Network Basics

1 Overview

This exercise explores basic network concepts in a Linux environment. These include the ARP protocol, the use of ping and a brief introduction to TCP/IP. The tcpdump utility is used to view network traffic.

This exercise, (and manual), is not intended to replace instruction or independent reading on networking protocols. The exercise is intended to provide students with an environment with which they can observe traffic generated by basic network operations.

This lab and its prerequisite knowlege provide background for other Labtainer networking exercises including a lab on ARP spoofing.

2 Lab Environment

This lab runs in the Labtainer framework, available at http://nps.edu/web/c3o/labtainers. That site includes links to a pre-built virtual machine that has Labtainers installed, however Labtainers can be run on any Linux host that supports Docker containers.

From your labtainer-student directory start the lab using:

```
labtainer network-basics
```

A link to this lab manual will be displayed.

3 Network Configuration

This lab includes two networked computers: box1 and box2. The two computers are connected via a virtual network that you can treat as a single Ethernet cable connecting the two computers.

When the lab starts, you will see two virtual terminals, one connected to each computer.

This lab is designed to avoid use of name servers for the local computers, the intention is to focus on IP addresses and network interfaces.

```
box1 <===> box2
      LAN
```

4 Lab Tasks

4.1 Explore

Use this command:

```
ip addr
```

on the two computers to familiarize yourself with the network interfaces. Our focus is on the 2nd entry in each display from `ip addr`, i.e., the `eth0` entries. The `link/ether` value, e.g., the value such as `02:42:ac:00:00:03` is the network interface MAC address that appears in Ethernet packet headers. Often, MAC addresses are tied to physical hardware, e.g., an Ethernet interface. In this lab, the network is virtual, as are the MAC addresses.

The other address of interest is labeled `inet`, e.g., `172.0.0.3/24`. That is the IPv4 address that appears in IP packet headers. In this example, the `/24` tells us the *subnet* associated with this interface is `172.0.0`, and the address of the computer on that subnet is `3`. The value following the slash tells you how many bits of the 32 bit IP address is allocated to name the subnet. The remaining bits are used to name the device on the subnet.

The IP addresses described above allow computers to name each other, and thus communicate. However, at the lowest layer, i.e., the *link layer* that talks directly with the physical or virutal media, MAC addresses are used to communicate. Therefore there must be some way to translate IP addresses to MAC addresses.

4.2 ARP

The Address Resolution Protocol (ARP) is used to map IP addresses to MAC addresses. On box2, use the

```
arp -a
```

command to view the current mapping. Note that nothing displays because the ARP table is empty. When our two computers first start up, they do not each other's MAC addresses.

On box1, start the tcpdump program so that you can observe network traffic:

```
sudo tcpdump -vv -n -e -i eth0
```

These options to tcpdump are:

- `-vv` – Provide verbose output
- `-n` – Do not perform reverse DNS lookup, just show the IP addresses.
- `-e` Show Ethernet MAC addresses.
- -i eth0 Show traffic on interface eth0.

You may notice some traffic being displayed that is not related to actions you take, e.g., "router solicitation" packets or flow packets. Ignore those. On box2, use the ping command to ping box1:

```
ping 172.0.0.2 -c 2
```

Observe the traffic in tcpdump. You should see the ARP request from box2 asking for the MAC address of whoever handles traffic to the box1 IP address. Note that the destination MAC address in the ARP request is `ff:ff...`, which is a broadcast message seen by every Ethernet interface on the LAN. And you'll see the ARP response from box1. Once the two boxes associate IP addresses with MAC addresses, they can exchange network layer packets. In this case, you will see ICMP packets. The first packet is an `ICMP echo request` from box2 to box1. This is a "ping". The next packet is the `ICMP echo reply` from box1.

Use `ctrl-c` to break out of the tcpdump on box1. The use the `arp -a` command on both boxes to view the current content of the ARP table. Those ARP entries allow the two boxes to address each other without repeating the ARP request/response.

4.2.1 Trusting ARP responses

The use of ARP to map IP addresses to MAC addresses requires some level of trust. When a box asks, "to which MAC address should I send packets for this IP address?", it is trusting that only the proper box will respond. See the `arp-spoof` lab for an example of how this trust can be misplaced.

4.2.2 Communicating beyond the subnet

You have observed how ARP allows computers on a common subnet to communicate using IP addresses mapped to MAC addresses. But what about computers that do not share a subnet? That type of communication requires the use of packet forwarding and routing, e.g., via a gateway component. See the `routing-basics` lab for examples of communication between computers that do not share a subnet.

4.3 TCP

In this section we'll briefly look at some IP packets within a TCP session, specifically the "three way handshake". Restart the tcpdump on box1, this time without the `-e` switch:

```
sudo tcpdump -vv -n -i eth0
```

Then, on box2, initiate a ssh session to box1. We won't actually complete the login, we simply wish to look at the start of the session:

```
ssh 172.0.0.2
```

This results in IP packets that include `proto TCP`, which tells us that the TCP protocol is being used. The address information in the first packet, e.g.,

```
172.0.0.3.34104 > 172.0.0.2.22
```

tells us the packet is sent from box2. The final number in the box2 address, in this case `34014` is an *ephemeral* TCP port number that will be used for the session. The final number in the box1 address is 22, which is the port number used by the ssh protocol. The *Flags* values in brackets on the first packet is `[S]`, which indicates it is a *SYN* packet, i.e., the first of the three handshake packets that initiate a TCP session. The 2nd packet is from box1 to box2. Note the port numbers. The Flags on the second packet are `[S.]`, which indicates a *SYN-ACK* packet, i.e., the 2nd packet in the handshake. The third packet completes the handshake with an `ACK` sent from box2 to box1. Note the Flags field is just a period, which reflects no flags. The

```
seq 1, ack 1,
```

in that 3rd packet reflects that ack and that the TCP session sequence number is starting at 1.

That completes the handshake and the subsequent packets are part of a reliable session in which the TCP protocol ensures that packets are not lost and are arranged in the proper order for delivery to the applications, which in this case is a SSH client and SSH server. The notion of packets getting lost or out of order may seem unlikely when considering our simple two-box example. But consider computers on opposite sides of the world exchanging information using router paths that may vary within the duration of the TCP session.

5 Submission

After finishing the lab, go to the terminal on your Linux system that was used to start the lab and type:

```
stoplab
```

When you stop the lab, the system will display a path to the zipped lab results on your Linux system. Provide that file to your instructor, e.g., via the Sakai site.

This lab was developed for the Labtainer framework by the Naval Postgraduate School, Center for Cybersecurity and Cyber Operations under National Science Foundation Award No. 1438893. This work is in the public domain, and cannot be copyrighted.