Southeast Technological University

High Level Packet Sniffer

Functional Specification

Thassanai McCabe – C00250439
Supervisor: Paul Barry

Contents

Table of Figures

Abstract

This functional specification document describes the features of the high-level packet sniffer and how the project brief will be fulfilled. Interactions between users and the application are also described along with concept designs for key interfaces. It serves as a project plan and timeline to refer to during the development of the application. The High-Level Packet Sniffer aims to adapt the functionality of the packet sniffers available today to all users and not just those with technical knowledge of computing and networking.

1. Terms/Acronyms and Definitions

Term	Definition	Description
GUI	Graphical User Interface	The use of icons, menus, and other visual indicators allows a user to interact with an electronic device. (Stoltzfus, 2021)
TCP/IP	Transmission Control Protocol/Internet Protocol	TCP and IP are communication protocols that are used on computer networks. These protocols describe how data should be broken into packets, addressed, transmitted, routed, and received on a network. (Shacklett, 2021)
PCAP	Packet Capture	Packet Capture involves the interception of data packets travelling on a network. Packet data can be stored in .pcap files and as a result, can be analysed for network characteristics. (SolarWinds, 2023)
Packet	A small segment of a larger message.	Packets are used to send data over the internet. Packets can be defined by the protocol they use. They contain headers which describe important information such as packet source and destination. (Cloudflare, 2023)
UX	User Experience	Describes everything that influences a user's interaction with a product. Factors such as value, function, and usability are considered when designing the user experience. Best UX practices allow products to be accessible and functional. (Babich, 2020)
Flag	A command option	Flags are used to change the behaviour of a command. They can be combined to chain together multiple parameters/options when executing a command on a command line interface. (IBM, 2022)

2. Introduction

Packet sniffers are an application that allows users to examine the activity on their network in real-time. Using them has many security and troubleshooting benefits. It is common for packet sniffers to be designed with the assumption that the user of the application is familiar with concepts such as the TCP/IP model and .pcap files. As a result, non-technical users can be deterred from using packet sniffers because of the knowledge they require.

The purpose of the high-level packet sniffer is to allow any user and not just those with computing and networking knowledge to use and receive the benefits of a packet sniffer. The high-level packet sniffer aims to be as accessible and user-friendly as possible. The tool is planned to be presented as an executable application.

2.1. Purpose of the Document

Before beginning the development phase of the final year project, a specification document is produced to establish a foundation and develop a plan to follow during the development of the application. Details such as the functional requirements, appearance, and use cases are described. The scope of the overall project is defined using the functional specification document.

2.2. Project Scope

The high-level packet sniffer is targeted towards a non-technical audience. It should provide a user-friendly graphical user interface (GUI) that describes its functions clearly as opposed to a command line interface. Utilization of a centralised dashboard, as well as widgets, enforce this usability. Users should be able to monitor their network with little configuration and knowledge of networking required.

Non-technical users should be able to complete relatively complex tasks using the application. They should be able to identify aspects such as devices, destinations, vulnerabilities, network speed, and composition of their network data. The high-level packet sniffer could also see use in a learning environment where it is utilized as a teaching tool.

Functionality added to the high-level packet sniffer will be prioritized based on how useful that function might be to a non-technical user. The high-level packet sniffer does not focus on providing additional functionality that might not be found in existing packet sniffers and instead allows for non-technical users to use existing core functions of packet sniffers available today.

2.3. System Overview

It is planned to develop the application using optimal UX methods. This should allow for the adaptation of existing packet sniffer functions to non-technical users as well as learner users. It is planned to build this application to be compatible with Microsoft Windows 10 as an executable.

Development for Microsoft Windows 10 was chosen because as (Taylor, 2022) states, Windows operating systems account for 70.68% share in the OS market as of August 2022. Furthermore, Microsoft Windows is found in both home and enterprise environments. These two environments could benefit from the high-level packet sniffer.

After a user executes the high-level packet sniffer on their network, it should provide useful telemetry to the user that should allow them to troubleshoot/secure their network. The identification of devices on the network along with their bandwidth usage should facilitate this troubleshooting. Furthermore, the identification of new devices and potentially malicious traffic should facilitate network security. The high-level packet sniffer acts as an intermediary device between a user and its network. Telemetry is planned to be delivered to users using reports generated by the application.

