The purpose of this project was to capture packets, save them and use filters to display certain HTTP and HTTPS packets. The packets to examined are TCP/IP packets going to or from a server.

Scenario:

I am a security analyst working for a company that wants to detect certain TCP/IP packets on the server, specifically web traffic. The company wants to test an external server by looking at TCP/IP requests and responses, as well as look at inappropriate website access through the server. The IT manager wants to be able to capture certain ethernet network traffic and be able to detect certain IP addresses as well.

## Objective 1 – Install on Ubuntu and set up Wireshark for users belonging to the Wireshark group.

To get the latest version of Wireshark the command sudo add-apt-repository ppa:wireshark-dev/stableis typed. To just get Wireshark just add-apt install wireshark would have surficed.

See also: Installing Wireshark on Ubuntu.png

In addition a user needed to be added to the wireshark group, and this had to be done using sudo permission. The command is sudo usermod -aG wireshark $USER**which will add the existing user in the group.**

**See also: Adding a new user to the wireshark group with sudo permission.png**

**Finally, the now log user logs out, and logs in as himself. This is necessary because as a security analyst I want to use the principle of least admission. The user needs to have access to Wireshark but not be able save any files under root nor have any other sudo permissions.**

## Objective 2 - Start a packet capture on an ethernet port and save it to file

**Opening Wireshark shows the available interfaces. Lookback capture for instance shows the packet capture from a machine to itself such as running one’s own web application on local host.**

**Here en will be used for ethernet (ens5).**

**See also: The Wireshark interface.**

**See also: Capture, stop capturing and save packets in Wireshark**

## Objective 3 - Use a display filter to detect HTTPS packets

**A display filter was used, and not a capture filter. Specifically port 443 to capture only HTTPS traffic.**

**See also: Display captured packets using tcp port 443.png**

**See also: Information about the captured package using tcp port 443.png**

## Objective 3 extra - Use a display filter to detect HTTPS packets

Capture only HTTP traffic (not HTTPS) from a website, and save that to a file.

HTTP usus port 80, so after using tcp.port == 80 as a display filter in Wireshark interface, the result can be shown in the attached file below.

See also: Display captured packets for HTTP traffic.png

## Objective 4 - Visit a web page and detect its IP address using a display filter

There may be an incident where a certain known IP address causes problem for the server.

We can use the filter tls.handshake == 1

See also: Client Hello for two websites.png

See also: Display all traffic from that website.png

See also: Display when the website is sending something.png

See also: Display all cases when we are sending something to that website.png

## Objective 5 - Locate all HTTPS packets from a capture not containing a certain IP address

It is important to locate import data to avoid clutter.

See also: Traffic from one of the websites.png

See also: Traffic from one of the websites and only HTTPS.png

See also: The traffic that does not come from that website and is HTTPS

See also: Traffic not from that website and are both HTTP and HTTPS.png

## Final objective – use Wireshark to create a capture file, and then use a display filter to list all https and http packets. Then eliminate one IP address from the capture using a filter

First step is using tcp.port == 80 as a filter which singled out one IP know to be http.

The next step is to eliminate that and show all traffic for the rest of the IP addresses.

See also: Capstone final.png

That displays the final result.