**PcapXray Design Specification**

**Goal:**

* Given a Pcap File, plot a network diagram displaying hosts in the network, network traffic, highlight important traffic and Tor traffic as well as potential malicious traffic including data involved in the communication.

**Problem:**

* Investigation of a Pcap file takes a long time given initial glitch to start the investigation
  + Faced by every forensics investigator and anyone who is analyzing the network

**Location:** *https://github.com/Srinivas11789/PcapXray*

**Solution:** *Speed up the investigation process*

* Make a network diagram with the following features from a Pcap file
  + **Tool Highlights:**
    - Network Diagram – Summary Network Diagram of full network
    - Information:
      * Traffic with Server Details
      * Tor Traffic
      * Possible Malicious traffic
      * Data Obtained from Packet in Report – Device/Traffic/Payloads
      * Device Details

**Components:**

* **Mandatory:**
  + Network Diagram
  + Device/Traffic Details and Analysis
  + Malicious Traffic Identification
  + Tor Traffic
  + GUI – a gui with options to upload pcap file and display the network diagram
* **Optional but Useful:**
  + Files Exchanged
  + Server Details in traffic

PcapRead Module

**Block Diagram:**

PlotMAP

Tk Widget GUI

Pcap File

Tor/malicious

Json DB

TrafficInfo

Front End

Back End

**Method or Process Description:**

* **Module1 – main.py - Main (Driver):**
  + Main program driver
  + Drives the whole Application by spawning a TK widget interface
* **Module2 – userInterface.py – GUI:**
  + Used **Tk** and **Ttk** Widget for the Graphical User Interface
  + Designed a UI with three frames,
    - First frame, **accepting input file** from the user and **Button action**, an added gimmick of **progress bar** showing progressing scenario
    - Second frame, providing **options to select** from a list to display different graphs in the third frame
    - Third frame initially contains a label displaying the tool information
      * Based on the option setting at second frame, it **displays different graphs** in the third frame
    - Button action calls packet read to initially **perform pcap reading** and update the **json database or dictionary**
    - Option action or option **variable trace** observes change in the option value and triggers function call to **plotLan or draw graph** and display
* **Module3 – pcapReader.py – Pcap Reading:**
  + Reads the given packet capture file and populates a dictionary of various information of the packets
    - First Key of the Dictionary is DB[ip] **– collects the private Ips**
    - Second Key of the Dictionary is **TCP or UDP** – Basis of communication
    - Third Key of the Dictionary is **HTTP, HTTPS, Ports Connected** information
    - Few other keys collecting the **HTTP Servers, Payload** also are segregated
    - Json DB Structure:
      * ***DB[PrivateIp]*** 
        + ***TCP***

***HTTP***

***Server***

***Payload***

***HTTPS***

***PortsConnected***

* + - * + ***UDP***

***PortsConnected***

* + - * + ***Ethernet***
* **Module4 – plotLanNetwork.py – Network Graph Drawing:**
  + Uses graphviz module to plot network graph
  + Classifies **all the private IP** in the network from the packetDB into **nodes**
  + **Traces all the traffic** based on the category under consideration and **draws edges**
  + **Style** added to differentiate different traffic
* **Module 5 – torTrafficHandle.py – Tor Traffic Detection:**
  + Obtains **consensus data** from **the tor authority nodes** using **the stem library** and matches all the destination address of packets to view any match
  + Classifies destination of such an address as a potential Tor traffic displayed with a **white edge**
* **Module 6 – maliciousTrafficIdentifier.py – Malicious Traffic Detection:**
  + Obtains the **Non-resolved IP address** (by reverse DNS lookup) or connection to any **unknown ports or not well-known ports** are assumed to be a malicious connection
  + Well known ports database is kept small as of now with the most well-known ports such as 53, 80, 443. It should be updated with a proper db of well-known ports to compare against.
* **Module 7 – communicationDetailsFetch.py – Traffic Details Fetch**
  + **Ipwhois:**
    - Ipwhois details are fetched with the ipwhois library
    - Every ip is resolved for the whois information and the report is updated
    - This feature already exists but is kept disabled to achieve performance and speed (Ex: scenario to solve: Some pcap files contain over 100 hosts)
  + **Reverse dns lookup:**
    - Reverse dns lookup is performed with the socket library which is default and domain name is obtained from gethostbyaddr function
* **Module 8 – deviceDetailsFetch.py – Device Details Fetch**
  + Device details are obtained from the Ethernet key of the packet DB
    - For each private IP the mac OUI is compared with the OUI database and information is fetched
* **Module 9 – reportGen.py – Report Generator**
  + Report generator module generates report at a given path,
    - *Device details*
    - *Communication details*
    - *HTTPPayload details*
  + Copies all the Json database contents into the files based on the category

