Final Project – Pcap Visualizer

CS6963 Digital Forensics

Professor Mark Budofsky

Amandeep Singh 12/10/2017

Forensics Final Project - Pcap Visualizer

The purpose of this project is to create an open source tool to visualize a pcap file. The tool is able to input a pcap file and output a visual representation of the network depicted in the pcap file. Furthermore, it also shows the type of protocols/services being used by the nodes/hosts in the network.

Dependencies

Python 2.7.12

Linux Dependencies

```
sudo apt-get install python-pip python-tk
sudo pip install networkx pypcapfile matplotlib
```

Windows bash Dependencies

```
sudo apt-get install python-pip python-tk
sudo pip install networkx pypcapfile matplotlib
```

To see the graph in windows, download xming server from https://sourceforge.net/projects/xming/files/latest/download.

Install it using default settings. Set the $DISPLAY variable appropriately:

```
export DISPLAY=127.0.0.1:0.0
echo $DISPLAY
```

It should have been set to 127.0.0.1:0.0 and that is where the xming server is running

Usage

python pcap_parser_project.py [-h] [-i IP_ADDR] [-b START] [-f END] [-r] [-s] pcap outfile

Mandatory Arguments

1. Pcap file

Input pcap file to be processed by the tool. This file must be in the same directory as the tool (pcap_parser_project.py)

2. Output image file name

Name of the image file you want the graph (generated by the tool) to be stored as

Optional Arguments

1. IP Address (-i <IP address> --ip_addr=<IP address>)

If you want to focus the analysis on a specific IP address, you can give that ip address and the tool will only display the connections related to that ip address

2. Starting Connection Edge (-b <Start edge number> --start=<Start edge number>)

If the visual representation of the network gets too clustered, you can use this option along with -f option to limit the number of connection edges you want to see in the graph. This argument is the starting connection edge you want to see out of the available connection edges. You must use -f option if you use this option. Default value of this argument is 0.

3. Ending Connection Edge (-f <Last edge number> --end=<Last edge number>)

This argument is the upper limit of the connection edges you want to see. You must use this option if you use -b option.

4. Recurse (-r --recurse)

This option can be set if you have given a specific IP address to analyze and you also want to see the network connections the nodes connected to this IP address are making.

5. Spring Layout (-s --spring_layout)

The default layout of the graph is shell. The layout can be changed to spring layout by using this option. Spring layout helps in identifying the number of networks in the pcap file.

Methodology

At first, I tried to use pypcapfile python package https://pypi.python.org/pypi/pypcapfile to parse through the entire pcap file and extract the required information. This package provides an easy way to load all packets from the pcap file, however, it lacks the functionality to extract additional application layer protocol information. Moreover it extends ctypes.Structure class for its link, network and transport layer protocol classes. So, I decided to write my own link, network and transport layer classes to handle those layers. I referenced pypcapfile package for this purpose. My classes do not use any ctypes and are easy to use.

The tool parses through each packet in the pcap file and extracts the ethernet header, network header and transport layer frame, finally inferring the application layer protocol. Ethernet, ip, tcp and udp classes look at the packet's appropriate bytes and cross references it with its corresponding header to extract information from the packets.

The data structure that I used to save all the information is a list of nodes. Each node is represented by a set of IP address and ethernet address. No two nodes will have the same IP address and ethernet address combination. Node class also stores a list of edges that denotes the network connections between nodes. Each edge is a tuple of two nodes.

A few problems I came across while writing this tool were that one network connection may have hundreds of packets corresponding to it. If I use all those packets as individual edges, the graph will get too messy and we won't be able to clearly see the network. I had to make sure one network connection (comprised of hundreds of packets) corresponds to only one edge.

Even after the above modification, the network graph was still getting too messy. So, I added another feature to the tool that will give the user the option to view the graph a few edges at a time. (using -b and -f options). For Example, if the tool outputs a graph with 100 network connection edges for a certain command, and you want to only see the 40^{th} to 70^{th} network connections, you can use '-b 40 -f 70' options. These options will only display the 40^{th} to 71^{st} network connections. This feature added more flexibility to the tool.