2.4. Risks and Assumptions

Assumptions

1. Users of the high-level packet sniffer have the authorization to analyse the network in which it is deployed.
2. Users of the high-level packet sniffer are using it for ethical and non-malicious reasons.
3. Users are connected to a network through Wi-Fi or Ethernet cable.
4. I have access to libraries that allow for the creation of a GUI and packet sniffing capabilities.

Risks

1. Completing the development of the application within the given timeframe.
2. Users utilize the high-level packet sniffer for malicious purposes.
3. The user has difficulty using the application.

Risk Mitigation

1. Mitigation to Risk 1: Avoid scope creep to increase the chance of project completion within the timeframe and follow the specification document carefully.
2. Mitigation to Risk 2: Disclaimer displayed to users of the high-level packet sniffer that describes laws, penalties, and ethics associated with the use of a packet sniffer.
3. Mitigation to Risk 3: A focus on optimal UX can decrease the chance that users have difficulty using the application. The user interface should be clear and understandable for any user. Furthermore, instructions and information should be clearly described.

3. Functional Requirements

3.1. Core Requirements

1.	**Functional Requirements**
1.1.	Allow users to apply filters before and during packet capture.
1.2.	Allow users to generate graphs using analysis statistics.
1.3.	Allow users to manually start and stop packet capture.
1.4.	Allow users to open existing .pcap files.
1.5.	Allow users to save capture data to .pcap files.
1.6.	Allow users to select a network interface to begin capturing packets on.
1.7.	Allow users to select all interfaces to begin capturing packets on.
1.8.	Allow users to set a specific duration for packet capture.
1.9.	Automatically detect devices on a network and label them accordingly for the user.
1.10.	Automatically detect protocols on a network and label them accordingly for the user.
1.11.	Automatically identify potentially malicious traffic on the network.
1.12.	DNS Resolution.
1.13.	Identify all network interfaces on the current device.
1.14.	Identify incompatible packet capture files.
1.15.	Measure network speed using upload and download data.
1.16.	Provide the user with a graphical user interface (GUI).
1.17.	Generate reports for the user.

3.2. Other Requirements

These requirements are considered non-essential and are of lesser priority than the functional requirements listed above.

2.	**Other Requirements**
2.1.	Allow the application to run on UNIX systems.
2.2.	Minimize system resource usage.
2.3.	Automatically save data upon a loss of connection to the network.
2.4.	Remember recently opened/saved files by the user.
2.5.	Allow for the customization of colours used to represent packets, devices, and protocols.
2.6.	Allow users to save and load packet capture filters.

4. Use Cases

This section describes various interactions that a user may have with the application.

ID	**UC 1**
Title	Initiate Packet Capture
Description	The user begins capturing data on one or all interfaces on their device
Primary Actor	User
Preconditions	The user has selected which interface they want to utilize for packet capture
Postconditions	The application is analysing network activity on the correct interface
Success Scenario	Users can view network activity in real-time
Use Frequency	Often

ID	**UC 2**
Title	Save data to a packet capture file.
Description	Save data that was captured to a .pcap file. The user can enter the name of the packet capture file upon saving.
Primary Actor	User
Preconditions	The user has captured data on the network and network analysis was successful.
Postconditions	The application is not currently capturing data.
Success Scenario	A packet capture file is created with the name provided by the user.
Use Frequency	Often

ID	**UC 3**
Title	Open data from an existing packet capture file.
Description	User selects a file from their file explorer to open within the application.
Primary Actor	User
Preconditions	The user supplies a compatible packet capture file.
Postconditions	The application can read data from the provided packet capture file.
Success Scenario	The data on the packet capture file is successfully displayed to the user.
Use Frequency	Often

ID	**UC 4**
Title	Generate report using pack capture data.
Description	The report is generated and saved as a file for the user to open. This report should allow users to quickly discover high-level information about their network analyses.
Primary Actor	User
Preconditions	Sufficient network data has been captured.
Postconditions	Report data is written to a created file.
Success Scenario	User has a report file that they can use to analyse packet capture data.
Use Frequency	Occasionally

ID	**UC 5**
Title	Filter packets.
Description	Filter the packets being displayed to the user during network analyses.
Primary Actor	User
Preconditions	The application is analysing network data.
Postconditions	Data being displayed by the application is only data which matches the specified criteria.
Success Scenario	The user only sees data that matches specified criteria during network analyses.
Use Frequency	Often

