Network Health Monitoring App

Identify risks to your network quickly with an intuitive GUI-based App that scans your network and highlights any security concerns.

AC2

Sarah Gardiner*, [Redacted], [Redacted],

[Redacted], [Redacted], [Redacted], and [Redacted]

CMP311: Professional Project Development & Delivery

BSc Ethical Hacking

2022/23

Note that Information contained in this document is for educational purposes only.

**My section has been highlighted in blue*

Team Member	**Section**	Technical Contribution (list contributed work, e.g., coded, designed, developed, investigated sections etc.)
[Redacted]	Executive Summary	Designed, researched, investigated. Coded some functions, debugged.
[Redacted]	Introduction	Coding, debugging, network setup and network testing, team management, researching.
[Redacted] Sarah Gardiner	Method	Dylan: Coding, network setup, research, and testing. Sarah: Coding and creation of the user interface, alongside research and testing of the UI.
[Redacted]	Results	Coding, NMAP scanning implementation, research and testing.
[Redacted]	Discussion	Coded, debugged, researched, investigated, designed, tested, checked project progress with quality baseline, engaged in team discussions, aided other members when there was nothing to do.
[Redacted]	User Manual	Network scanning, specifically the route tracing code Classification of device type and vulnerability status code Led technical reviews Physical network setup

Executive Summary

The business world relies on computer networks to stay connected to their customers and share resources across distance. This keeps down costs and offers flexibility in delivering what customers want. To do this effectively businesses hold a lot of data that is valuable. As a result, this makes it susceptible to attack from criminals wanting to disrupt business or steal this information. Successfully defending against cyber-attack ensures business continuity and the trust of their clients.

One way to address this problem is with the use of network monitoring tools.

As part of a strategy to foster learning and teach real world skills. Abertay creates a scenario, sometimes in collaboration with people in industry, for students to develop a solution to.

This project was carried out by AC2 a group of 3rd year ethical hacking students at Abertay University. Ross Heenan a teaching fellow at the same university acted as our client.

The brief

- To develop a network health monitoring app that would display the data visually.
- Any device receiving or transmitting abnormal data would be highlighted as potentially compromised.
- Open ports that constituted a concern would be flagged.
- Network map would be created.
- Each discovered device to be classified and identified.

Benefits for the client.

- Being able to monitor a network in real time and have that data shown visually lets you pinpoint areas of concern that can then be promptly addressed.
- Can be used as learning tool for the university and students. And teach them about networking and security and the possibilities that exist in network monitoring.
- Education and outreach are a major tool in the fight against cyber criminals.
- There is the chance to highlight the importance of cyber security.
- By making this a part of the wider conversation you are advertising the university and building a more cohesive community.

Outside of the university this tool has a major value for a variety of users.

- For non-technical users that might struggle to understand the technical aspects of cyber security this is an intuitive GUI based tool. Its simple interface doesn't ask many questions and all the work goes on behind the scenes.
- For security professionals it is customisable and scalable and way to quickly ascertain the health of a network.

How we delivered

Using python allows access to a range of libraries that are used in cyber security and networking. It is versatile and easy to adapt to our needs. PyQt5 is a python GUI framework that makes use of QT designer, a set of development tools, targeted for networking needs, and C++ libraries.

It also has the advantage of coming inbuilt with matplotlib which can be used to display the network traffic.

There is a drag and drop facility which makes it simpler and more streamlined to come up with pleasing designs and test what they look like.

Once happy with the designs Ui files were generated. These could then be converted into python files that could seamlessly be incorporated into the rest of the program. Custom made python functions were made to produce the network map, capture traffic, and run the Nmap scan that would parse the required information. These functions were then attached to buttons in the GUI.

An evolutionary prototype model was used to ensure our client was kept informed of the progress and milestones met. See figure 1 for how we progressed from using simple widgets to take input to the more complicated network map window.

