2

CAPTURING TRAFFIC WITH ARP SPOOFING

Pay no attention to the man behind the curtain!
–Noel Langley, *The Wizard of Oz*

Anyone who walks into a coffee shop and connects to its Wi-Fi network can intercept and view other users' unencrypted web traffic using a technique called *ARP spoofing*, which exploits a vulnerability in the design of the address resolution protocol (ARP). In this chapter, we explain how ARP works, describe the steps of an ARP spoofing attack, and then perform one ourselves.

How the Internet Transmits Data

Before we can discuss ARP spoofing, we must first understand the internet's general structure. This section describes how the internet transmits data through a hierarchical network using packets, MAC addresses, and IP addresses.

Packets

All information on the internet is transmitted in *packets*. You can think of a packet as an envelope that contains the data that you want to send. As with the postal service, these packets are routed to their destinations based on a specified address. Figure 2-1 shows some parallels between envelopes and packets.

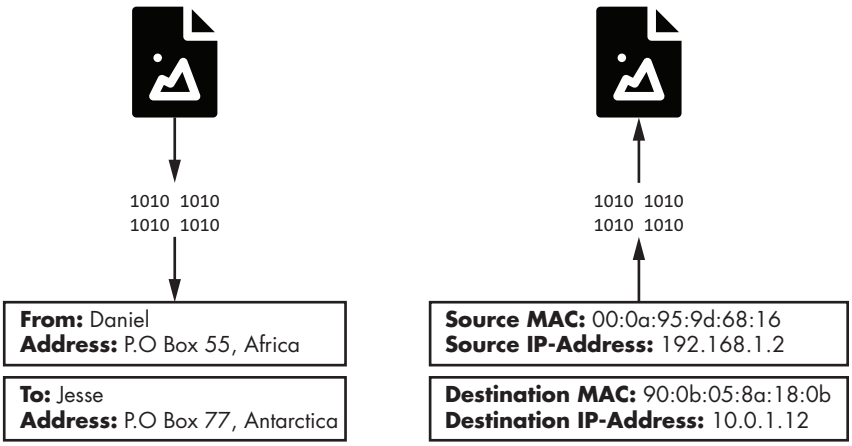

Figure 2-1: Parallels between envelopes and packets

The From Address section on an envelope contains two critical pieces of information: 1) the name of the person sending the letter, and 2) where they live. Similarly, packets have a source (*media access control [MAC] address*) that represents the machine sending the packet and a source (*IP address*) that represents where the packet came from. Other similar fields, known as *packet headers*, represent the packet's destination.

The internet uses devices called *routers* to sort and forward packets. Packets make their way through the internet, traveling from router to router like mail travels from post office to post office.

MAC Addresses

Your laptop contains a *network interface card (NIC)* that allows it to connect to Wi-Fi routers. This card has a unique address, called a MAC address, that identifies your machine on the network. When the router wants to send your computer information, it labels that packet with your laptop's MAC address and then broadcasts it as a radio signal. All machines connected to that router receive this radio signal and check the packet's MAC address to see whether the packet is intended for them. MAC addresses are normally 48-bit numbers written in hexadecimal (for example, 08:00:27:3b:8f:ed).

IP Addresses

You probably already know that IP addresses also identify machines on a network. So why do we need both IP and MAC addresses? Well, networks

consist of hierarchical regions similarly to how some countries are split into states, which themselves contain cities. IP addresses follow a structure that allows them to identify a device's place in the larger network. If you moved to another coffee shop, your laptop would be assigned a new IP address to reflect its new location; however, your MAC address would remain the same.

An IPv4 address encodes the network hierarchy information in a 32-bit number. This number is typically represented in four sections separated by dots (such as 192.168.3.1). Each section represents an 8-bit binary number. For example, the 3 in 192.168.3.1 actually represents the 8-bit binary number 00000011.

IP addresses in the same region of the hierarchy also share the same upper-level bits. For example, all machines on the University of Virginia campus have IPv4 addresses like 128.143.xxx.xxx. You'll also see this written in Classless inter-domain routing (CIDR) notation as 128.143.1.1/16, indicating that machines share the same 16 upper bits, or the first two numbers. Because IP addresses follow a particular structure, routers can use parts of the IP address to decide how to route a packet through the hierarchy. Figure 2-2 shows a simplified example of this hierarchy of routers.