ID	**UC 6**
Title	Measure network speed
Description	The upload and download speed of the device running the application is tested and the results are displayed to the user.
Primary Actor	User
Preconditions	The application has successfully established a connection with the internet.
Postconditions	Results of the network speed test are displayed.
Success Scenario	The test is successful, and the results of the test are accurate and available to the user for viewing.
Use Frequency	Occasionally

ID	**UC 7**
Title	Discover Devices on the Network
Description	The application discovers the MAC addresses and manufacturer names of the devices visible on the network.
Primary Actor	User
Preconditions	The application has successfully determined the network address of the current device.
Postconditions	The results of the device discovery sweep are displayed to the user.
Success Scenario	Device discovery is successful with MAC addresses and attempted resolved names are displayed for each device.
Use Frequency	Occasionally

5. Packet Capture File Parsing

Packet capture files can be parsed for information such as protocol types, protocol frequencies, IP addresses, and IP address occurrence frequencies.

A function “report_top_ten()” should extract the top 10 most common IP addresses in a packet capture file. If there are less than 10 IP addresses in the packet capture file, report only those present.

A function should be engineered to extract the main protocols from the packet capture file and place their occurrences into a Python counter.

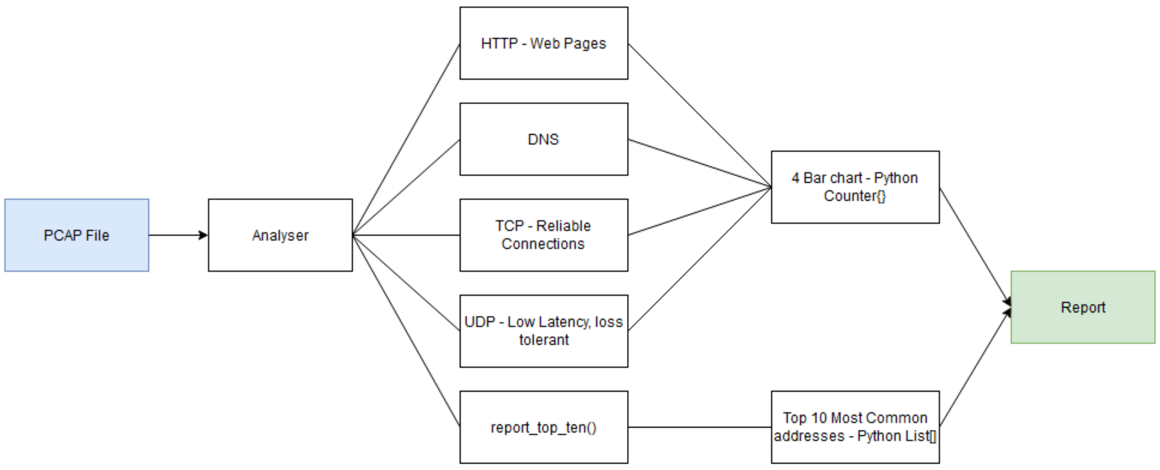

PCAP File
Analyser
HTTP - Web Pages
DNS
TCP - Reliable Connections
UDP - Low Latency, loss tolerant
report_top_ten()
4 Bar chart - Python Counter{}
Top 10 Most Common addresses - Python List[]
Report

Figure 1 Packet Capture File Parsing Plan

Users input existing .pcap files, which are analysed and parsed. Consequently, Python objects such as lists and counters are created to carry out calculations and parsing to build a report.

6. Reporting

As shown in Figure one, part of the reporting system involves analysing a packet capture file and parsing it for key information. This information is placed into Python objects so that calculations and storage can occur.

As functions are carried out on the application, the results of these functions should be stored in such a way that their results can be consolidated into a readable report.

7. Packet Capture Library – Scapy

Scapy is the core library that will allow me to capture, create, and analyse packets on the network. As stated by (Rosenbaum, 2022), using Scapy allows developers to send, sniff, and dissect network frames, allowing you to create networking applications. Furthermore, he explains how Scapy sees packets as objects, with several layers. Frames can be created with the layers you wish to use.

Additionally, Scapy is capable of VLAN hopping + ARP cache poisoning and VOIP decoding. Scapy's two main tasks are receiving responses and sending packets. Developers can specify a collection of packets, and Scapy sends them. It then matches those requests with responses, then delivers a list of packet pairs (request, answer), as well as a list of packets that could not be matched. This has benefits over tools such as Nmap in that the entire packet is returned as an answer rather than just an open/closed/filtered response. Furthermore, functions that perform traceroutes, port scans, and pings which return a list of responding endpoints can be created (Biondi, 2023).

