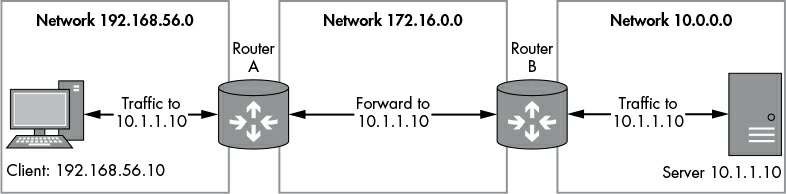
## ADVANCED APPLICATION TRAFFIC CAPTURE

Usually, the network traffic-capturing techniques you learned in [Chapter 2](#_bookmark35) should suffice, but occasionally you’ll encounter tricky situations that require more advanced ways to capture network traffic. Sometimes, the challenge is an embedded platform that can only be configured with the Dynamic Host Configuration Protocol (DHCP); other times, there may be a network that offers you little control unless you’re directly connected to it.

### Rerouting Traffic

IP is a *routed* protocol; that is, none of the nodes on the network need to know the exact location of any other nodes. Instead, when one node wants to send traffic to another node that it isn’t directly connected to, it sends the traffic to a *gateway* node, which forwards the traffic to the destination. A gateway is also commonly called a *router*, a device that routes traffic from one location to another.



#### Using Traceroute

When tracing a route, you attempt to map the route that the IP traffic will take to a particular destination. Most operating systems have built-in tools to perform a trace, such as traceroute on most Unix-like platforms and tracert on Windows.

[Listing 4-1](#_bookmark180) shows the result of tracing the route to [*www.google.com*](http://www.google.com/)from a home internet connection.

C:\Users\user>tracert [www.google.com](http://www.google.com/)

Tracing route to [www.google.com](http://www.google.com/) [173.194.34.176] over a maximum of 30 hops:

1 2 ms 2 ms 2 ms home.local [192.168.1.254]

2 15 ms 15 ms 15 ms 217.32.146.64

3 88 ms 15 ms 15 ms 217.32.146.110

4 16 ms 16 ms 15 ms 217.32.147.194

5 26 ms 15 ms 15 ms 217.41.168.79

6 16 ms 26 ms 16 ms 217.41.168.107

7 26 ms 15 ms 15 ms 109.159.249.94

8 18 ms 16 ms 15 ms 109.159.249.17

9 17 ms 28 ms 16 ms 62.6.201.173

10 17 ms 16 ms 16 ms 195.99.126.105

11 17 ms 17 ms 16 ms 209.85.252.188

12 17 ms 17 ms 17 ms 209.85.253.175

13 27 ms 17 ms 17 ms lhr14s22-in-f16.1e100.net [173.194.34.176]

Each numbered line of output (1, 2, and so on) represents a unique gateway routing traffic to the ultimate destination. The output refers to a maximum number of *hops*. A single hop represents the network between each gateway in the entire route. For example, there’s a hop between your machine and the first router, another between that router and the next, and hops all the way to the final destination. If the maximum hop count is exceeded, the traceroute process will stop probing for more routers. The maximum hop can be specified to the trace route tool command line; specify -h NUM on Windows and -m NUM on Unix-style systems.(The output also shows the round-trip time from the machine performing the traceroute and the discovered node.)

#### Routing Tables

The OS uses *routing tables* to figure out which gateways to send traffic to. A routing table contains a list of destination networks and the gateway to route traffic to. If a network is directly connected to the node sending the network traffic, no gateway is required, and the network traffic can be transmitted directly on the local network.

You can view your computer’s routing table by entering the command netstat -r on most Unix-like systems or route print on Windows. [Listing 4-2](#_bookmark183) shows the output from Windows when you execute this command.

As mentioned earlier, one reason routing is used is so that nodes don’t need to know the location of all other nodes on the network. But what happens to traffic when the gateway responsible for communicating with the destination network isn’t known? In that case, it’s common for the routing table to forward all unknown traffic to a *default gateway*. You can see the default gateway at ➊, where the network destination is 0.0.0.0. This destination is a placeholder for the default gateway, which simplifies the management of

the routing table. By using a placeholder, the table doesn’t need to be changed if the network configuration changes, such as through a DHCP configuration. Traffic sent to any destination that has no known matching route will be sent to the gateway registered for the 0.0.0.0 placeholder address.

### Configuring a Router

By default, most operating systems do not route traffic directly between network interfaces. This is mainly to prevent someone on one side of the route from communicating directly with the network addresses on the other side. If routing is not enabled in the OS configuration, any traffic sent to one of the machine’s network

interfaces that needs to be routed is instead dropped or an error message is sent to the sender. The default configuration is very important for security: imagine the implications if the router controlling your connection to the internet routed traffic from the internet directly to your private network.

#### Enabling Routing on Windows

By default, Windows does not enable routing between network interfaces. To enable routing on Windows, you need to modify the system registry. You can do this by using a GUI registry editor, but the easiest way is to run the following command as an administrator from the command prompt:

C> **reg add HKLM\System\CurrentControlSet\Services\Tcpip\Parameters ^**

**/v IPEnableRouter /t REG\_DWORD /d 1**

To turn off routing after you’ve finished capturing traffic, enter the following command:

C> **reg add HKLM\System\CurrentControlSet\Services\Tcpip\Parameters ^**

**/v IPEnableRouter /t REG\_DWORD /d 0**

You’ll also need to reboot between command changes.