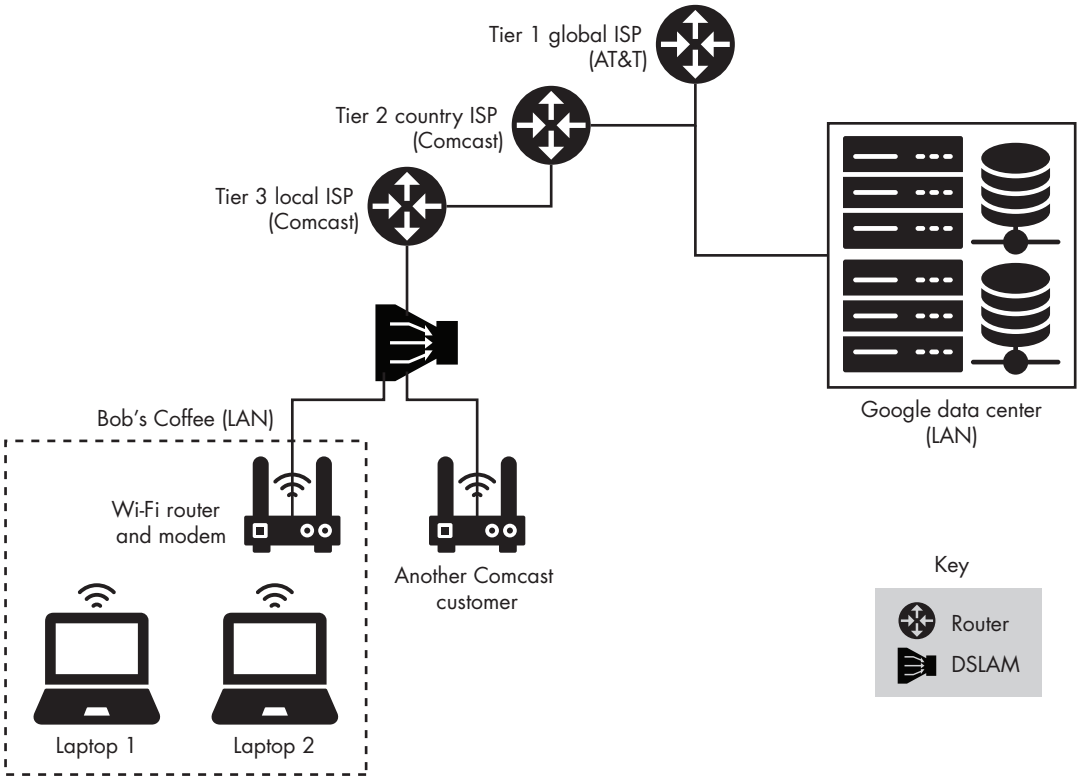

Figure 2-2: A simplified view of the network hierarchy

Figure 2-2 also shows a *digital subscriber line access multiplexer (DSLAM)*. A DSLAM allows signals associated with internet traffic to be sent over wires originally intended for cable television. The DSLAM distinguishes between internet and television signals, which is why you can connect both your television and router to the same cable socket.

Let's use the coffee shop example to follow a packet through the network hierarchy. Imagine you're in a coffee shop in San Francisco and access the following web page: *http://www.cs.virginia.edu*. This web page is hosted on a web server with the IP address 128.143.67.11. On the first leg of its journey, the web request passes through your laptop's NIC, which then sends it to the Wi-Fi router in the coffee shop. The router then sends the web request to the DSLAM, which forwards the request to a router owned by an *internet service provider (ISP)*, like Comcast. The Comcast routers then compare the IP address to a list of prefixes until it finds a match. For example, it might find a match for the prefix 128.xxx.xxx.xxx, indicating its connection to that section of the hierarchy. As the request is sent through the hierarchy, the matches will become more specific. For example, the address will need to match 128.143.xxx.xxx, then 128.143.67.xxx. Once the packet reaches the lowest level of the hierarchy, where there are no more routers, the router uses the MAC address in the packet to determine the request's final destination. We refer to the lowest level of the hierarchy as a *local area network (LAN)* because all of the machines in that level are connected through a single router.

Now that we have a general overview of the structure of the internet, we can discuss attacks that take place at the lowest level of the hierarchy.

