Using tshark to Watch and Inspect Network Traffic

From http://www.linuxjournal.com/content/using-tshark-watch-and-inspect-network-traffic

Most of you probably have heard of Wireshark, a very popular and capable network protocol analyzer. What you may not know is that there exists a console version of Wireshark called tshark. The two main advantages of tshark are that it can be used in scripts and on a remote computer through an SSH connection. Its main disadvantage is that it does not have a GUI, which can be really handy when you have to search lots of network data.

You can get tshark either from its Web site and compile it yourself or from your Linux distribution as a precompiled package. The second way is quicker and simpler. To install tshark on a Debian 7 system, you just have to run the following command as root:

```
# apt-get install tshark
Reading package lists... Done
Building dependency tree
Reading state information... Done

The following extra packages will be installed:
libc-ares2 libcap2-bin libpam-cap libsmi2ldbl
libwireshark-data libwireshark2
libwiretap2 libwsutil2 wireshark-common

Suggested packages:
libcap-dev snmp-mibs-downloader wireshark-doc

The following NEW packages will be installed:
libc-ares2 libcap2-bin libpam-cap libsmi2ldbl
libwireshark-data libwireshark2
libwiretap2 libwsutil2 tshark wireshark-common

0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.
Need to get 15.6 MB of archives.

After this operation, 65.7 MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
```

To find out whether tshark is installed properly, as well as its version, execute this command:

```
$ tshark -v TShark 1.8.2
```

Note: this article assumes that you already are familiar with network data, TCP/IP, packet capturing and maybe Wireshark, and that you want to know more about tshark.

About tshark

tshark can do anything Wireshark can do, provided that it does not require a GUI. It also can be used as a replacement for tcpdump, which used to be the industry standard for network data capturing. Apart from the capturing part, where both tools are equivalent, tshark is more powerful than tcpdump; therefore, if you want to learn just one tool, tshark should be your choice.
As you can imagine, tshark has many command-line options. Refer to its man page for the full list.

Capturing Network Traffic Using tshark

The first command you should run is `sudo tshark -D` to get a list of the available network interfaces:

```
$ sudo tshark -D
1. eth0
2. nflog (Linux netfilter log (NFLOG) interface)
3. any (Pseudo-device that captures on all interfaces)
4. lo
```

If you run tshark as a normal user, you most likely will get the following output, because normal users do not have direct access to network interface devices:

```
$ tshark -D tshark: There are no interfaces on which a capture can be done
```

The simplest way of capturing data is by running tshark without any parameters, which will display all data on screen. You can stop data capturing by pressing Ctrl-C.

The output will scroll very fast on a busy network, so it won't be helpful at all. Older computers could not keep up with a busy network, so programs like tshark and tcpdump used to drop network packets. As modern computers are pretty powerful, this is no longer an issue.

Saving and Reading Network Data Using Files

The single-most useful command-line parameter is -w, followed by a filename. This parameter allows you to save network data to a file in order to process it later. The following tshark command captures 500 network packets (-c 500) and saves them into a file called LJ.pcap (-w LJ.pcap):

```
$ tshark -c 500 -w LJ.pcap
```

The second-most useful parameter is -r. When followed by a valid filename, it allows you to read and process a previously captured file with network data.

Capture Filters

Capture filters are filters that are applied during data capturing; therefore, they make tshark discard network traffic that does not match the filter criteria and avoids the creation of huge capture files. This can be done using the -f command-line parameter, followed by a filter in double quotes.
The most important TCP-related Field Names used in capture filters are tcp.port (which is for filtering the source or the destination TCP port), tcp.srcport (which is for checking the TCP source port) and tcp.dstport (which is for checking the destination port).
Generally speaking, applying a filter after data capturing is considered more practical and versatile than filtering during the capture stage, because most of the time, you do not know in advance what you want to inspect. Nevertheless, if you really know what you're doing, using capture filters can save you time and disk space, and that is the main reason for using them.
Remember that filter strings always should be written in lowercase.

Display Filters

Display filters are filters that are applied after packet capturing; therefore, they just “hide” network traffic without deleting it. You always can remove the effects of a display filter and get all your data back.

Display Filters support comparison and logical operators. The http.response.code == 404 && ip.addr == 192.168.10.1 display filter shows the traffic that either comes from the 192.168.10.1 IP address or goes to the 192.168.10.1 IP address that also has the 404 (Not Found) HTTP response code in it. The !bootp && !ip filter excludes BOOTP and IP traffic from the output. The eth.addr == 01:23:45:67:89:ab && tcp.port == 25 filter displays the traffic to or from the network device with the 01:23:45:67:89:ab MAC address that uses TCP port 25 for its incoming or outgoing connections.

When defining rules, remember that the ip.addr != 192.168.1.5 expression does not mean that none of the ip.addr fields can contain the 192.168.1.5 IP address. It means that one of the ip.addr fields should not contain the 192.168.1.5 IP address! Therefore, the other ip.addr field value can be equal to 192.168.1.5! You can think of it as “there exists one ip.addr field that is not 192.168.1.5”. The correct way of expressing it is by typing !(ip.addr == 192.168.1.5). This is a common misconception with display filters.

Also remember that MAC addresses are truly useful when you want to track a given machine on your LAN, because the IP of a machine can change if it uses DHCP, but its MAC address is more difficult to change.

Display filters are extremely useful tools when used correctly, but you still have to interpret the results, find the problem and think about the possible solutions yourself. It is advisable that you visit the display filters reference site for TCP-related traffic at http://www.wireshark.org/docs/dfref/t/tcp.html. For the list of all the available field names related to UDP traffic, see http://www.wireshark.org/docs/dfref/u/udp.html.