8. Interface Requirements

This section describes high-level concepts and ideas of key interfaces that a user will be presented with so that they can interact with the high-level packet sniffer.

8.1. Home

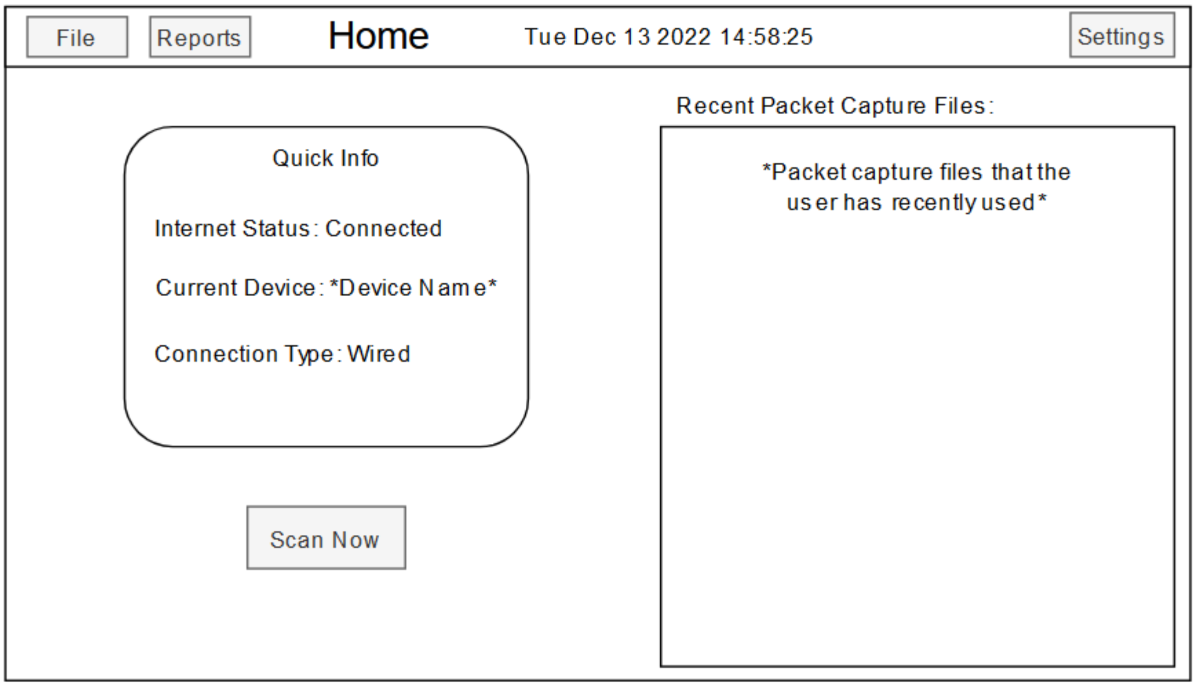
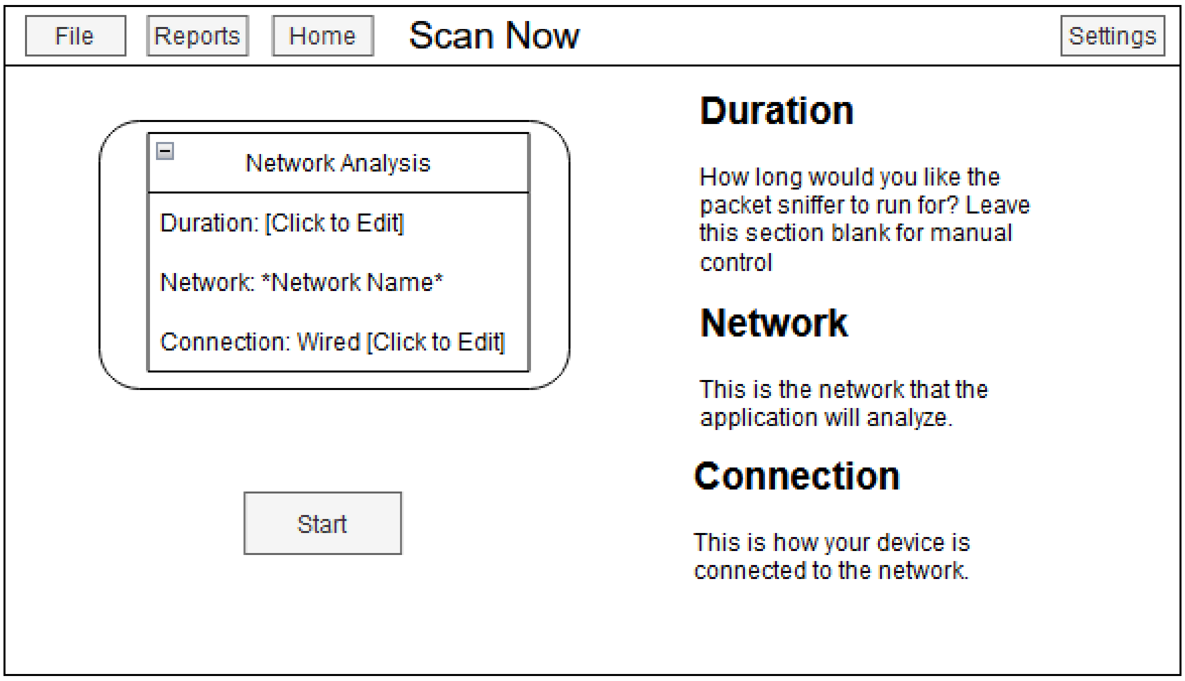

