IP/ICMP Attacks

Overview of the lab:

The objective of this lab is for students to gain the first-hand experience on various attacks at the IP layer. Some of the attacks may not work anymore, but their underlying techniques are quite generic, and it is important for students to learn these attacking techniques, so when they design or analyze network protocols, they are aware of what attackers can do to protocols. Moreover, due to the complexity of IP fragmentation, spoofing fragmented IP packets is non-trivial. Constructing spoofed IP fragments is a good practice for students to hone their packet spoofing skills, which are essential in network security. We will use Scapy to conduct packet spoofing. This lab covers the following topics:

1. Lab consist of few concepts of IP layer like **IP fragmentation, routing and attacks related to it.**

2. The lab consist of 3 main taks:

    a. **IP Fragmentation**

    b. **ICMP Redirect attack**

    c. **Routing and Reverse Path Filtering**

3. The lab uses pre-built Ubuntu 16.04 VM from seedlab website.

Task 1: IP Fragmentation

1. IP packets of huge size cannot travel in data-link layer. Hence they are cut into fragments of size MTU on sender side and then reassembled on the receiver side.

2. The task 1 consist of 4 sub tasks

    a. Conducting IP fragmentation.

    b. IP fragment with overlapping content

    c. Sending a super large packet

    d. Sending incomplete IP packet.

3. Set-up : Requires two 16.04 Ubuntu VMs

    a. VM1: UDP Client – 10.0.2.15

    b. VM2: UDP Server – 10.0.2.6

Task 1.a: Conducting IP fragmentation

1. In this task we need to spoof IP fragments using scapy.

2. On UDP server – 10.0.2.6 (VM2) run the command “nc -lu 9090”.

3. We need to build multiple IP fragments and send it across UDP server.

4. Open wireshark on both VMs and uncheck the "Reassemble fragmented IPv4 datagrams" option. (Click the following menu sequence: Edit → Preferences; click the Protocols dropdown menu, find and click IPv4. Uncheck the "Reassemble fragmented IPv4 datagrams" option.)

5. Write the python program and run on UDP client. Specify your observations on it.

Task 1.b: IP Fragments with Overlapping Contents

1. Again we need to send fragments to UDP server but this time the contents of fragments need to overlap.

2.The end of the first fragment and the beginning of the second fragment should have K bytes of over-lapping, i.e., the last K bytes of data in the first fragment should have the same offsets as the first K bytes of data in the second fragment. The value of K is decided by students (K should be greater than zero and smaller than the size of either fragment). In the reports, students should indicate what their K values are. Part of fragment one overlap with fragment two. Write the explanation with screenshots.

3. The second fragment is completely enclosed in the first fragment. The size of the second fragment must be smaller than the first fragment (they cannot be equal). The whole fragment overlap in another fragment. Explain the output with screenshots.

4. If second fragment is sent first and then first fragment in scenario 2 and 3 then what are the observations?

Task 1.c: Sending a Super-Large Packet

1. IP packets can have a max of 2^16 payload.

2. If the fragments in total have more payload (>2^16) then explain the observations with screenshots.

Task 1.d: Sending Incomplete IP Packet

1. In this task, we are going to use VM1 to launch a Denial-of-Service attack on VM2 In the attack, VM1 sends a lot of incomplete IP packets to VM2, i.e., these packets consist of IP fragments, but some fragments are missing. All these incomplete IP packets will stay in the kernel, until they time out. Potentially, this can cause the

kernel to commit a lot of kernel memory. In the past, this had resulted in denial-of-service attacks on the server. Write down important observations.

Task 2: ICMP Redirect Attack

1. An ICMP redirect is an error message sent by a router to the sender of an IP packet. Redirects are used when a router believes a packet is being routed incorrectly, and it would like to inform the sender that it should use a different router for the subsequent packets sent to that same destination.

2. Set-up: Requires three 16.04 Ubuntu VMs

a. VM1: Attacker – 10.0.2.15

b. VM2: Victim – 10.0.2.6

c. VM3: Observer – 10.0.2.7

3. Remove the countermeasure: sudo sysctl net.ipv4.conf.all.accept_redirects=1

Code fragment:

```
#!/usr/bin/python3
from scapy.all import *
ip = IP(src = @@@@, dst = @@@@)
icmp = ICMP(type=@@@@, code=@@@@)
icmp.gw = @@@@
# The enclosed IP packet should be the one that
# triggers the redirect message.
ip2 = IP(src = @@@@, dst = @@@@)
send(ip/icmp/ip2/UDP());
```

4. Fill appropriate places in the code and run it to observe wireshark and understand the redirect packets.

5. Run the attack using netwox tool : netwox 86 --device “enp0s3” --filter “src host 10.0.2.6” --gw 10.0.2.15 --spoofip “raw” --code 0 --src-ip 10.0.2.7