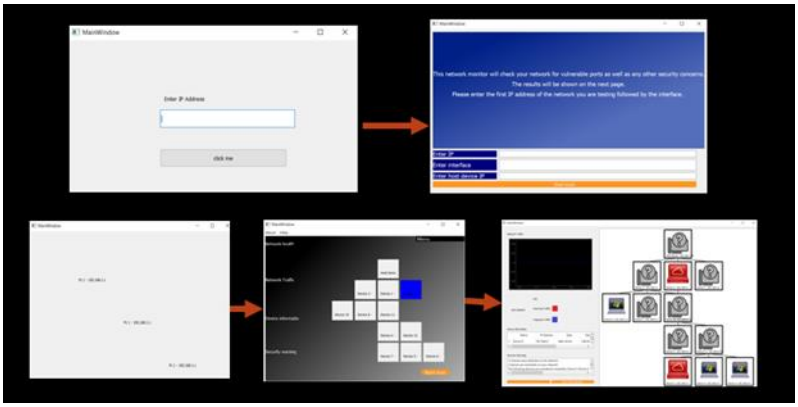


Figure 1 evolution of the app

The results

Simple interface only requires the interface that you are attached to with the choice of using it against a single IP or an IP address range. See figure 2 for the user-friendly design.

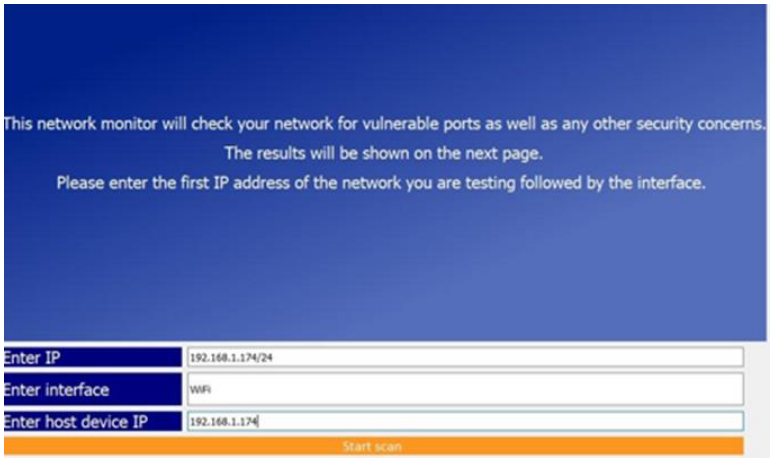


Figure 2 input screen

Once these have been put in you are taken to the main screen where you will be presented with the network map of the scanned network. The information gathered during this scan will then populate the network map complete with ip addresses and any identified security concerns. The device will show red if identified as being compromised.

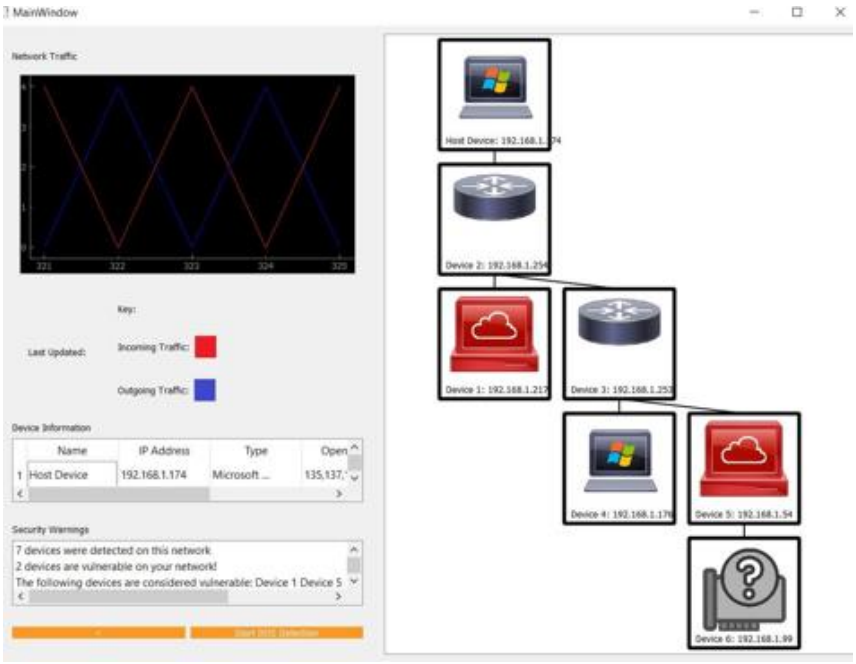


Figure 1 main network window

Clicking on each device on the network map will allow you to view information for that device. As seen in figure 4 you will be given a rundown of the traffic from the device, which ports are open and the services running on them along with specific device information.