File
Reports
Home
Tue Dec 13 2022 14:58:25
Settings
Recent Packet Capture Files:
Packet capture files that the user has recently used
Quick Info
Internet Status: Connected
Current Device: *Device Name*
Connection Type: Wired
Scan Now

Figure 2 High-level packet sniffer landing page

This is the first interface that the user should see when first starting the application. The "Quick Info" section should gather key networking information about the current device. In the example shown, an ethernet connection to the internet has been discovered. The application has listed this connection as "Wired" to accommodate non-technical users. Similarly, Wi-Fi connections should be listed as "Wireless" for the user.

Users can also see “Recent Files” on the right-hand side of this interface. Files displayed here will only be those which can be opened using the packet sniffer. The “Scan Now” button allows the user to commence network analysis.

8.2. Pre-Scan



Duration: [Click to Edit]

Network: *Network Name*

Connection: Wired [Click to Edit]

Duration

How long would you like the packet sniffer to run for? Leave this section blank for manual control

Network

This is the network that the application will analyze.

Connection

This is how your device is connected to the network.

Figure 3 "Scan Now" Interface

After the user selects the "Scan Now" button on the home interface, this should be the interface that users see. The user should have the option to configure the duration of the network analysis here. It can also be unconfigured for manual stopping and starting.

Information about the network that will be analysed as well as the interface used is displayed here. Clicking the “start” button should commence network analysis.