ARP Tables

We've established that after a packet has reached its designated LAN, the network uses the packet's MAC address to determine its final destination. But how does the router know the MAC address of the machine with the IP address 128.143.67.11? This is where ARP is useful. Following ARP, the router sends a message called an *ARP query* to all machines on the network, asking the machine with the IP address 128.143.67.11 to reply with an *ARP response* containing its MAC address. The router will then store this mapping between the IP address and MAC in a special table, called an *ARP table*. By storing this information in the ARP table, the router reduces the need to issue ARP queries in the near future.

THE QUICK VERSION

MAC addresses identify who you are, IP addresses identify where you are, and ARP tables manage the mapping between who you are and where you are on the network. In an ARP spoofing attack, we pretend to be someone else.

ARP Spoofing Attacks

An ARP spoofing attack consists of two phases. During the first phase, the attacker sends a fake ARP response to the victim, stating that the attacker's MAC address maps to the router's IP address. This allows the attacker to trick the victim into believing that the attacker's machine is the router. During the second phase, the victim accepts the fake ARP packet sent by the attacker and updates the mapping in its ARP table to reflect that the attacker's MAC address now maps to the router's IP address. This means that the victim's internet traffic will be sent to the attacker's machine instead of the router. The attacker's machine can then forward this information to the router after inspecting it.

If the attacker also wants to intercept internet traffic intended for the victim, the attacker must also trick the router into sending it the victim's traffic. Therefore, the attacker must create a fake ARP packet indicating that the victim's IP address maps to the attacker's MAC address. This allows the attacker to intercept and inspect incoming internet traffic and then forward that traffic to the victim.

We can explain the ideas behind an ARP spoofing attack with a simple diagram, shown in Figure 2-3. Here, Jane (the attacker) tricks Alice (the victim) into sending her mail to Jane.

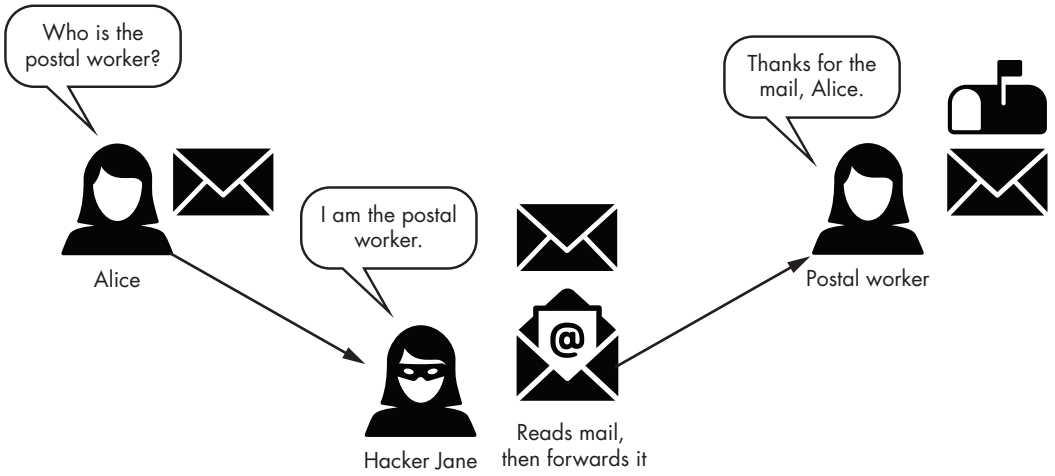

Figure 2-3: An example of a spoofing attack involving a postal worker

The ARP spoofing attack is an example of a *man-in-the-middle* attack, because the attacker places themselves between the victim and router.

Performing an ARP Spoofing Attack

Let's perform an ARP spoofing attack. First, you must ensure that you've started the pfSense, Kali, and Metasploitable virtual machines before

beginning this attack. Visit Chapter 1 for instructions on doing so. Now let's install the tools that we'll need to perform the ARP spoofing attack. Open a terminal on the Kali Linux virtual machine and install the `dsniff` tool. The default password for the Kali Linux virtual machine is "kali". Start by running **`sudo -i`** to become a root user. You will also need to update the `apt-get` package manager by running **`apt-get update`**.

```
kali@kali:~$ sudo -i
kali@kali:~$ apt-get update
kali@kali:~$ apt-get install dsniff
```

The `dsniff` tool contains several useful tools for intercepting network traffic, such as `arpspoof`, a tool that executes an ARP spoofing attack.

