix`) and click the tool icon next to the Firewall setting (in the Security column). This will open a dialogue that lets you enter firewall rules based on the `iptables` tool (`man iptables`).

Enter the following as the basis for your firewall rules:

STEP 1. Default Policy

```
#!/bin/sh
# Set default policy: drop all packets
# This means that the firewall blocks all traffic for the identified chain
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

After setting up this firewall, you should not be able to reach any computer inside the DMZ or the Internal network. In fact no traffic will be allowed to pass through, be sent from, or be delivered to the firewall.

The `iptables` commands use different chains for different tasks. The `INPUT` chain is for packets that have the firewall router as its destination. The `OUTPUT` chain is for packets that software on the firewall router wants to send somewhere else. The FORWARD chain controls the packets for which the firewall router performs routing. When a packet is passing through the firewall, entering from one interface, routed by the firewall, and then exiting from another interface, then the packet is checked against the `FORWARD` chain rules.

STEP 2. External to DMZ

Now that the system is secure (all traffic is blocked), we can start allowing traffic for individual services to pass through. We will start by allowing the servers in DMZ to be accessible to external hosts.

For the configuration to persist in core we need to use the GUI however as we are learning the `iptables` rules it would be better to try the commands while the core configuration is running. This way we can test the rules and make sure the rules correctly match the intended traffic and then we can add the rules to the configuration via GUI.

Note: Keep a copy of the tested and correct rules in a text file so that you can add them to the Firewall service through GUI.

To test the policy, we just configured start the emulation and open a terminal on `helios` and try to visit the web server in DMZ:

```
lynx 10.1.1.71
```

STEP 3. Incoming Rule

Now we add a rule that allows TCP packets for port 80 to pass from outside to web server in DMZ. Open a terminal on `phoenix` and enter the following command:

```
iptables -A FORWARD -i eth2 -o eth1 -d 10.1.1.71 -p tcp --dport 80 -j ACCEPT
```

The command line arguments specify the criteria for a packet to match against and the policy to apply if the packet matches the criteria:

- `-A FORWARD` append the rule to the FORWARD chain
- `-i eth2` input interface must match `eth2` (the interface the packet came in) when omitted will match any interface
- `-o eth1` output interface must match `eth1` (the interface the packet is going out) when omitted will match any interface
- `-d 10.1.1.70` destination IP address must match the specified address or address range (with subnet mask)
- `-p tcp` the protocol must match TCP
- `--dport 80` the destination port must match 80, only if TCP or UDP is specified as protocol
- `-j ACCEPT` jump to ACCEPT target, the target policy to apply to the matched packet

With this rule, a client can now send a HTTP request to the web server. However, the server response will not pass through the firewall as all traffic from within the DMZ to the outside is still blocked. You can observe this by opening a Wireshark on firewall (node `phoenix`) interface `eth1` and another on the node `Internet` interface `eth0`. Then in the terminal on node `helios` enter:

```
lynx 10.1.1.71
```

Note: You can use `tcpdump` as a command line alternative to Wireshark. To do so in a separate terminal on `phoenix` issue `tcpdump -l -i eth1` and in a terminal on `Internet` issue `tcpdump -l -i eth0` which will print a summary of captured packets in the terminal.

STEP 4. Outgoing Rule

To allow the response from the server to pass through we use the following command (on `phoenix`):

```
iptables -A FORWARD -i eth1 -o eth2 -s 10.1.1.71 -p tcp --sport 80 -j ACCEPT
```

Notice the changes in the input and output interfaces and `-s` instead of `-d` to specify the source IP address that must be matched and the `--sport` instead of `--dport` indicating that the source port must be 80.

Try to visit the web server from `helios` using web server IP address.

Note: Observe that the `iptables` option `-P` is for policy and -p to identify the protocol. If you mistake one for the other, in terminal you will get an error, however in firewall service configuration of a node you will not see the errors.

To check if the rules that you have added to the Firewall service of a node have taken effect, run the emulation and execute the following command on that node:

```
iptables -L
```

If the policy is not changed or a rule does not appear then check the command to see if there are any syntax or option errors.

STEP 5. Other Services – DNS(Name resolution) & SMTP(Email)

Now add necessary rules that allow DNS and SMTP services by the nodes ns-argos and mail-argos respectively. For each service you need to add an incoming and an outgoing rules to the forward chain.

