LAB ASSIGNMENT No. 5

Aim: Study of Packet Sniffer tool TCPDUMP. Use it to capture and analyze the packet.

Lab Outcome Attained: LO5

Theory:

Tcpdump is a command line utility that allows you to capture and analyze network traffic going through your system. It is often used to help troubleshoot network issues, as well as a security tool.

Below are a few options you can use when configuring Tcpdump.

Options

- -i any : Listen on all interfaces just to see if you're seeing any traffic.
- -i eth0 : Listen on the eth0 interface.
- -D : Show the list of available interfaces
- -n : Don't resolve hostnames.
- -nn : Don't resolve hostnames *or* port names.
- -q : Be less verbose (more quiet) with your output.
- -t : Give human-readable timestamp output.
- -tttt : Give maximally human-readable timestamp output.
- -X : Show the packet's *contents* in both hex and ASCII.
- -XX : Same as -X, but also shows the ethernet header.
- -v, -vv, -vvv : Increase the amount of packet information you get back.
- -c : Only get *x* number of packets and then stop.
- -s : Define the *snaplength* (size) of the capture in bytes. Use -s0 to get everything, unless you are intentionally capturing less.
- -S : Print absolute sequence numbers.
- -e : Get the ethernet header as well.

1. To install tcpdump

$ sudo apt-get install tcpdump

2. Choosing an interface:
By default, tcpdump captures packets on all interfaces. To view a summary of available interfaces, run the command:

tcpdump -D

3. Basic command for sniffing

tcpdump -n

The -n parameter is given to stop tcpdump from resolving ip addresses to hostnames.

Now lets increase the display resolution of this packet, or get more details about it. The verbose switch comes in handy. Here is a quick example

4. tcpdump -v -n

5. Getting the ethernet header (link layer headers)

In the above examples details of the ethernet header are not printed. Use the -e option to print the ethernet header details as well.

Filtering packets using expressions

6. selecting packets with specific protocol

tcpdump -n tcp

#tcpdump -n icmp

7. Capturing traffic from particular host or port

Expressions can be used to specify source ip, destination ip, and port numbers. The next example picks up all those packets with source address 172.16.92.1

tcpdump -n src 172.16.92.1

tcpdump -n dst 172.16.92.1

tcpdump -n port 80

tcpdump port 80

Specific Packets from specific port

tcpdump udp and src port 53

observing packets within a specific port range

tcpdump -n portrange 1-80
It shows all packets whose source or destination port is between 1 to 80

tcpdump -n src port 443

Writing Captures to a File

It's often useful to save packet captures into a file for analysis in the future. These files are known as PCAP (PEE-cap) files, and they can be processed by hundreds of different applications, including network analyzers, intrusion detection systems, and of course by tcpdump itself. Here we're writing to a file called *capture_file* using the -w switch.

tcpdump port 80 -w capture_file

Reading PCAP files

You can read PCAP files by using the -r switch. Note that you can use all the regular commands within tcpdump while reading in a file; you're only limited by the fact that you can't capture and process what doesn't exist in the file already.

tcpdump -r capture_file

From specific IP and destined for a specific Port

Let's find all traffic from 10.5.2.3 going to any host on port 3389.

tcpdump -nnvvS src 10.5.2.3 and dst port 3389

From One Network to Another

Let's look for all traffic coming from 192.168.x.x and going to the 10.x or 172.16.x.x networks, and we're showing hex output with no hostname resolution and one level of extra verbosity.

tcpdump -nvX src net 192.168.0.0/16 and dst net 10.0.0.0/8 or 172.16.0.0/16

Non ICMP Traffic Going to a Specific IP

This will show us all traffic going to 192.168.0.2 that is *not* ICMP.

tcpdump dst 192.168.0.2 and src net and not icmp

Complex Grouping and Special Characters

Also keep in mind that when you're building complex queries you might have to group your options using single quotes. Single quotes are used in order to tell tcpdump to ignore certain special characters—in this case below the "( )" brackets. This same technique can be used to group using other expressions such as host, port, net, etc. Take a look at the command below.

(incorrect)

tcpdump src 10.0.2.4 and (dst port 3389 or 22)

If you tried to run this otherwise very useful command, you'd get an error because of

the parenthesis. You can either fix this by escaping the parenthesis (putting a before each one), or by putting the entire command within single quotes:

(correct)

tcpdump 'src 10.0.2.4 and (dst port 3389 or 22)'

Isolating Specific TCP Flags:

You can also capture traffic based on specific TCP flag(s).

The filters below find these various packets because tcp[13] looks at offset 13 in the TCP header, the number represents the location within the byte, and the !=0 means that the flag in question is set to 1, i.e. it's on.

Show me all URGENT **(URG)** packets…

tcpdump 'tcp[13] & 32!=0'

Show me all ACKNOWLEDGE **(ACK)** packets…

tcpdump 'tcp[13] & 16!=0'

Show me all PUSH **(PSH)** packets…

tcpdump 'tcp[13] & 8!=0'

Show me all RESET **(RST)** packets…

tcpdump 'tcp[13] & 4!=0'

Show me all SYNCHRONIZE **(SYN)** packets…

tcpdump 'tcp[13] & 2!=0'

Show me all FINISH **(FIN)** packets…

tcpdump 'tcp[13] & 1!=0'

Show me all SYNCHRONIZE/ACKNOWLEDGE **(SYNACK)** packets…

tcpdump 'tcp[13]=18'

Only the PSH, RST, SYN, and FIN flags are displayed in tcpdump's flag field output. URGs and ACKs are displayed, but they are shown elsewhere in the output rather than in the flags field.

As with most powerful tools, however, there are multiple ways to do things. The example below shows another way to capture packets with specific TCP flags set.

tcpdump 'tcp[tcpflags] == tcp-syn'

tcpflags option…

tcpdump 'tcp[tcpflags] == tcp-rst'

tcpflags option…

tcpdump 'tcp[tcpflags] == tcp-fin'