We must discover the IP addresses of the other machines on the network to *spoof* them (that is, pretend to be them). Run the `netdiscover` tool using the following command:

```
kali@kali:~$ sudo netdiscover
```

The `netdiscover` works by scanning the network using ARP queries. It issues ARP queries for all possible IP addresses on the subnetwork, and when a machine on the network responds, it records and displays the machine's MAC address and IP address. The `netdiscover` tool also infers the NIC manufacturer from the MAC address. Because all MAC addresses must be unique, a central board at the Institute of Electrical and Electronics Engineers (IEEE) issues manufacturers a range of MAC addresses in order to ensure uniqueness.

Your scan should detect two machines on the network and generate the output shown here:

```
IP              At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.100.1   08:00:27:3b:8f:ed      1      60  PCS Systemtechnik GmbH
192.168.100.101 08:00:27:fe:31:e6      1      60  PCS Systemtechnik GmbH
```

The actual IP addresses returned will vary depending on your setup. The machine with the lowest IP address is normally the router on the LAN. We'll refer to this IP address as <*ROUTER_IP*> for the rest of this chapter. The second IP address belongs to the Metasploitable virtual machine (our victim), which we'll refer to as <*VICTIM_IP*>. Once you've discovered both machines, end the scan by pressing CTRL-C.

Next, you will need to allow the Kali Linux machine to forward packets on behalf of other machines by enabling IP forwarding. Make sure that you're a root user on Kali Linux, and then enable IP forwarding by setting the IP forwarding flag:

```
kali@kali:~$ echo 1 > /proc/sys/net/ipv4/ip_forward
```

Now that you've enabled IP forwarding, you'll need to trick the victim into believing you're the router. Do this by issuing fake ARP replies stating that your MAC address maps to the router's IP address. Figure 2-4 shows an example of this step in the attack.

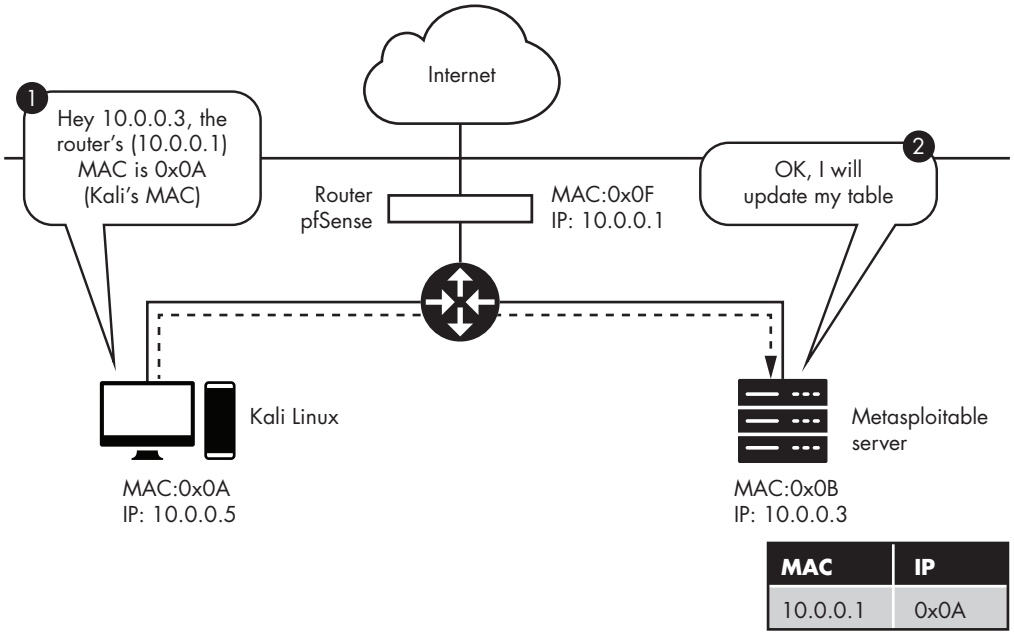

Figure 2-4: The first stage of an ARP spoofing attack

You can generate multiple fake ARP replies by running the following command:

```
arpspoof -i eth0 -t <VICTIM_IP> <ROUTER_IP>
```

The `-t` flag specifies the target, and the `-i` flag represents the interface. Your NIC supports several ways of connecting to the network. For example, `wlan` represents a wireless LAN (Wi-Fi connection), and `eth0` represents an Ethernet connection. In this virtual lab environment, the machines are virtually connected by Ethernet, so you'll use `eth0` for your interface. In the coffee shop environment, the interface would be set to `wlan`.