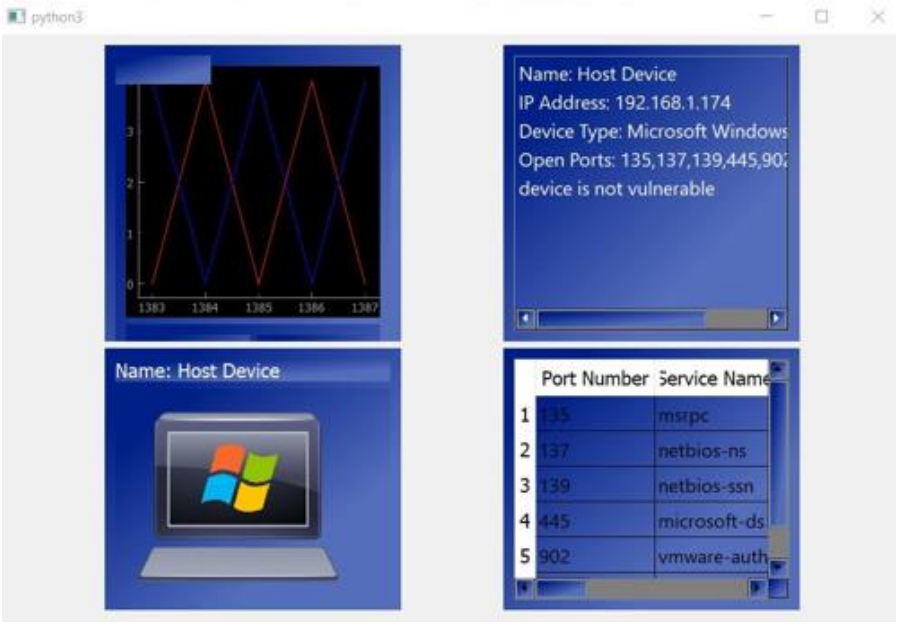


Figure 2 closer view of information generated.

Conclusion

All the aims were met in this project. The app that we developed uses a combination of custom-made functions and PyQt5 generated files to produce an app that will scan a network. It will classify and identify the discovered devices and display any security concerns if the device appears to be sending/transmitting data that is flagged as being abnormal. Further security information will be displayed concerning open ports along with detailed device information. Armed with this information, security professionals get an accurate picture of the network and any potential concerns that need addressing.

Time constraints halted further development but there is so much more that could be done. Adjusting the flags used in the Nmap script can get more specific information about possible vulnerabilities. Making use of an IDS to get a more accurate picture of the traffic going through the network rather than the throughput of pings used to simulate a DDOS attack.

But that's why the choice to use python was good. Combined with a modular approach to the program gives the developers a degree of flexibility in customising the program to the needs of the client.

Table of Contents

1 INTRODUCTION

1.1 BACKGROUND

The field of computer networking is critical to the success of any modern businesses. As more and more businesses become reliant on technology to operate, it is essential to have a secure and efficient network to facilitate communication, data transfer, and other essential functions. However, network administrators face several challenges in maintaining the health and security of their networks. This is an issue as 39% of UK businesses identified a cyber-attack within 2022 (GOV.UK, 2022). One of the biggest challenges is identifying and addressing potential vulnerabilities before they can be exploited. This is where our network health monitoring app comes in, offering a solution to the ongoing problem of network vulnerability management. In this report, we will provide a detailed explanation to the development of our network health monitoring app, including the business problem it aims to solve, the functionally our app provides, and the limitations of our solutions.

To properly understand the significance and potential impact of our network health monitoring app, it's essential to first have a clear understanding of the current technology and design used for network monitoring. The current technology for network monitoring typically involves the use of network analyzers or sniffers that capture and analyze network traffic. Network analyzers are powerful tools that can provide network administrators with an in-depth view of the network's performance, including packet capture, flow analysis, and protocol analysis. They are useful for detecting anomalies in network traffic and identifying potential security threats (Froehlich, 2021). However, network analyzers can be complex and challenging to use, requiring extensive technical knowledge to interpret and analyze the data effectively. Additionally, network analyzers may not provide a clear visual representation of the network, which can make it challenging to quickly identify potential issues. Despite these limitations, network analyzers remain a crucial tool for network administrators, and our network health monitoring app aims to build upon this technology by providing a more intuitive and user-friendly way to monitor and manage network health.