```
For DNS service

iptables -A FORWARD -i eth2 -o eth1 -d 10.1.1.70 -p udp --dport 53 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth2 -s 10.1.1.70 -p udp --sport 53 -j ACCEPT

iptables -A FORWARD -i eth2 -o eth1 -d 10.1.1.72 -p tcp --dport 25 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth2 -s 10.1.1.72 -p tcp --sport 25 -j ACCEPT
```

STEP 6. Internal to DMZ

Write rules that allow traffic from Internal network to reach servers in DMZ: web, mail, and DNS.

```
iptables -A FORWARD -i eth0 -o eth1 -d 10.1.1.70 -p udp --dport 53 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -s 10.1.1.70 -p udp --sport 53 -j ACCEPT

iptables -A FORWARD -i eth0 -o eth1 -d 10.1.1.71 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -s 10.1.1.71 -p tcp --sport 80 -j ACCEPT

iptables -A FORWARD -i eth0 -o eth1 -d 10.1.1.72 -p tcp --dport 25 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -s 10.1.1.72 -p tcp --sport 25 -j ACCEPT
```

STEP 7. Internal to External

Now that we have opened up the DMZ, we configure the firewall to allow the clients in Internal network to access web servers outside. Try to find the correct `iptables` rules that would implement the following:

- Any packet from Internal network is allowed to pass to outside network
- Any traffic from the outside toward the Internal network is only accepted and allowed to pass if it is a response to a request made by a local client.

This can be achieved using only `ESTABLISHED` and `RELATED` packets to pass through in the opposite direction (from outside to Internal). This is the `iptables` method of *Stateful Inspection*. Do a research on this topic to find out how to implement these rules (`man iptables-extensions`).

```
iptables -A FORWARD -i eth0 -o eth2 -s 10.1.1.0/26 -p tcp --dport 80 -m state --
state NEW,RELATED,ESTABLISHED -j ACCEPT

iptables -A FORWARD -i eth2 -o eth0 -d 10.1.1.0/26 -p tcp --sport 80 -m state --
state RELATED,ESTABLISHED -j ACCEPT

Note: Enter each command in one line, that is -- and what follows is the
continuation of the iptables command, all in one line.
```

STEP 8. DMZ to External

Certain services require to connect to external servers as part of their operation. For instance, a DNS server can be configured to perform the *forwarding* operation which is to ask a query on behalf of its clients from another DNS server. In the given configuration the node `ns-argos` will resolve any query for the domain `argos.edu` and forward any other query to the node `Internet`. For simplicity the DNS service is configured on `Internet` node rather than a separate server. Similarly, the node `sphinx` resolves queries for `delos.edu` and forwards any other queries to `Internet` node. The `Internet` node then forwards the relevant queries to appropriate server. As the configuration is isolated the scenario is only simulating the real-life example where the DNS server of an organization forwards its Internet queries to its Internet Service Provider.

Add required rules that allows `ns-argos` to forward queries for other domains to the Internet node. In this case `ns-argos` will be the client and the Internet the DNS server.

```
iptables -A FORWARD -i eth1 -o eth2 -s 10.1.1.70 -d 10.1.1.130 -p udp --dport 53
-m state --state NEW,ESTABLISHED -j ACCEPT

iptables -A FORWARD -i eth2 -o eth1 -d 10.1.1.70 -s 10.1.1.130 -p udp --sport 53
-m state --state ESTABLISHED -j ACCEPT

Note: Enter each command in one line, that is -m and what follows is the
continuation of the iptables command, all in one line.
```

To test the rules, the Internal client `selene` should be able to resolve the name `www.delos.edu`.

This will test the Internal to DMZ rules as well as DMZ to External rules. For this to succeed a DNS query from `selene` must reach `ns-argos`, then a DNS query from `ns-argos` must reach Internet, then the reply from Internet must reach `ns-argos`, and finally the reply from `ns-argos` must reach `selene`. So, in total 4 rules must be correct for this test to succeed.

When an SMTP server needs to send an email to another domain it will be the client and the mail server of the recipient of the email will be the server. This also requires a different pair of rules compared to the ones you have written for the External to DMZ. Add the necessary rules to allow `mail-argos` to send email to any external mail server. Although we cannot test the rule as there is no other mail server in the given scenario.