The following snippet shows the result of running `arpspoof`. You'll need to generate multiple fake ARP replies to ensure that the table is always updated with the incorrect information. The tool will generate multiple packets for you, so you need to run it only once.

```
kali@kali:~$ sudo arpspoof -i eth0 -t 192.168.100.101 192.168.100.1
[sudo] password for kali:
8:0:27:1f:30:76 8:0:27:fe:31:e6 0806 42: arp reply 192.168.100.1 is-at 8:0:27:1f:30:76 ❶
8:0:27:1f:30:76 8:0:27:fe:31:e6 0806 42: arp reply 192.168.100.1 is-at 8:0:27:1f:30:76
```

Let's examine the command's output, paying particular attention to the first line ❶. This line represents a summary of the information in the packet that was just sent. The summary is composed of five key parts:

1. `8:0:27:1f:30:76` is the MAC address of the Kali Linux machine (attacker), which created the packet.
2. `8:0:27:fe:31:e6` is the MAC address of the machine (victim) that will receive the packet.
3. `0806` is a type field indicating that an ARP packet is contained within the Ethernet frame being transmitted.
4. `42` represents the total number of bytes associated with the Ethernet frame.
5. The remaining section, `arp reply 192.168.100.1 is-at 8:0:27:1f:30:76`, is a summary of the ARP reply that falsely states that the router's IP address (`192.168.100.1`) is associated with the Kali Linux machine's MAC address (`8:0:27:1f:30:76`).

You must also trick the router into believing you're the victim so that you can intercept incoming internet traffic on the victim's behalf. Open a new terminal and run the command that follows. Notice that <*ROUTER_IP*> and <*VICTIM_IP*> are now reversed. This is because you're now generating packets to trick the router into believing you're the victim:

```
kali@kali:~$ arpspoof -i eth0 -t <ROUTER_IP> <VICTIM_IP>
```

Now that you've spoofed the victim and router, what can you do with the intercepted packets? Let's inspect the packets we've intercepted and extract URLs from them. This will allow us to generate a list of websites that the victim visits. Extract the URLs by running the following command in a new terminal:

```
kali@kali:~$ urlsnarf -i eth0
```

You can also generate some internet traffic on the victim machine. Log in to the Metasploitable virtual machine using **msfadmin** for both the username and password, and then enter the following command to generate a web request to *google.com*:

```
msfadmin@metasploitable:~$ wget http://www.google.com
```

Figure 2-5 shows an overview of what's occurring during this step.

