Overview of tcpdump

As a security analyst, you'll use network protocol analyzers to help defend against any network intrusions. Previously, you learned the following terms related to network monitoring and analysis:

- A **network protocol analyzer (packet sniffer)** is a tool designed to capture and analyze data traffic within a network.
- **Packet sniffing** is the practice of capturing and inspecting data packets across a network.

In this reading, you'll learn more about tcpdump, a network protocol analyzer that can be used to capture and view network communications.

What is tcpdump?

Tcpdump is a command-line network protocol analyzer. Recall that a **command-line interface (CLI)** is a text-based user interface that uses commands to interact with the computer.

Tcpdump is used to capture network traffic. This traffic can be saved to a **packet capture (p-cap)**, which is a file containing data packets intercepted from an interface or network. The p-cap file can be accessed, analyzed, or shared at a later time. Analysts use tcpdump for a variety of reasons, from troubleshooting network issues to identifying malicious activity. Tcpdump comes pre-installed in many Linux distributions and can also be installed on other Unix-based operating systems such as macOS®.

Note: It's common for network traffic to be encrypted, which means data is encoded and unreadable. Inspecting the network packets might require decrypting the data using the appropriate private keys.

Capturing packets with tcpdump

Previously in this program, you learned that a Linux **root user (or superuser)** has elevated privileges to modify the system. You also learned that the **sudo** command temporarily grants elevated permissions to specific users in Linux. Like many other packet sniffing tools, you'll need to have administrator-level privileges to capture network traffic using tcpdump. This means you will need to either be logged in as the root user or have the ability to use the sudo command. Here is a breakdown of the tcpdump syntax for capturing packets:

```
sudo tcpdump [-i interface] [option(s)] [expression(s)]
```

- The `sudo tcpdump` command begins running tcpdump using elevated permissions as sudo.

- The **`-i`** parameter specifies the network interface to capture network traffic. You must specify a network interface to capture from to begin capturing packets. For example, if you specify **`-i any`** you'll sniff traffic from all network interfaces on the system.
- The **`option(s)`** are optional and provide you with the ability to alter the execution of the command. The **`expression(s)`** are a way to further filter network traffic packets so that you can isolate network traffic. You'll learn more about **`option(s)`** and **`expression(s)`** in the next section.

Note: Before you can begin capturing network traffic, you must identify which network interface you'll want to use to capture packets from. You can use the **`-D`** flag to list the network interfaces available on a system.

Options

With tcpdump, you can apply options, also known as flags, to the end of commands to filter network traffic. Short options are abbreviated and represented by a hyphen and a single character like **`-i`**. Long options are spelled out using a double hyphen like **`--interface`**. Tcpdump has over fifty options that you can explore using the manual page. Here, you'll examine a couple of essential tcpdump options including how to write and read packet capture files.

Note: Options are case sensitive. For example, a lowercase **`-w`** is a separate option with a different use than the option with an uppercase **`-W`**.

Note: tcpdump options that are written using short options can be written with or without a space between the option and its value. For example, **`sudo tcpdump -i any -c 3`** and **`sudo tcpdump -iany -c3`** are equivalent commands.

`-w`

Using the **`-w`** flag, you can write or save the sniffed network packets to a packet capture file instead of just printing it out in the terminal. This is very useful because you can refer to this saved file for later analysis. In this command, tcpdump is capturing network traffic from all network interfaces and saving it to a packet capture file named **`packetcapture.pcap`**:

```
sudo tcpdump -i any -w packetcapture.pcap
```

`-r`

Using the **`-r`** flag, you can read a packet capture file by specifying the file name as a parameter. Here is an example of a tcpdump command that reads a file called **`packetcapture.pcap`**:

```
sudo tcpdump -r packetcapture.pcap
```

`-v`

As you've learned, packets contain a lot of information. By default, tcpdump will not print out all of a packet's information. This option, which stands for verbose, lets you control how much packet information you want tcpdump to print out.