```
iptables -A FORWARD -i eth1 -o eth2 -s 10.1.1.72 -p tcp --dport 25 -m state --
state NEW,ESTABLISHED -j ACCEPT

iptables -A FORWARD -i eth2 -o eth1 -d 10.1.1.72 -p tcp --sport 25 -m state --
state ESTABLISHED -j ACCEPT

Note: Enter each command in one line, that is -- and what follows is the
continuation of the iptables command, all in one line.
```

STEP 9. Internal to Firewall

Write rules that allow clients in Internal network to connect to SSH service on `phoenix`.

```
iptables -A INPUT -i eth0 -s 10.1.1.0/26 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -o eth0 -d 10.1.1.0/26 -p tcp --sport 22 -j ACCEPT
```

- # ACTIVITY B: VPN

Download the core file **FIT9137_Week-11_Activity_VPN.imn** from Moodle and save it to the shared folder on your host. Open the file in core (from shared folder in VM). The configuration contains a network design with a private network on the right, some public (Internet) routers in the middle, and a client in the top left that wants to connect to the private network. Start the emulation and perform the following tasks:

STEP 1. See if you can reach the devices inside the private network. You can for instance ping the clients or run `lynx` with the address of the server.

yes, we can reach the web server

```
lynx 10.0.6.10
```

STEP 2. Run a `traceroute` between `vpnclient` and `web`. What do you see?

```
traceroute 10.0.6.10
traceroute to 10.0.6.10 (10.0.6.10), 64 hops max
  1   10.0.200.1  0.445ms  0.321ms  0.388ms
  2   10.0.6.10  0.402ms  0.321ms  0.323ms
```

checking local network interfaces we find the `tun0` interface:

```
2: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state\
    UNKNOWN group default qlen 100
    link/none
    inet 10.0.200.4 peer 10.0.200.1/32 scope global tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::10b5:f2b:3bef:247b/64 scope link flags 800
       valid_lft forever preferred_lft forever
```

checking the routing table, we find all traffic is redirected through `tun0` interface:

```
0.0.0.0/1 via 10.0.200.1 dev tun0
default via 10.0.0.1 dev eth0
10.0.0.0/24 dev eth0  proto kernel  scope link  src 10.0.0.20
10.0.1.10 via 10.0.0.1 dev eth0
10.0.6.0/24 via 10.0.200.1 dev tun0
10.0.200.1 dev tun0  proto kernel  scope link  src 10.0.200.4
128.0.0.0/1 via 10.0.200.1 dev tun0
```

STEP 3. Run Wireshark on one of the routers. Then run `lynx` again from `vpnclient` accessing `web` or ping `web`. Can you see the HTTP protocol messages or the ICMP echo request and replies?

No, we only see OpenVPN packets as all traffic is redirected through VPN.

STEP 4. Run Wireshark on the interface `eth1` (with IP `10.0.6.1`) of the `vpnserver`. Run `lynx` as before. What can you see in the captured traffic?

We see the communication between `vpnclient` and `web`. This is because the VPN connection terminates on the external interface of `vpnserver` so the packets are decrypted.

STEP 5. Check the file `client.conf` on `vpnclient` and `server.conf` on `vpnserver` and use `openvpn` manual (`man openvpn`) to learn more about the settings. For instance, what encryption algorithm is used, what key exchange algorithm is used, what are the client and server key pairs etc.

the location of the keys depend on the session as shown below.

`client.conf`

```
client
dev tun
proto udp
remote 10.0.1.10 1194
nobind
ca /tmp/pycore.50682/certs/ca-cert.pem
cert /tmp/pycore.50682/vpnclient.conf/vpnclient.pem
key /tmp/pycore.50682/vpnclient.conf/vpnclient.key
dh /tmp/pycore.50682/certs/dh2048.pem
cipher AES-256-CBC
log /var/log/openvpn-client.log
verb 4
daemon
```

`server.conf`

```
local 10.0.1.10
server 10.0.200.0 255.255.255.0
push redirect-gateway def1
push route 10.0.6.0 255.255.255.0
keepalive 10 120
ca /tmp/pycore.50682/certs/ca-cert.pem
cert /tmp/pycore.50682/vpnserver.conf/vpnserver.pem
key /tmp/pycore.50682/vpnserver.conf/vpnserver.key
dh /tmp/pycore.50682/certs/dh2048.pem
cipher AES-256-CBC
status /var/log/openvpn-status.log
log /var/log/openvpn-server.log
ifconfig-pool-linear
ifconfig-pool-persist /tmp/pycore.50682/vpnserver.conf/ippool.txt
port 1194
proto udp
dev tun
verb 4
daemon
```