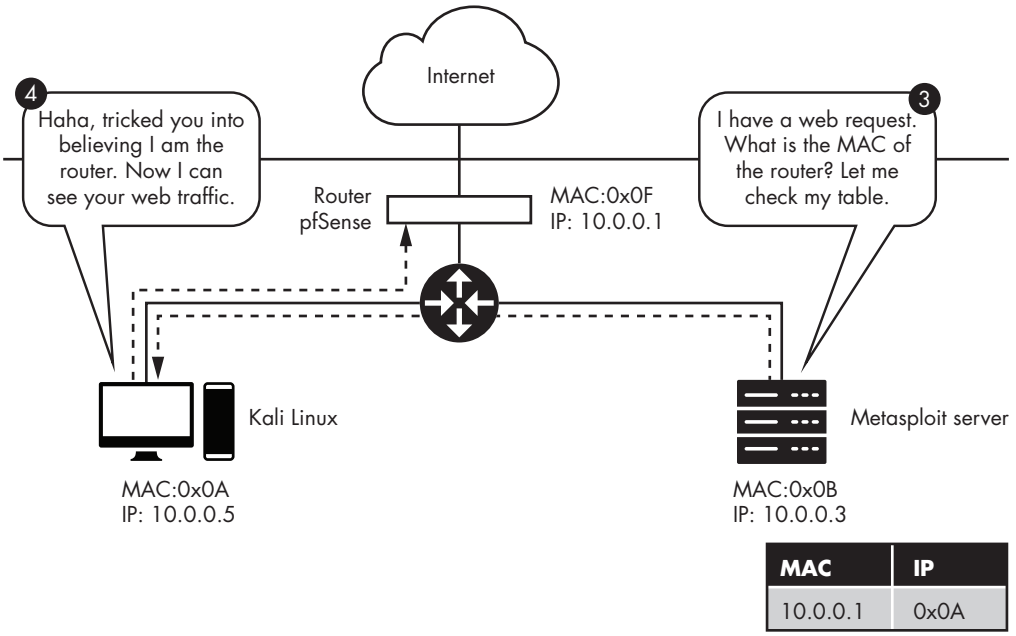

Figure 2-5: The second stage of the ARP spoofing attack, in which the victim uses the corrupted ARP table to address packets

If you've done everything correctly, the URL associated with the web request will show up in the terminal after a couple of minutes. Be patient; it takes time to parse the packets:

```
kali@kali:~$ sudo urlsnarf -i eth0
urlsnarf: listening on eth0 [tcp port 80 or port 8080 or port 3128]
192.168.100.101 - - "GET http://www.google.com/ HTTP/1.0"
```

Take a look at this output. Although we're showing only the URL here, the attacking machine is capturing all of the packets the victim sends and receives from the internet. This means that the attacker can see any unencrypted information the victim sends over the network. It also means an attacker can modify packets to inject malicious code on the machine.

Once you're done performing your malicious actions, don't leave the ARP tables in the corrupted state. After the attacker leaves the coffee shop, the victim will no longer be able to connect to the internet, and they'll suspect foul play. You must restore the ARP tables to their original configurations before shutting down the attack. Thankfully, `arpspoof` does this for us. Shut down the attack by pressing CTRL-C in both terminals running `arpspoof`.

PROTECTING YOURSELF AGAINST ARP SPOOFING

Although it's difficult to prevent an ARP spoofing attack, encrypting your internet traffic helps protect your information from being stolen or modified. Any traffic sent over an HTTPS connection is encrypted. However, manually checking to ensure that every URL you visit uses HTTPS is tedious, so the Electronic Frontier Foundation (*eff.org*) has created a web browser extension (for Chrome, Edge, Firefox, and Opera) called HTTPS Everywhere that ensures that all your web traffic goes over an HTTPS connection. Installing this plug-in is a great way to protect yourself.

Detecting an ARP Spoofing Attack

In this section, we'll write a Python program to heuristically detect an ARP spoofing attack. We'll build our own ARP table using a dictionary and then check to see whether the packet we receive has changed an entry. We'll assume that any packet that changes the state of our table is malicious.

We'll begin by selecting a library that can both intercept and parse the packets that pass through our NIC. Scapy is a popular Python package that allows us to read and send packets. Before you can use Scapy, you'll need to install it with `pip3`. Use the following commands to get both `pip3` and Scapy:

```
kali@kali:~$ sudo apt-get install python3-pip
kali@kali:~$ pip3 install --pre scapy[basic]
```

Once you've installed Scapy, you can import the `sniff` library, which allows us to capture and inspect the packets that pass through our NIC. Copy and paste the following Python program (*arpDetector.py*) into Mousepad or the code editor of your choice. To start Mousepad, run **`mousepad &`**.

```
from scapy.all import sniff
IP_MAC_Map = {}

def processPacket(packet):
    src_IP = packet['ARP'].psrc
    src_MAC = packet['Ether'].src
    if src_MAC in IP_MAC_Map.keys():
        if IP_MAC_Map[src_MAC] != src_IP :
            try:
                old_IP =  IP_MAC_Map[src_MAC]
            except:
                old_IP = "unknown"
            message = ("\n Possible ARP attack detected \n "
                       + "It is possible that the machine with IP address \n "
                       + str(old_IP) + " is pretending to be " + str(src_IP)
                        +"\n ")
            return message
```

```
    else:
        IP_MAC_Map[src_MAC] = src_IP

❶ sniff(count=0, filter="arp", store = 0, prn = processPacket)
```