Despite the availability of many network health monitoring tools, network administrators still face several challenges in maintaining the health and security of their networks. Existing tools are often limited in their functionality and fail to provide a comprehensive view of the network and all its connected devices. Additionally, they can be difficult to use and require extensive training, making them inaccessible to smaller businesses or those without dedicated IT teams. Furthermore, these tools often produce a large number of false positives, which can lead to alert fatigue and desensitization to genuine security threats. Our network health monitoring app aims to address these challenges by offering an intuitive and easy-to-use solution that provides a comprehensive view of the network's health and empowers network administrators to proactively identify and address potential vulnerabilities before they become significant problems.

Our client for this project is Ross Henan, a lecturer at Abertay University with expertise in computer networking. Mr. Henan has years of experience working with networks, and he understands the importance of having a reliable and secure network for businesses to operate. As a lecturer, Mr. Henan is also interested in exploring new technologies and solutions that can improve the efficiency and security of networks. He has approached us to develop a network health monitoring app that can help network administrators identify and address potential vulnerabilities in their networks before they can be exploited. Our app aims to provide Mr. Henan and Abertay University with a more efficient and effective tool for network management, allowing them to better support their students and the wider university community.

The brief for the developed application is focused on network health monitoring, specifically on visualizing and classifying devices on the network. Our application can distinguish between clients, servers, and network devices from the analyzed network traffic. The key goal of the application is to identify potentially compromised or vulnerable systems on the network. This is achieved by highlighting clients that are sending or requesting abnormal data and clients with vulnerable application ports open on the network map. This highlights the importance of identifying and mitigating potential security risks on a computer network, which is crucial for any organization to ensure the security and reliability of their network [Appendix A].

The business problem that our application solves is the need for a reliable and efficient way to monitor the health of computer networks. The network administrators responsible for managing the network need to be able to quickly identify potential security risks, compromised systems, and other issues that could impact the performance and stability of the network. Without a proper network health monitoring tool, these administrators may not be able to detect these issues until it's too late, resulting in network downtime, lost productivity, and increased costs. Our app provides an intuitive and easy-to-use solution that allows network administrators to quickly visualize and analyze the network's health, enabling them to proactively identify and address potential issues before they become significant problems. By providing a comprehensive view of the network and all its connected devices, our app empowers network administrators to take control of their network's health and improve its overall performance and security.

1.2 AIM

The aims for our network health monitoring app includes:

- Visualization of the network to provide a clear understanding of all connected devices and their activity.
- Classify devices on the network as clients, servers, or network devices to facilitate vulnerability management.
- Highlight potentially compromised systems by detecting abnormal data requests or transfers between devices.
- Identify potentially vulnerable systems by detecting open application ports and highlighting them for further review.

2 METHOD

The methodology used to create the application split into two sections, creation of the user interface (UI) and the backend. This methodology will outline the steps taken in each area of the application.

2.1 USER INTERFACE

The creation of the application's UI involved following the prototyping process model, as it allowed the team to create and implement design ideas for the client to actively use and review. The lifecycle of the chosen prototyping method involved four main steps:

1. Developing/refining the program's design and architecture.
2. Implementing the designs.
3. Testing and evaluating the program.
4. Presenting the program to the client.

These steps were adapted from those detailed in the book Prototype to Product, written by Alan Cohen, and were repeated for each new prototype of the UI. The following section of the methodology will outline the actions taken during each step.

2.1.1 Development of designs and architecture

2.1.1.1 Low-Fidelity Prototyping

Development of the first prototype depended on extracting key points from the client's brief and creating a low-fidelity mockup of the potential layout of the final product. This first prototype was a wireframe, and creating it allowed for the team to produce conceptual assumptions made from points within the client brief (Jonathan Arnowitz, 2010).

Microsoft Paint was used, as it allowed for creating basic shapes used to represent key areas of the UI, alongside enabling the inclusion of images and text to represent possible results from the backend algorithms. Figure 2.1 and Figure 2.2 show the two windows from the first prototype, where black text was used to represent the on-screen text that a user could read and interact with, while blue text was used to describe further actions or data that could be presented in those areas in future prototypes.