8.3. Network Sniffing

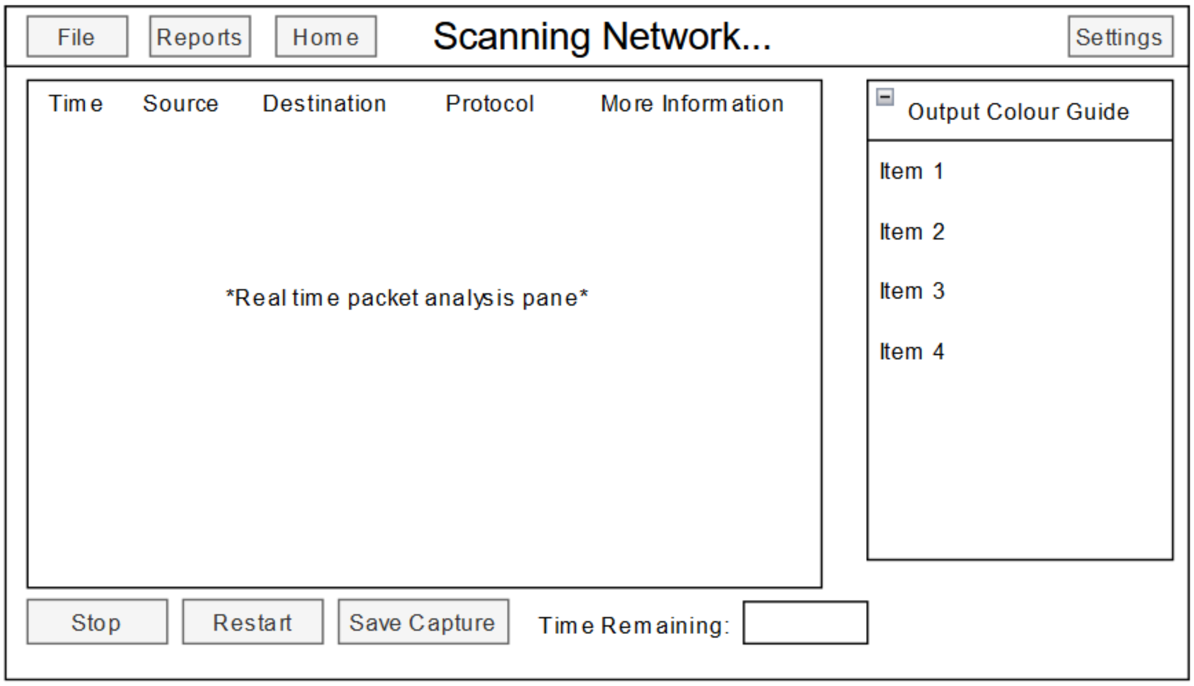


Figure 4 Network Analysis Interface

The network analysis interface should display network data as it is captured in real-time. The "Real-time packet analysis pane" displays a self-updating list of packets with details in their respective columns which are time, source, destination, protocol, and more information.

Different types of traffic captured should have specific colours associated with them. Users can refer to the section on the right with the heading “Output Colour Guide”, where they can match colours with their respective representatives.

There should be buttons at the bottom of the interface that allows the user to control the analysis. They should be able to stop and restart the analysis as well as save the captured network data.