The `sniff()` function ❶ in the Scapy library takes several optional parameters. In this implementation, we use the `count` parameter to indicate the number of packets to sniff. A count value of 0 means that the library should continuously sniff packets. We also use the `filter` parameter, which specifies the type of packet to capture. Because we're interested in only ARP packets, we specify a filter value of `"arp"`. The `store` parameter indicates the number of packets to store. We set the parameter to 0 because we don't want to waste memory by storing packets. Lastly, the `prn` parameter is a functional pointer that points to the function called whenever a packet is received. It takes a single parameter, which represents the received packet, as input.

```
kali@kali:~$ sudo python3 arpDetector.py
```

As the program is running, open another Kali terminal and execute an ARP spoofing attack by running the following command:

```
kali@kali:~$ arpspoof -i eth0 -t  <ROUTER_IP>  <VICTIM_IP>
```

Then, quit the attack by pressing CTRL-C. This will cause `arpspoof` to issue packets that restore the ARP table. When your Python program detects these packets, you'll see a message like the following:

```
Possible ARP attack detected
It is possible that the machine with IP address
192.168.0.67 is pretending to be 192.168.48.67
```

Exercises

Deepen your understanding of ARP spoofing and forwarding by attempting the following exercises, listed in order of increasing difficulty. The first exercise requires running only a single command, but the second is more challenging because it requires you to write a Python program and deepen your understanding of the Scapy library. The final exercise prompts you to apply the fundamentals you learned in this chapter to a new attack.

Inspect ARP Tables

Inspect the ARP tables on the Metasploitable virtual machine by running this command:

```
msfadmin@metasploitable:~$ sudo arp -a
```

Compare the state of the ARP tables on the Metasploitable server before and after the ARP spoofing attack. Do you notice any differences? If so, which entries have changed?

Implement an ARP Spoofer in Python

In this chapter, we discussed how to execute an ARP spoofing attack. For this exercise, you'll write a Python program that allows you to perform an ARP spoofing attack with a single command, shown here:

```
kali@kali:~$ sudo python3 arpSpoof.py <VICTIM_IP>  <ROUTER_IP>
```

To do this, you'll need to write a program that performs the steps discussed in this chapter. Your program should generate spoofed ARP packets and send them to both the victim and router. Once the attack is complete, your program should restore the ARP tables to their original state. Write your program (*arpSpoof.py*) in Python, and use the Scapy library to construct and send the packets. We've included skeleton code here:

```
from scapy.all import *
import sys

❶ def arp_spoof(dest_ip, dest_mac, source_ip):
      pass

❷ def arp_restore(dest_ip, dest_mac, source_ip, source_mac):
      packet= ❸ARP(op="is-at", hwsrc=source_mac,
                      psrc= source_ip, hwdst= dest_mac , pdst= dest_ip)
    ❹ send(packet, verbose=False)

  def main():
      victim_ip= sys.argv[1]
      router_ip= sys.argv[2]
      victim_mac = getmacbyip(victim_ip)
      router_mac = getmacbyip(router_ip)

      try:
          print("Sending spoofed ARP packets")
          while True:
              arp_spoof(victim_ip, victim_mac, router_ip)
              arp_spoof(router_ip, router_mac, victim_ip)
      except KeyboardInterrupt:
          print("Restoring ARP Tables")
          arp_restore(router_ip, router_mac, victim_ip, victim_mac)
          arp_restore(victim_ip, victim_mac, router_ip, router_mac)
          quit()
```

Implement the `arp_spoof()` function ❶. This function should be very similar to `arp_restore()` ❷, which restores the ARP tables to their original state.

You can use `arp_restore()` as a guide. Within that function, we create a new ARP packet. The `ARP()` function ❸ takes several options (`op`). The `"is-at"` option represents an ARP reply, and the `"who-has"` option represents an ARP request. You might also see these options listed as the numbers 2 and 1, respectively. Finally, we send the packet we created ❹.

MAC Flooding

Content addressable memory (CAM) is the memory hardware used in both routers and switches. In switches, these memories map MAC addresses to the corresponding ports. Thus, CAM can store only a limited number of entries. If the switch's CAM is full, it will broadcast a message on all ports. Attackers can force this behavior by sending the switch packets with random MAC addresses. Write a Scapy program that performs this attack.