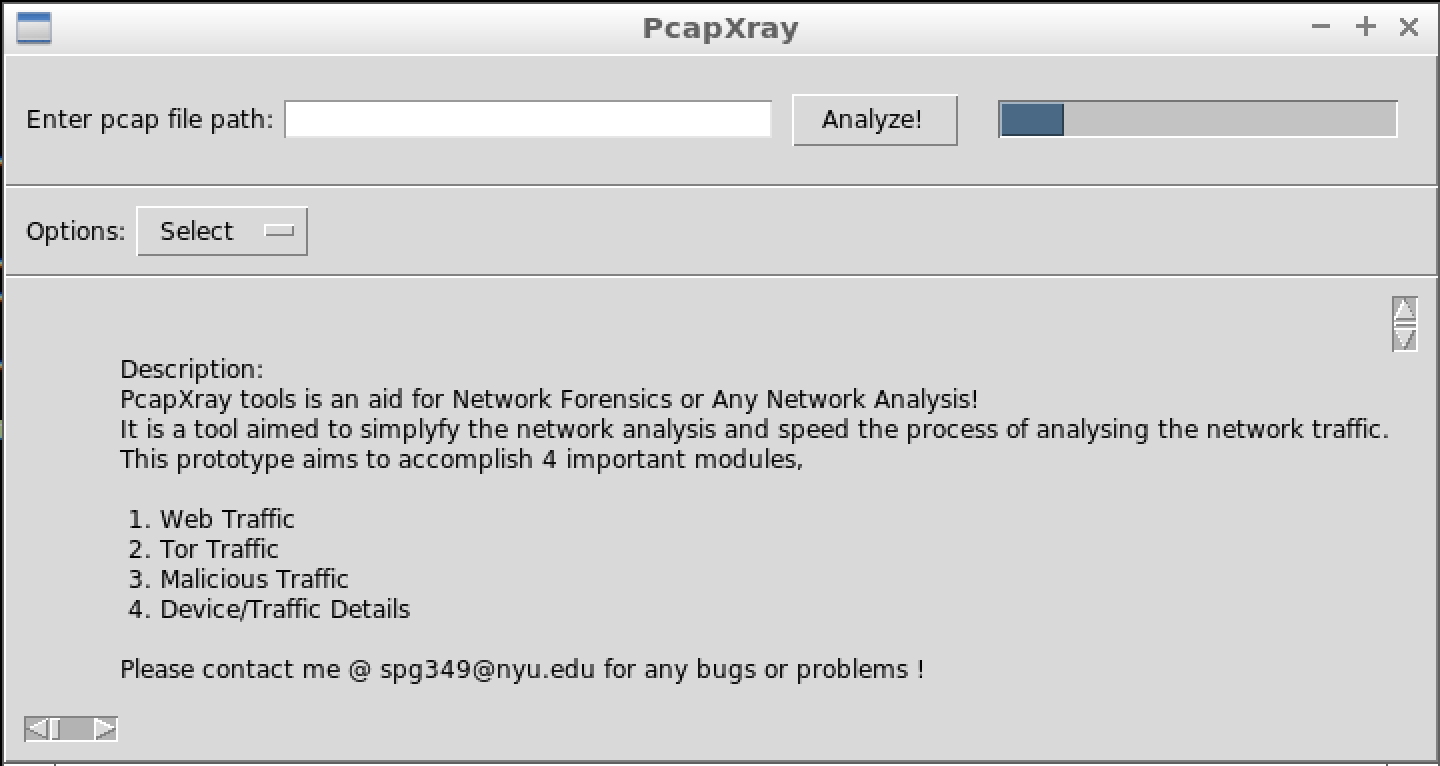
**Output:**

* Provides network graph of all the different traffic – Tor, Malicious, All, HTTP and HTTPS
* Create a Report Folder to dump all the “PNG” files of different graphs
  + Generates files with information from the database