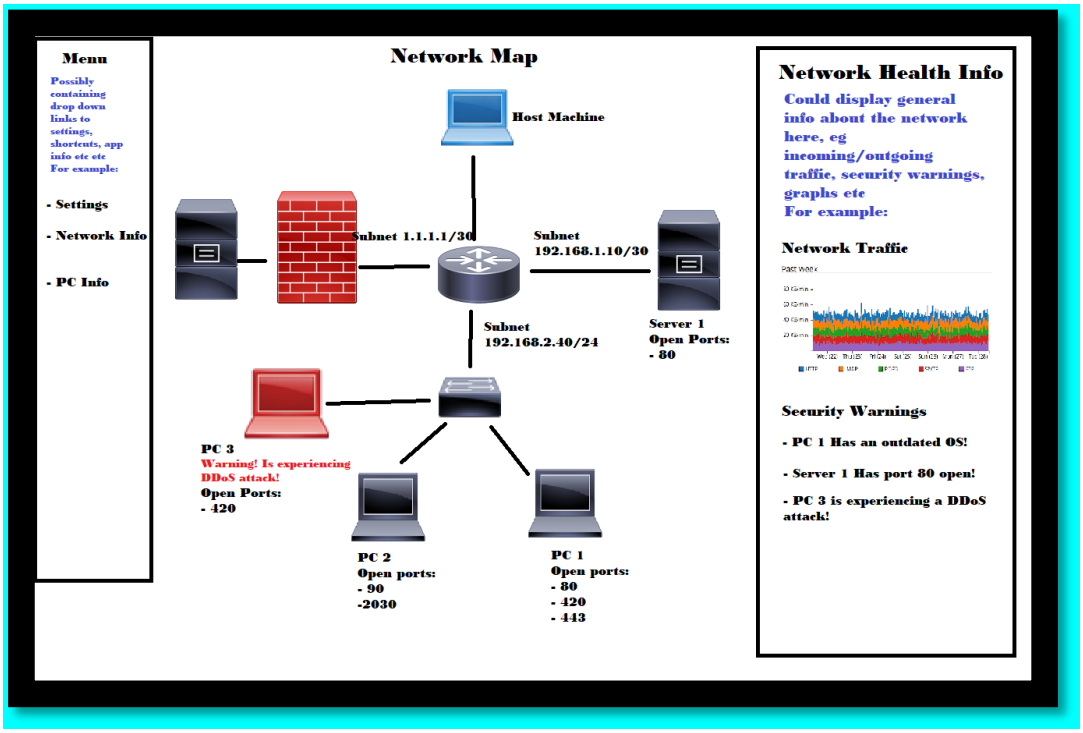


Figure 2.1: A prototype of what the potential user interface could look like, created using Microsoft Paint.

