**Design Document – PcapXray** ***Srinivas Piskala Ganesh Babu – spg349 and N13138339***

# 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*

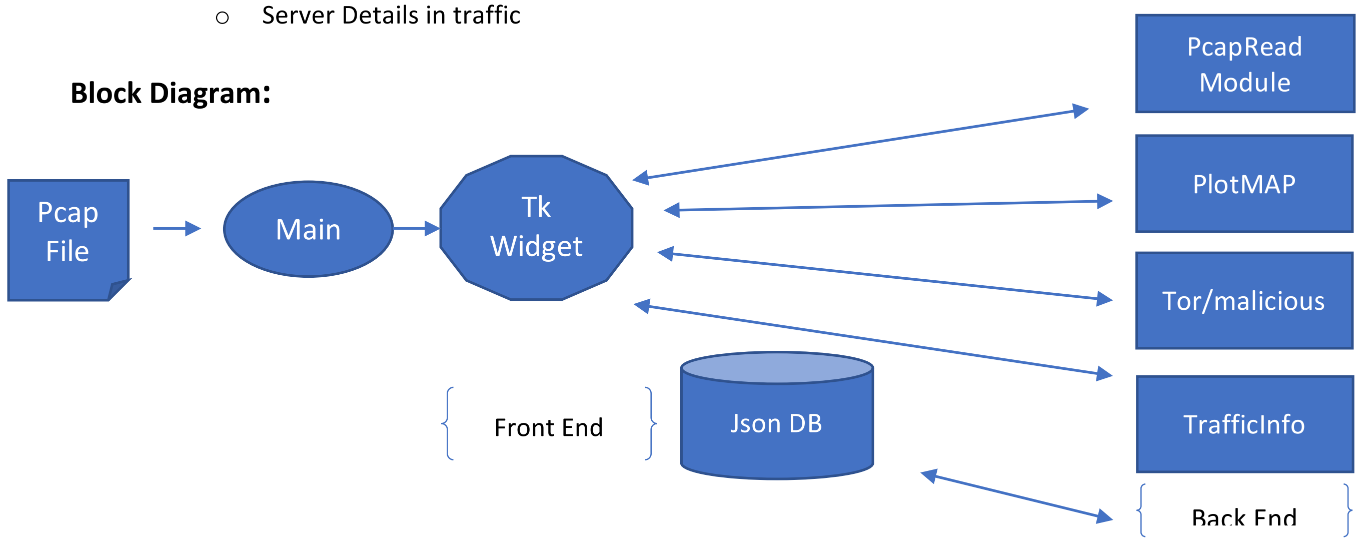
**Solution:** *Speed up the investigation process*

- 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

## Block Diagram:



**Design Document – PcapXray** *Srinivas Piskala Ganesh Babu – spg349 and N13138339*

## 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

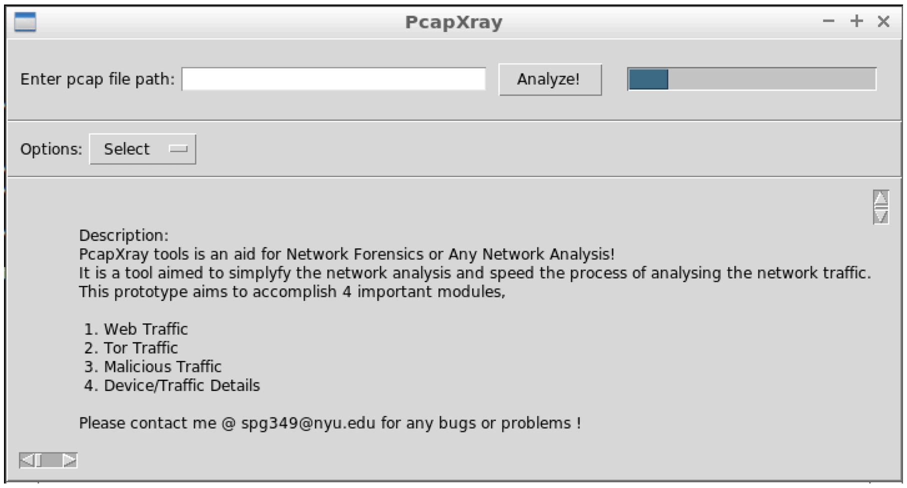
- **Module 8 – deviceDetailsFetch.py – Device Details Fetch**
  - Device details are obtained from the Ethernet key of the packet DB
    - For each private IP the mac OUI is compared with the OUI database and information is fetched
- **Module 9 – reportGen.py – Report Generator**
  - Report generator 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

**Design Document – PcapXray** ***Srinivas Piskala Ganesh Babu – spg349 and N13138339***

## Demo: Screen shots: Initial Screen



Enter pcap file path:

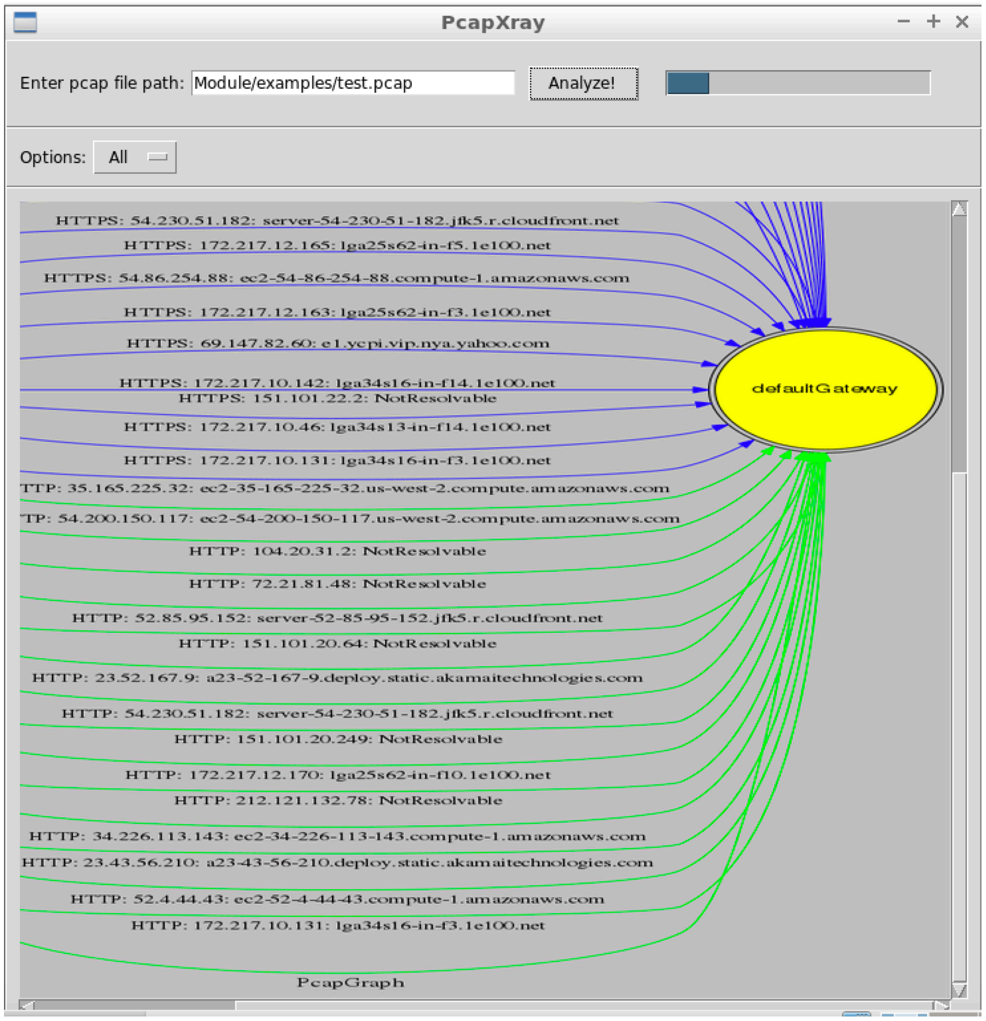
Options: Select

Description:
PcapXray tools is an aid for Network Forensics or Any Network Analysis!
It is a tool aimed to simplyfy the network analysis and speed the process of analysing the network traffic.
This prototype aims to accomplish 4 important modules,

1. Web Traffic
2. Tor Traffic
3. Malicious Traffic
4. Device/Traffic Details

Please contact me @ spg349@nyu.edu for any bugs or problems !

## Result Screen:


PcapXray
Analyze!
HTTPS: 54.230.51.182: server-54-230-51-182.jfk5.r.cloudfront.net
HTTPS: 172.217.12.165: lga25s62-in-f5.1e100.net
HTTPS: 54.86.254.88: ec2-54-86-254-88.compute-1.amazonaws.com
HTTPS: 172.217.12.163: lga25s62-in-f3.1e100.net
HTTPS: 69.147.82.60: e1.ycpi.vip.nya.yahoo.com
HTTPS: 172.217.10.142: lga34s16-in-f14.1e100.net
HTTPS: 151.101.22.2: NotResolvable
HTTPS: 172.217.10.46: lga34s13-in-f14.1e100.net
HTTPS: 172.217.10.131: lga34s16-in-f3.1e100.net
TTP: 35.165.225.32: ec2-35-165-225-32.us-west-2.compute.amazonaws.com
TP: 54.200.150.117: ec2-54-200-150-117.us-west-2.compute.amazonaws.com
HTTP: 104.20.31.2: NotResolvable
HTTP: 72.21.81.48: NotResolvable
HTTP: 52.85.95.152: server-52-85-95-152.jfk5.r.cloudfront.net
HTTP: 151.101.20.64: NotResolvable
HTTP: 23.52.167.9: a23-52-167-9.deploy.static.akamaitechnologies.com
HTTP: 54.230.51.182: server-54-230-51-182.jfk5.r.cloudfront.net
HTTP: 151.101.20.249: NotResolvable
HTTP: 172.217.12.170: lga25s62-in-f10.1e100.net
HTTP: 212.121.132.78: NotResolvable
HTTP: 34.226.113.143: ec2-34-226-113-143.compute-1.amazonaws.com
HTTP: 23.43.56.210: a23-43-56-210.deploy.static.akamaitechnologies.com
HTTP: 52.4.44.43: ec2-52-4-44-43.compute-1.amazonaws.com
HTTP: 172.217.10.131: lga34s16-in-f3.1e100.net
defaultGateway
PcapGraph

Enter pcap file path: Module/examples/test.pcap

Options: All

# Design Document – PcapXray

*Srinivas Piskala Ganesh Babu – spg349 and N13138339*

## 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://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://pythonhaven.wordpress.com/2009/12/09/generating_graphs_with_pydot/*
- *http://graphviz.readthedocs.io/en/stable/examples.html*
- *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://www.tkdocs.com/tutorial/firstexample.html*
- *https://graph-tool.skewed.de/static/doc/index.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://matthiaseisen.com/articles/graphviz/*
- *https://stem.torproject.org/tutorials/mirror_mirror_on_the_wall.html*
- *http://effbot.org/tkinterbook/frame.html*
- *https://pythonprogramming.net/styling-gui-bit-using-ttk/*
- *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*