7. If remote gateway is used then what is the observation

8. If local gateway which is offline or unavailable then what is its observation.

Task 3: Routing and Reverse Path Filtering

Task 3.a: Network setup

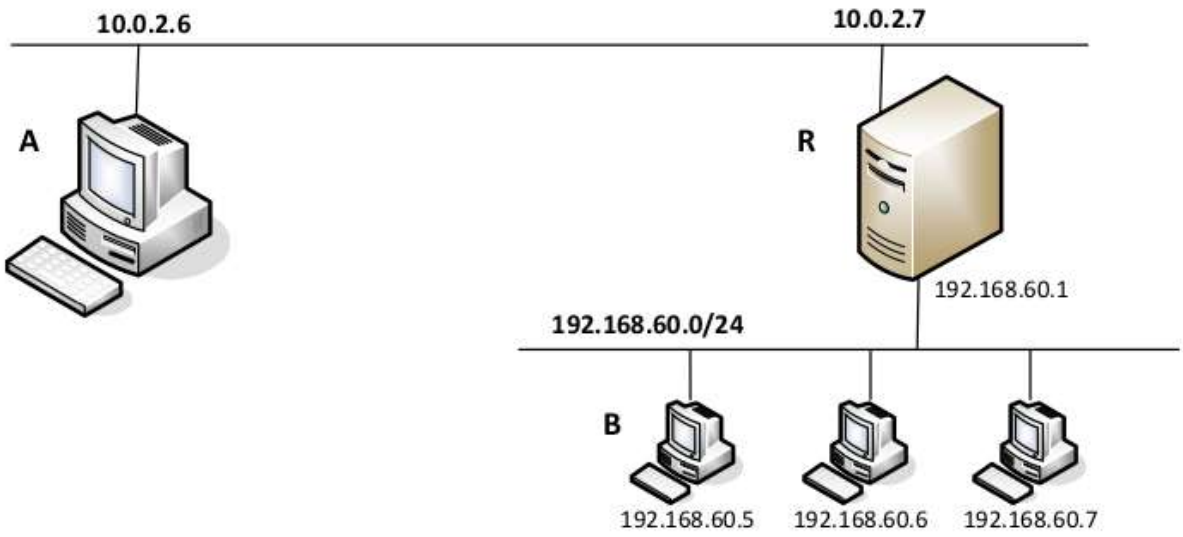

Figure 1: Network setup

VMs attached to the "NAT Network" network will automatically get their IP addresses from the DHCP server, but for VMs on the "Internal Network", VirtualBox does not provide DHCP, so the VM must be statically configured. To do this, click the network icon on the top-right corner of the desktop, and select "Edit Connections". You will see a list of "Wired connections", one for each of the network adaptors used by the VM. For Machine B, there is only one connection, but for Machine R, we will see two. To make sure that you pick the one attached to the "Internal Network" adapter, You can check the MAC address displayed in the pop-up window after you have picked a connection to edit. Compare this MAC address with the one displayed by the ifconfig command, and you will know whether you have picked the right connection or not. After selecting the right connection to edit, pick the "ipv4 Settings" tab and select the "Manual" method, instead of the default "Automatic (DHCP)". Click the "Add" button to set up the new IP address for the VM.

1. Set-up: 3 VMs

    a. VM1 – 10.0.2.6 – (1 interface - Nat Network)

    b. VM2 – 10.0.2.15 (2 interfaces – Nat Network, Internal Network)

    c. VM3 – 192.168.60.5 (1 interface – Internal network)

2. Initially, the VM1 cannot interact with VM3.

Task 3.b: Routing Setup

1. Appropriate routing entries are to be done in VM1, VM2 and VM3 for VM1 to interact with VM3. Add these entries only if not present. First check with ip route.

2. In VM2 add this as it needs to forward packets sudo sysctl net.ipv4.ip_forward=1

3. Write down all routing entries and observation on connection between VM1 and VM3 after adding routing entries. Attach the wireshark screenshots of all three VMs.

Task 3.c: Reverse Path Filtering

1. Linux kernel implements a filtering rule called reverse path filtering, which ensures the symmetric routing rule. When a packet with the source IP address X comes from an interface (say I), the OS will check whether the return packet will return from the same interface, i.e., whether the routing for packets going to X is symmetric.

2. Students should send three spoofed packets on Machine A (VM1). All these packets should be sent to Machine B (VM3), but the source IP addresses should use one of the following:

a. An IP address belonging to the network 10.0.2.0/24.

b. An IP address belonging to the internal network 192.168.60.0/24.

c. An IP address belonging to the Internet, such as 1.2.3.4.

3. Attack screenshots of wireshark of all three VMs and explain the observations individually