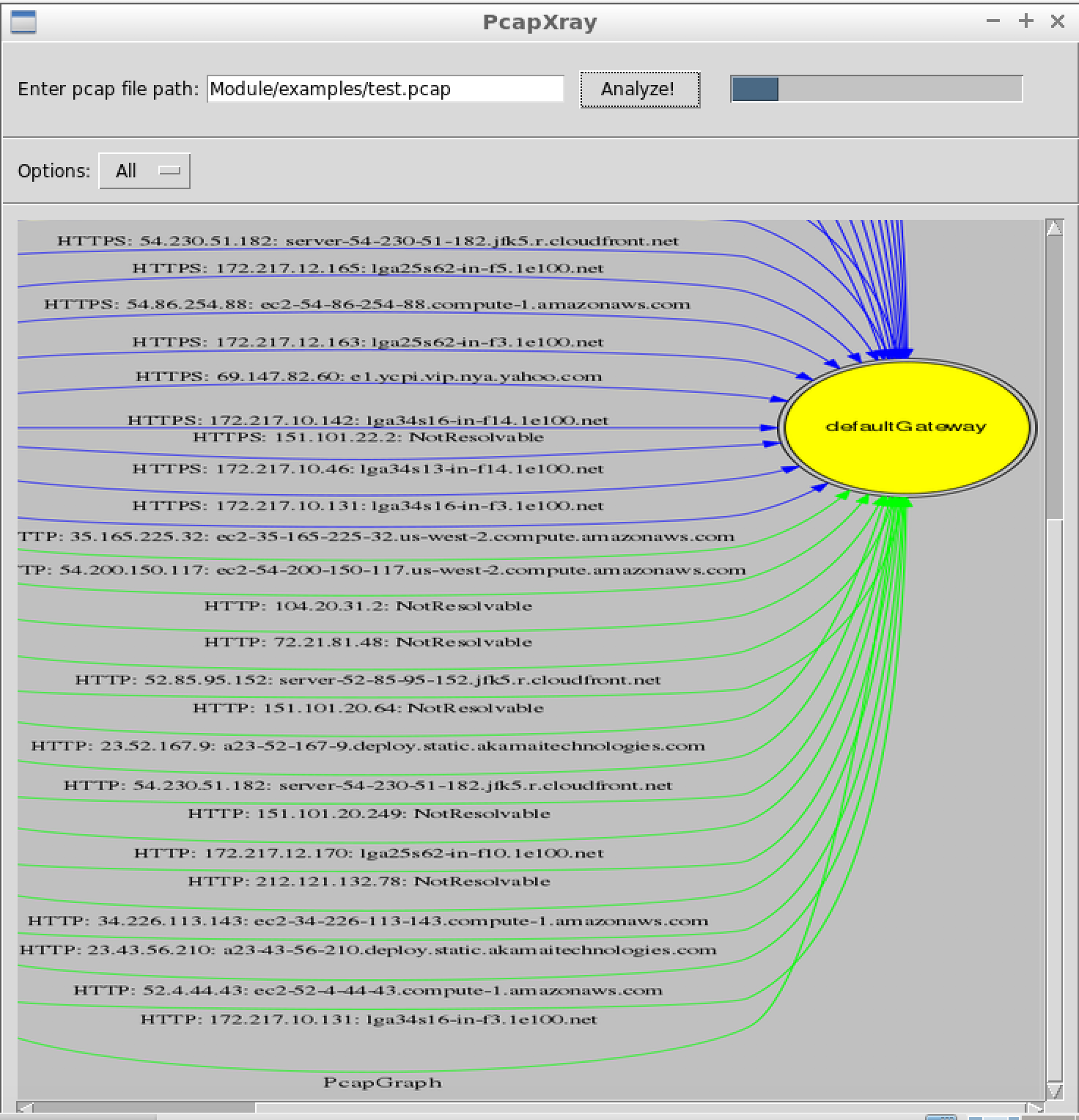
**Python Libraries Used: - All these libraries are required for functionality**

* + - * Tkinter and TTK – Install from pip or apt-get – Ensure Tkinter and graphviz is installed (Most Linux contain by default)
        + **apt install python-tk**
        + **apt install graphviz**
      * **All these are included in the requirements.txt file**
        + Scapy – rdpcap to read the packets from the pcap file
        + Ipwhois – to obtain whois information from ip
        + Netaddr – to check ip information type
        + Pillow – image processing library
        + Stem – tor consensus data fetch library
        + pyGraphviz – plot graph
        + Networkx – plot graph
        + Matplotlib – plot graph