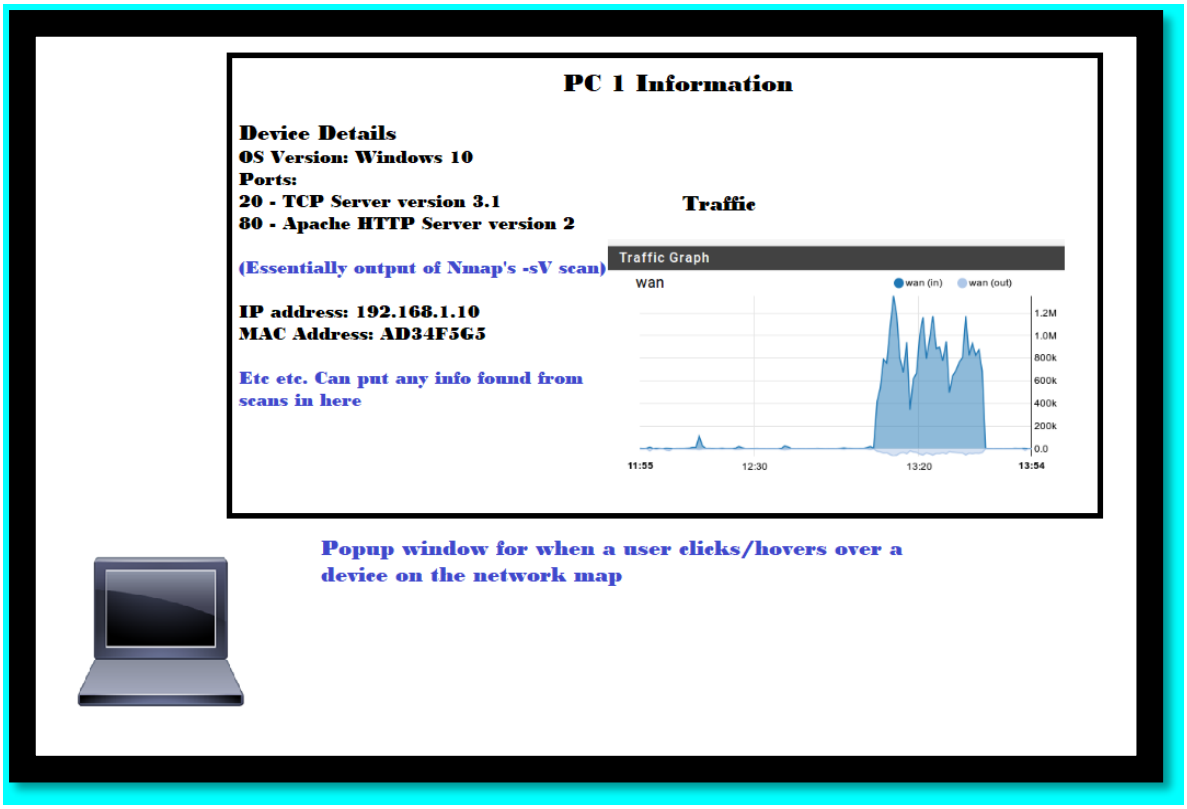


Figure 2.2: A second window from the first prototype, representing the information a user might see upon clicking a device.

2.1.1.2 High-Fidelity Prototyping

The models created later in the process were high-fidelity prototypes and were created using the program Qt Designer. Built to work with the main library used for handling the UI elements in this application, PyQt, Qt Designer allowed for the basic elements of the UI to be placed in a drag-and-drop manner. This type of UI design creation was chosen as it was much faster than the only other alternative method, which involved each element being manually coded in. In particular, the designer allowed for any created windows to be converted into Python code, which let the UI team to add in code created especially for the program. Figure 2.3**Error! Reference source not found.** shows an example of the code generated by Qt Designer.

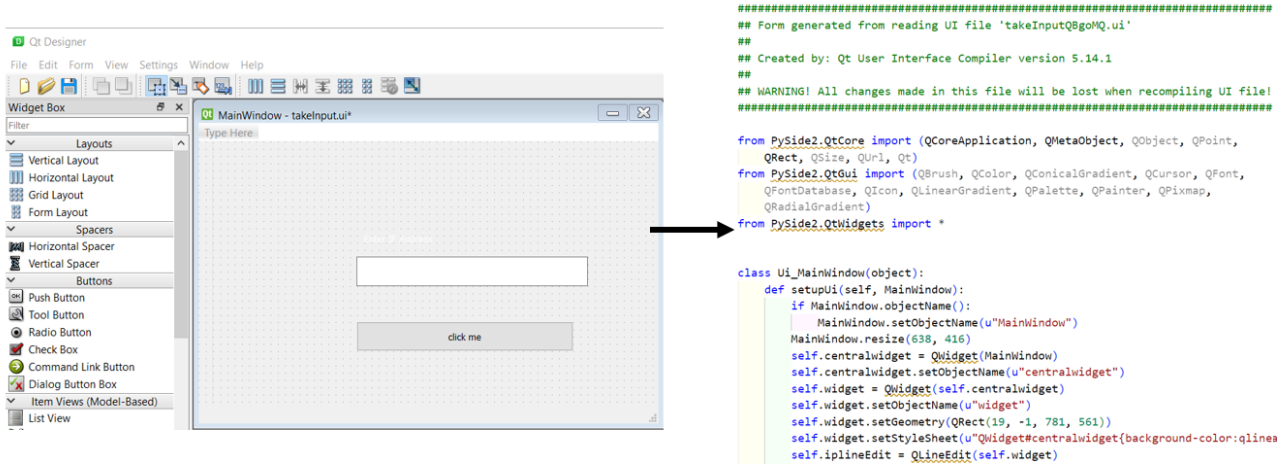


Figure 2.3: An early prototype of the input page, with its corresponding Python code, as generated by Qt Designer.

As the project progressed, each prototype created had more features than the one previous. Originally, just the necessary buttons and text were added to test the UI's functionality and compatibility with the backend network scanning algorithms, while later prototypes had better designs and features to allow for a better user experience. Figure 2.4 shows the progression of the network map as an example of this, with the first window containing just IP addresses in a cascading order, while the second shows the network after the device placement algorithm had been finalized. The third window shows the screen of the final application, with the network map being fully functional with images of the devices and lines between each connected device.