#### Enabling Routing on \*nix

To enable routing on Unix-like operating systems, you simply change the IP routing system setting using the sysctl command. (Note that the instructions for doing so aren’t necessarily consistent between systems, but you should be able to easily find specific instructions.)

To enable routing on Linux for IPv4, enter the following command as root (no need to reboot; the change is immediate):

# **sysctl net.ipv4.conf.all.forwarding=1**

To enable IPv6 routing on Linux, enter this:

# **sysctl net.ipv6.conf.all.forwarding=1**

You can revert the routing configuration by changing 1 to 0 in the previous commands. To enable routing on macOS, enter the following:

* **sysctl -w net.inet.ip.forwarding=1**

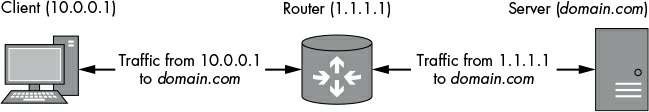
### Network Address Translation

When trying to capture traffic, you may find that you can capture outbound traffic but not returning traffic. The reason is that an upstream router doesn’t know the route to the original source network; therefore, it either drops the traffic entirely or forwards it to an unrelated network. You can mitigate this situation by using *Network Address Translation (NAT)*, a technique that modifies the source and destination address information of IP and higher-layer protocols, such as TCP. NAT is used extensively to extend the limited IPv4 address space by hiding multiple devices behind a single public IP address.

NAT can make network configuration and security easier, too. When NAT is turned on, you can run as many devices behind a single NAT IP address as you like and manage only that public IP address.

#### Enabling SNAT

When you want a router to hide multiple machines behind a single IP address, you use SNAT. When SNAT is turned on, as traffic is routed across the external network interface, the source IP address in the packets is rewritten to match the single IP address made available by SNAT.



#### Configuring SNAT on Linux

Although you can configure SNAT on Windows and macOS using Internet Connection Sharing, I’ll only provide details on how to configure SNAT on Linux because it’s the easiest platform to describe and the most flexible when it comes to network configuration.

Before configuring SNAT, you need to do the following:

* Enable IP routing as described earlier in this chapter.
* Find the name of the outbound network interface on which you want to configure SNAT. You can do so by using the ifconfig command. The outbound interface might be named something like eth0.
* Note the IP address associated with the outbound interface when you use ifconfig.

#### Enabling DNAT

DNAT is useful if you want to redirect traffic to a proxy or other service to terminate it, or before forwarding the traffic to its original destination. DNAT rewrites the destination IP address, and optionally, the destination port. You can use DNAT to redirect specific traffic to a different destination, as shown in [Figure 4-3](#_bookmark194), which illustrates traffic being redirected from both the router and the server to a proxy at 192.168.0.10 to perform a man-in-the- middle analysis.

### Forwarding Traffic to a Gateway

You’ve set up your gateway device to capture and modify traffic. Everything appears to be working properly, but there’s a problem: you can’t easily change the network configuration of the device you want to capture. Also, you have limited ability to change the network configuration the device is connected to. You need some way to reconfigure or trick the sending device into forwarding traffic through your gateway. You could accomplish this b exploiting the local network by spoofing packets for either DHCP or *Address Resolution Protocol (ARP)*.

#### DHCP Spoofing

DHCP is designed to run on IP networks to distribute network configuration information to nodes automatically. Therefore, if we can spoof DHCP traffic, we can change a node’s network configuration remotely. When DHCP is used, the network configuration pushed to a node can include an IP address as well as the default gateway, routing tables, the default DNS servers, and even additional custom parameters. If the device you want to test uses DHCP to configure its network interface, this flexibility makes it very easy to supply a



custom configuration that will allow easy network traffic capture.

#### ARP Poisoning

ARP is critical to the operation of IP networks running on Ethernet because ARP finds the Ethernet address for a given IP address. Without ARP, it would be very difficult to communicate IP traffic efficiently over Ethernet. Here’s how ARP works: when one node wants to communicate with another on the same Ethernet network, it must be able to map the IP address to an Ethernet MAC address (which is how Ethernet knows the destination node to send traffic to). The node generates an ARP request packet (see [Figure 4-8](#_bookmark207)) containing the node’s 6-byte Ethernet MAC address, its current IP address, and the target node’s IP address. The packet is transmitted on the Ethernet network with a destination MAC address of ff:ff:ff:ff:ff:ff, which is the defined broadcast address. Normally, an Ethernet device only processes packets with a destination address that matches its address, but if it receives a packet with the destination MAC address set to the broadcast address, it will process it, too.

### Final Words

In this chapter, you’ve learned a few additional ways to capture and modify traffic between a client and server. I began by describing how to configure your OS as an IP gateway, because if you can forward traffic through your own gateway, you have a number of techniques available to you.

Of course, just getting a device to send traffic to your network capture device isn’t always easy, so employing techniques such as DHCP spoofing or ARP poisoning is important to ensure that traffic is sent to your device rather than directly to the internet. Fortunately, as you’ve seen, you don’t need custom tools to do so; all the tools you need are either already included in your operating system (especially if you’re running Linux) or easily downloadable.