**Demo: Screen shots:** Initial Screen

****

**Result Screen:**

****

**Challenges:**

* **Unstability of the TK GUI:**
  + Decision on the GUI between Django and TK, settled upon tk for a simple local interface, but the unstability of the tk gui caused a number of problems
* **Graph Plotting:**
  + Plotting a proper network graph which is readable from the data obtained was quite an effort, used different libraries to arrive at one.
* **Performance and Timing:**
  + The performance and timing of the total application was a big challenge with different data gathering and output generation

**Known Bugs:**

* **Memory Hogging**
  + Sometimes memory hogging occurs when lower RAM is present in the system as the data stored in the memory from the pcap file is huge
    - Should be Fixed by moving data into a database than the memory itself
* **Race Condition**
  + Due to mainloop of the TK gui, other threads could undergo a race condition
    - Should be fixed by moving to a better structured TK implementation or Web GUI
* **Tk GUI Unstability:**
  + Same reason as above
* Current Fix in rare occasions: If any of the above issue occurs the progress bar keeps running and no output is generated, a restart of the app would be required.

**Future:**

* Change the database from JSON to sqlite or prominent database, due to memory hogging
* Change fronend to web based such as Django
* Make the application more stable

**References:**

* *https://graphviz.gitlab.io/\_pages/doc/info/lang.html*
* [*https://www.swharden.com/wp/2010-03-03-viewing-large-images-with-scrollbars-using-python-tk-and-pil/*](https://www.swharden.com/wp/2010-03-03-viewing-large-images-with-scrollbars-using-python-tk-and-pil/)
* [*https://stackoverflow.com/questions/40025616/multithreading-from-a-tkinter-app*](https://stackoverflow.com/questions/40025616/multithreading-from-a-tkinter-app)
* [*https://stackoverflow.com/questions/6893968/how-to-get-the-return-value-from-a-thread-in-python*](https://stackoverflow.com/questions/6893968/how-to-get-the-return-value-from-a-thread-in-python)
* [*https://pythonhaven.wordpress.com/2009/12/09/generating\_graphs\_with\_pydot/*](https://pythonhaven.wordpress.com/2009/12/09/generating_graphs_with_pydot/)
* [*http://graphviz.readthedocs.io/en/stable/examples.html*](http://graphviz.readthedocs.io/en/stable/examples.html)
* [*https://medium.com/@vworri/extracting-the-payload-from-a-pcap-file-using-python-d938d7622d71*](https://medium.com/@vworri/extracting-the-payload-from-a-pcap-file-using-python-d938d7622d71)
* [*http://isrg.blogs.southwales.ac.uk/2009/09/16/graphviz-python-and-tk/*](http://isrg.blogs.southwales.ac.uk/2009/09/16/graphviz-python-and-tk/)
* [*http://www.tkdocs.com/tutorial/firstexample.html*](http://www.tkdocs.com/tutorial/firstexample.html)
* [*https://graph-tool.skewed.de/static/doc/index.html*](https://graph-tool.skewed.de/static/doc/index.html)
* [*https://networkx.github.io/documentation/networkx-1.10/reference/classes.multidigraph.html*](https://networkx.github.io/documentation/networkx-1.10/reference/classes.multidigraph.html)
* [*http://isrg.blogs.southwales.ac.uk/2009/09/16/graphviz-python-and-tk/*](http://isrg.blogs.southwales.ac.uk/2009/09/16/graphviz-python-and-tk/)
* [*http://matthiaseisen.com/articles/graphviz/*](http://matthiaseisen.com/articles/graphviz/)
* [*https://stem.torproject.org/tutorials/mirror\_mirror\_on\_the\_wall.html*](https://stem.torproject.org/tutorials/mirror_mirror_on_the_wall.html)
* [*http://effbot.org/tkinterbook/frame.html*](http://effbot.org/tkinterbook/frame.html)
* [*https://pythonprogramming.net/styling-gui-bit-using-ttk/*](https://pythonprogramming.net/styling-gui-bit-using-ttk/)
* [*https://stackoverflow.com/questions/6865792/fresh-tutorial-on-tkinter-and-ttk-for-python-3*](https://stackoverflow.com/questions/6865792/fresh-tutorial-on-tkinter-and-ttk-for-python-3)
* *http://coreygoldberg.blogspot.com/2009/12/python-3-tkinter-ttk-tk-themed-widgets\_07.html*