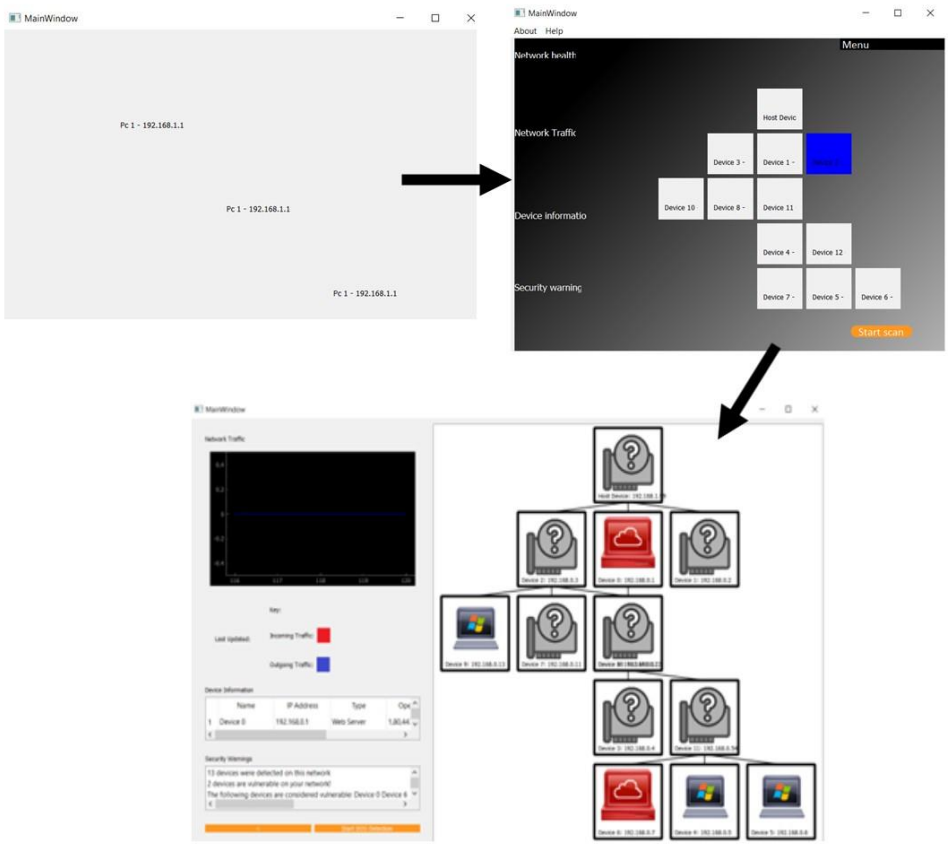


Figure 2.4: The progression of the network map throughout the project's lifecycle.

2.1.2 Design Implementation

As discussed previously, Python was used to implement the coded side of the UI. It was decided to use object orientated programming (OOP) for this code, as it would allow for data and features added in previous prototypes to be easily used in new additions to the application, as well as keeping the code in an organized form that could be easily used by all members of the team. On top of using OOP, the singleton design pattern was followed, which meant that the program would have only one main object of a class created during runtime. The team felt this OOP pattern would fit well with the application's intended behavior, as the main class, named "cNetwork" would be used to essentially log all data coming in from the backend, which fits with the typical use case for singleton classes (Phillips, Giridhar, & Kasampalis, 2016). The main classes for the program were decided upon during the planning phase of the project, however additional classes were added later as new features were implemented. After each prototype was presented to the client, the project team would make necessary changes or additions to the existing code to implement feedback from the client.

2.1.3 Testing

2.1.3.1 Using Placeholder Data

Running the backend algorithms was always feasible during creation of the UI, as the scans could often take up a large amount of time and were not possible to do on public networks.

To manage this, placeholder data was created to allow the UI team to ensure the UI ran smoothly and worked without any major bugs. The data created mimicked the style and information conveyed by the real backend algorithms but had enough data in it to test the UI during its development. Figure 2.5 shows a sample of some of this data, which was taken from a Nmap scan and edited slightly to fit with the testing needs of the UI team.