There are three levels of verbosity you can use depending on how much packet information you want tcpdump to print out. The levels are **`-v`**, **`-vv`**, and **`-vvv`**. The level of verbosity increases with each added v. The verbose option can be helpful if you're looking for packet information like the details of a packet's IP header fields. Here's an example of a tcpdump command that reads the **`packetcapture.pcap`** file with verbosity:

```
sudo tcpdump -r packetcapture.pcap -v
```

`-c`

The **`-c`** option stands for count. This option lets you control how many packets tcpdump will capture. For example, specifying **`-c 1`** will only print out one single packet, whereas **`-c 10`** prints out 10 packets. This example is telling tcpdump to only capture the first three packets it sniffs from **`any`** network interface:

```
sudo tcpdump -i any -c 3
```

`-n`

By default, tcpdump will perform name resolution. This means that tcpdump automatically converts IP addresses to names. It will also resolve ports to commonly associated services that use these ports. This can be problematic because tcpdump isn't always accurate in name resolution. For example, tcpdump can capture traffic from port 80 and automatically translates port 80 to HTTP in the output. However, this is misleading because port 80 isn't always going to be using HTTP; it could be using a different protocol.

Additionally, name resolution uses what's known as a reverse DNS lookup. A reverse DNS lookup is a query that looks for the domain name associated with an IP address. If you perform a reverse DNS lookup on an attacker's system, they might be alerted that you are investigating them through their DNS records.

Using the **`-n`** flag disables this automatic mapping of numbers to names and is considered to be best practice when sniffing or analyzing traffic. Using **`-n`** will not resolve hostnames, whereas **`-nn`** will not resolve *both* hostnames or ports. Here's an example of a tcpdump command that reads the **`packetcapture.pcap`** file with verbosity and disables name resolution:

```
sudo tcpdump -r packetcapture.pcap -v -n
```

Pro tip: You can combine options together. For example, **`-v`** and **`-n`** can be combined as **`-vn`**. But, if an option accepts a parameter right after it like **`-c 1`** or **`-r capture.pcap`** then you can't combine other options to it.

Expressions

Using filter expressions in tcpdump commands is also optional, but knowing how and when to use filter expressions can be helpful during packet analysis. There are many ways to use filter expressions.

If you want to specifically search for network traffic by protocol, you can use filter expressions to isolate network packets. For example, you can filter to find only IPv6 traffic using the filter expression **`ip6`**.

You can also use boolean operators like **`and`**, **`or`**, or **`not`** to further filter network traffic for specific IP addresses, ports, and more. The example below reads the **`packetcapture.pcap`** file and combines two expressions **`ip and port 80`** using the **`and`** boolean operator:

```
sudo tcpdump -r packetcapture.pcap -n 'ip and port 80'
```

Pro tip: You can use single or double quotes to ensure that tcpdump executes all of the expressions. You can also use parentheses to group and prioritize different expressions. Grouping expressions is helpful for complex or lengthy commands. For example, the command **`ip and (port 80 or port 443)`** tells tcpdump to prioritize executing the filters enclosed in the parentheses before filtering for IPv4.

Interpreting output

Once you run a command to capture packets, tcpdump will print the output of the command as the sniffed packets. In the output, tcpdump prints one line of text for each packet with each line beginning with a timestamp. Here's an example of a command and output for a single TCP packet:

```
sudo tcpdump -i any -v -c 1
```

This command tells tcpdump to capture packets on **`-i any`** network interface. The option **`-v`** prints out the packet with detailed information and the option **`-c 1`** prints out only one packet. Here is the output of this command:

1. **Timestamp**: The output begins with the timestamp, which starts with hours, minutes, seconds, and fractions of a second.
2. **Source IP:** The packet's origin is provided by its source IP address.
3. **Source port:** This port number is where the packet originated.
4. **Destination IP:** The destination IP address is where the packet is being transmitted to.
5. **Destination port:** This port number is where the packet is being transmitted to.

The remaining output contains details of the TCP connection including flags and sequence number. The **`options`** information is additional packet information that the **`-v`** option has provided.

Key takeaways

In security, you'll likely encounter using network protocol analyzer tools like tcpdump. It's important to be equipped with the knowledge of capturing, filtering, and interpreting network packets on the command line.

Resources for more information

- Learn more with tcpdump's tutorials and guides, which includes additional educational resources.
- Learn more about using expressions to filter traffic with this tcpdump tutorial by Daniel Miessler.