8.4. Post Scan

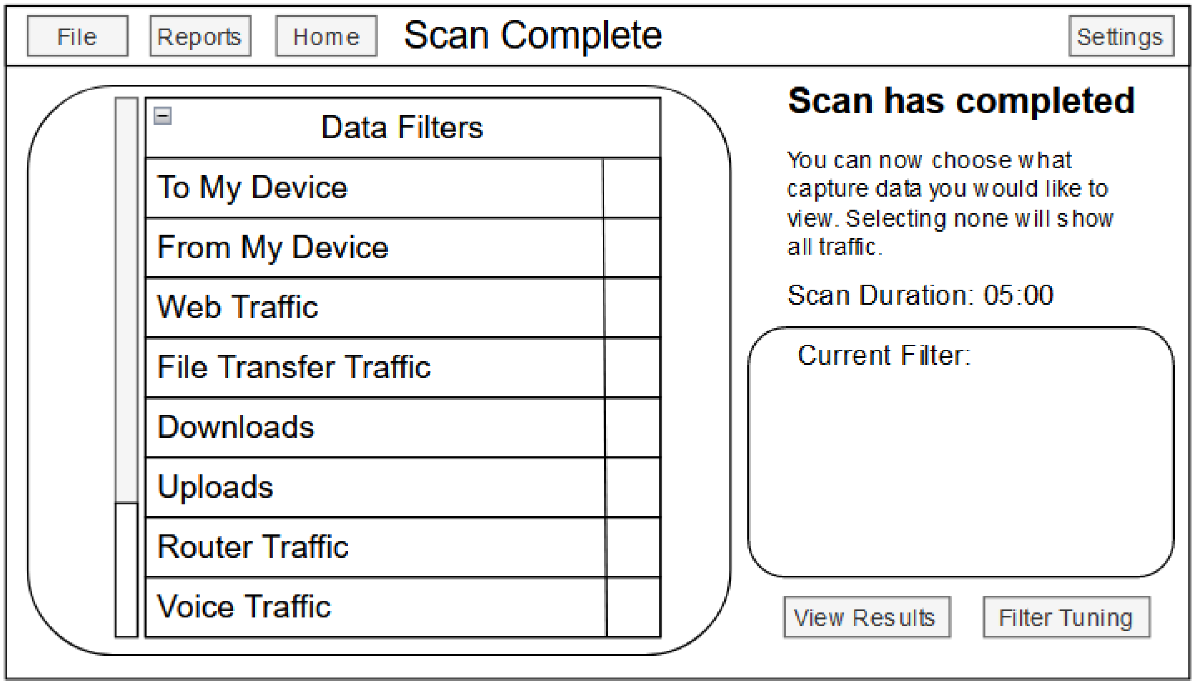


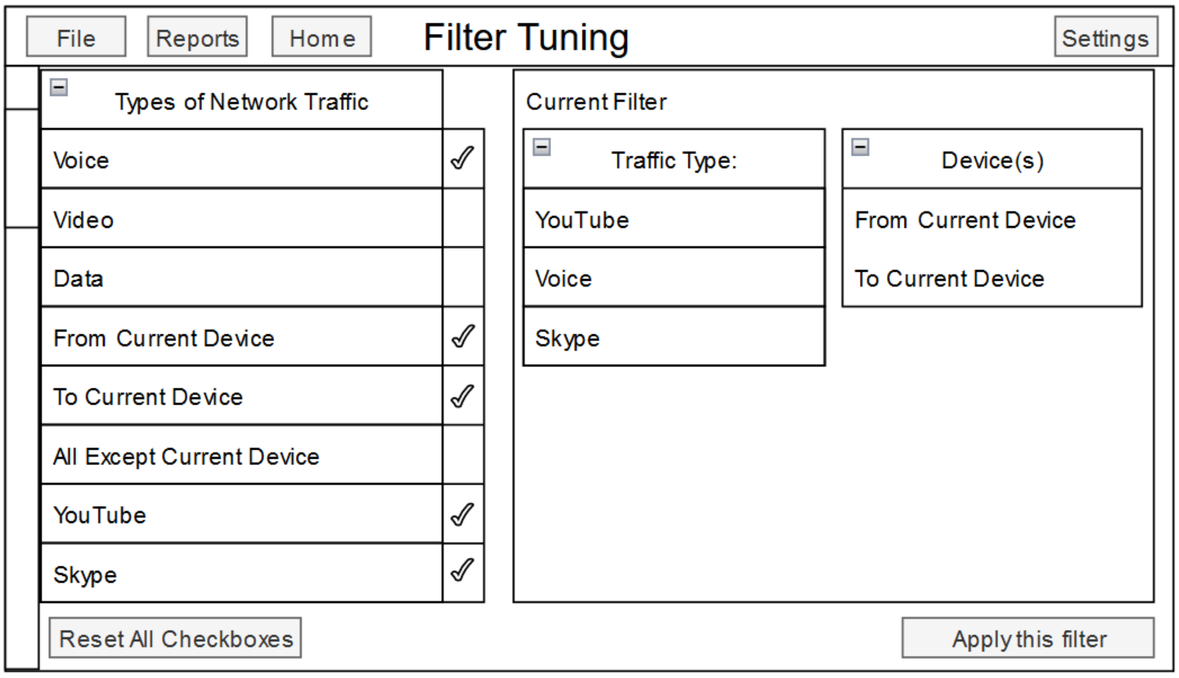
Figure 5 Interface which is displayed after network analysis has been completed.

At this point, network analysis should be completed.

After network analysis is complete, the user should be presented with the option to filter the packet capture data and view only what they need to. The "Data Filters" menu should have key network traffic types that users can select. When selecting these filters, the “Current Filter” section should dynamically display what filters are going to be applied based on the user’s selection.

As checkboxes cannot accommodate all use cases, should users require, they can fine-tune filters in the "Filter Tuning" menu which is accessible from this menu. The checkboxes aim to provide most non-technical users with sufficient options.

8.5. Filter Tuning



Types of Network Traffic	
Voice	✓
Video	
Data	
From Current Device	✓
To Current Device	✓
All Except Current Device	
YouTube	✓
Skype	✓

Current Filter

Traffic Type:	Device(s)
YouTube	From Current Device
Voice	To Current Device
Skype	

Figure 6 Creating a network analysis filter.

Before viewing network data, users can choose specific traffic types and devices to single out from the rest of the data. Checkboxes are used to select and deselect different parameters. This is designed to be easier to understand than the existing “flags” method used in for example tcpdump.

9. FURPS

FURPS, which stands for functionality, usability, reliability, performance, and supportability, is used in requirement management. It aids in the completion of requirements without neglecting non-functional needs and expectations that end-users might have. (Gekht, 2020).

- **Functionality:** This heading discusses how and why a user would use the application. The application should serve a purpose and solve a problem (Gekht, 2020). In the case of the high-level packet sniffer, it should focus on usability and accessibility. The functionality of the high-level packet sniffer is described in the functional specification document.
- **Usability:** This is a key factor within the high-level packet sniffer. It describes the applications' suitability for use by users. It is important to differentiate the users of the application as well as their environment from the developer and the developer's environment. While developers may be able to navigate an application with ease, it is important that end users can do the same. (Gekht, 2020).
  - As described by (Gekht, 2020), the operating conditions to consider include the following:
  - Physical, mental, educational, and social characteristics.
  - Environmental conditions, what would the user be doing when using the application?
  - Process conditions, distracting factors, price of error.
  - Ease of use for all users is a key factor that will be considered during the development of the application.

