

Report

Angelos Kelekoglou

Email : aggeloskel@outlook.com

Academic email: akelekog@csd.auth.gr

Github: <https://github.com/akelekog>

Project Title: Network Information and Packet Capturing Tool.

Project Specifications , notes :

The project is implemented on jupyter notebook .

To run the project it is imperative that these libraries are installed and imported :

- 1) Scapy
- 2) Tkinter (gui)
- 3) Psutil (already in standard python)
- 4) Geolite2 (GeoLite2 is a free and open-source geolocation database developed by MaxMind that provides geolocation information for IP addresses allowing to find the approximate geographic location of an IP address.)

Results interpretation:

The project results are visible in the gui of the application :

- 1) Mb sent , received and total amount
- 2) Captured packets and source/destination geolocation
- 3) Connected devices on network(IP)

Implementation:

This is a simple Network Information and Packet Capturing Tool Python program designed to provide network traffic information and capture packets from a network interface. It utilizes various libraries such as Scapy, tkinter, psutil, geolite2, and socket to achieve its functionality.

The project starts by importing the necessary modules, Scapy for packet manipulation, tkinter for GUI development, psutil for system information retrieval, geolite2 for IP geolocation, and socket for working with IP addresses and network communication.

Proceeding there is the initialization of the geolite2 reader object to enable geolocation on IP addresses. We proceed on to retrieve initial number of bytes received, sent, and the total using the psutil module. These values will be used to calculate the network traffic.

We continue to define the `update_info()` function which is responsible for realtime network traffic showing on the gui. The function retrieves the mb sent/received and total calculates the difference with the previous values and shows it on the gui every second.

The next function is `geolocation(packet)`. The function takes 'packet' as input calling the `summary()` method on it to get a summarized string representation of the packet. It then extracts the source and destination IP address from the 'packet' object (The packet is assumed to have a specific structure, where the source and destination IP addresses are stored at `packet[0][1].src` and `packet[0][1].dst`, respectively.). Then it uses the 'geoip_reader' object to perform geolocation lookup on the IP addresses. Calling the

Get() method to retrieve geolocation information stored in the variables `src_location` and `dst_location`. It checks the variables if they have 'None' values (if the geolocation of the IP addresses are not found) and creates string `src_info` and `dst_info` that represent the IP along with the country. The country name is accessed `src_location['country']['names']['en']` and `dst_location['country']['names']['en']`. It then inserts the packet summary, source and destination IP information into the GUI. The `geolocation()` function is being called for each captured packet by the `sniff()` function in the `capturePackets()` function shown later.

The function `capturePackets()` is for capturing network packets and applying to them the `geolocation()` function. It utilizes `sniff()` from `scapy` to capture the packets. The `prn` parameter specifies the callback function for each packet (`geolocation()` function). The `coun` indicates the number of packets to capture.

The function `clear()` is called when the button "CLEAR" is clicked on the GUI and clears the contents of the `packet_textbox` component. The `delete()` method called on the `packet_textbox` with arguments `1.0` and `tk.END`. This deletes text from first character (`1.0`) until the end of text (`tk.END`).

The final function `showDevices()` defines an inner function `update_device_list()` for updating the list of connected devices to the network and show them on the GUI. In the `update_device_list()` function the method `delete()` is called on the `device_listbox` to delete its current components and then makes an empty set "`connected_devices`" to store the devices addresses. The function iterates over the network interfaces and their associated addresses with the method "`psutil.net_if_addrs().items()`". For each address it checks if the family is `socket.AF_INET` which represents IPv4 addresses, if so it is added to `connected_devices`. When it has collected all devices addresses the function iterates over `connected_devices` and inserts each element into `device_listbox` with `insert()` method. It also schedules the `update_device_list()` function to be called every 5 seconds. It creates a separate GUI window.

Finally we create the GUI components and the function `update_info()` is called to start

updating the network information and the main event loop is started with the `mainloop()` method of the root window.

Assets used:

Code for monitoring function based on this tutorial : https://www.youtube.com/watch?v=O76lnYFvbTU&ab_channel=NeuralNine