- **Reliability**: This describes the ability of the application to withstand unexpected or malicious user behaviour. Furthermore, it describes the ability of the application to withstand 3rd party product/equipment failures. One way reliability can be enforced is by using error detection and handling. How fast a system recovers after a crash also describes its reliability. (Gekht, 2020). Reliability can be enforced during the development of the application using various tests such as load testing and limit testing.

- **Performance**: (Gekht, 2020) describes performance to be about how fast the system should answer and how much system resources, such as processor time, memory, and network traffic the system is allowed to use. The high-level packet sniffer should be optimised to consume as few system resources as possible. This allows for it to be used by a wide range of users who in some cases may have low-powered machines, with an example being a budget laptop.

- **Supportability**: This describes how convenient it would be to modify, extend, and support the application in the future and how it is configured and installed by users (Gekht, 2020). The application being presented as an executable minimises the amount of configuration that needs to be done by users. The use of established programming practices as well as precisely documented code would allow for the application to be modified/extended with ease by any developer who wishes to do so.

10. Technologies

Technologies required in the development and execution of the application.

10.1. Npcap

Installed on Microsoft Windows 10 Systems to allow for network packet capture and processing.

10.2. Python

High-level language with adequate libraries for the development of the application.

10.3. Scapy

Core library of the application, used to identify network interfaces, manipulate, capture, and analyse packets.

10.4. Tkinter

Used to engineer a graphical user interface front end. Allows for the use of widgets such as buttons, text fields, and labels.

11. Development Environment

11.1. Visual Studio Code

Main development environment. Visual Studio code has been modified to allow for Python development and code execution. PIP is used to install required libraries.

11.2. Visual Studio

Used to test different libraries for various programming languages.

11.3. Jupyter Notebook

Used to test code snippets for various libraries and functions.

11.4. Git Version Control

Used to create branches and commits as well as cloud backups for the code. The use of GitHub allows for easy swapping between development on various devices.

12. System Requirements

Technologies required to execute the application.

NAME	VERSION	DESCRIPTION
PYTHON	3.10.7	Application language
NPCAP	1.71	Windows packet capture library
MICROSOFT WINDOWS	Windows 10	The operating system of devices running the application
NETWORK INTERFACE		Ethernet/Wi-Fi Compatible interface

13. References

Taylor, P. (2022), *Global market share held by computer operating systems 2012-2022, by month* [Online] Available at: https://www.statista.com/statistics/268237/global-market-share-held-by-operating-systems-since-2009/ [Accessed 5 JAN 2023]

Cloudflare, (2023), *What is a packet? | Network packet definition* [Online] Available at: https://www.cloudflare.com/learning/network-layer/what-is-a-packet/ [Accessed 5 JAN 2023]

SolarWinds, (2023), *What is Packet Capture (PCAP)?* [Online] Available at: https://www.solarwinds.com/resources/it-glossary/pcap [Accessed 5 JAN 2023]

Stoltzfus, J. (2021), *Graphical User Interface (GUI)* [Online] Available at: https://www.techopedia.com/definition/5435/graphical-user-interface-gui [Accessed 5 JAN 2023]

Shacklett, M. (2021), *What is TCP/IP* [Online] Available at: https://www.techtarget.com/searchnetworking/definition/TCP-IP [Accessed 5 JAN 2023]

Babich, N. (2020), *What You Should Know About User Experience Design* [Online] Available at: https://xd.adobe.com/ideas/career-tips/what-is-ux-design/ [Accessed 5 JAN 2023]

IBM, (2022), *Command flags* [Online] Available at: https://www.ibm.com/docs/en/aix/7.1?topic=names-command-flags [Accessed 5 JAN 2023]

Rosenbaum, O. (2022). *How to Use Scapy – Python Networking Tool Explained*. [online] freeCodeCamp.org. Available at: https://www.freecodecamp.org/news/how-to-use-scapy-python-networking/ [Accessed 14 Apr. 2023].

Biondi, P. (2023). *Introduction — Scapy 2.4.4 documentation*. [online] scapy.readthedocs.io. Available at: https://scapy.readthedocs.io/en/latest/introduction.html#about-scapy [Accessed 14 Apr. 2023].