Placeholder_Nmap_Data.csv

```
1   host,hostname,hostname_type,protocol,port,name,state,product,extrainfo,reason,ve
2   192.168.0.1,,,tcp,1,tcpmux,closed,,,reset,,3,
3   192.168.0.1,,,tcp,80,http,open,lighttpd,,syn-ack,,10,cpe:/a:lighttpd:lighttpd
4   192.168.0.1,,,tcp,443,http,open,lighttpd,,syn-ack,,10,cpe:/a:lighttpd:lighttpd
5   192.168.0.2,,,tcp,1,tcpmux,closed,,,reset,,3,
6   192.168.0.3,,,tcp,1,tcpmux,closed,,,reset,,3,
7   192.168.0.4,,,tcp,10004,tcpmux,closed,,,reset,,3,
8   192.168.0.5,,,tcp,135,msrpc,open,Microsoft Windows RPC,,syn-ack,,10,cpe:/o:micro
9   192.168.0.5,,,tcp,137,netbios-ns,filtered,,,no-response,,3,
10  192.168.0.5,,,tcp,139,netbios-ssn,open,Microsoft Windows netbios-ssn,,syn-ack,,1
11  192.168.0.5,,,tcp,445,microsoft-ds,open,,,syn-ack,,3,
12  192.168.0.6,,,tcp,135,msrpc,open,Microsoft Windows RPC,,syn-ack,,10,cpe:/o:micro
13  192.168.0.6,,,tcp,137,netbios-ns,filtered,,,no-response,,3,
```

Figure 2.5: The file containing some of the placeholder data.

When using placeholder data, the scanning functionality was turned off by commenting out the code responsible for the scans.

2.1.3.2 Using Real Data

The formal testing of the UI's capabilities was done using both physical networks and virtual ones. During the team's weekly meetings, a router was used to connect two computers and a laptop together, mimicking a real physical network on a small and manageable scale. This network can be seen in Figure 2.6.

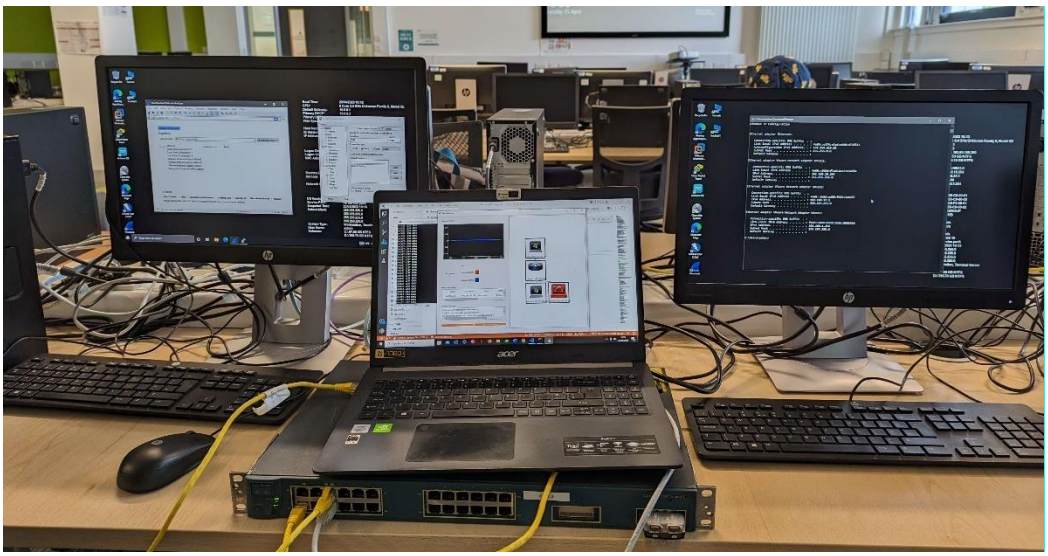


Figure 2.6: The physical network setup to test the UI and backend functionality together.

2.1.4 Presenting to the Client

The final step of each prototype cycle involved presenting the new prototype to the client. This was done during the weekly meetings, where any new features of the UI were demonstrated. Any feedback from the client was noted down and used to further improve the UI. Confirmation of any requested changes or additions was emailed to the client to allow for a record to be kept of their feedback.

2.2 BACKEND

For the backend of the project 2 teams were decided upon as it was the most efficient way to meet the deadline with 2 people working on each area with help from other team members where it was needed, clear communication between teams and especially the UI team as they would be reading the data from the backend to make the UI, was crucial to make the project meet the requirements and satisfy the client.

2.2.1 DDoS

The aim of the DDos Detection is to count how many pings a computer or device receives in a certain time frame and comparing that to the limit set by the user and then visual show if an attack is occurring in order to do that the first thing is to use Pyshark which is a python module based on wireshark which can be used to detect all traffic on a network. This was then thoroughly tested using a prototype DOS attack that sends a ping from one device to another to check that it picks up the pings and stores them correctly. This was shown to the client to check if this is what they wanted within the specification.

The DOS detection will specifically look at the pings to and from computers across a network and save the host and destination address along with a counter into a 2D array which will iterate every time that address receives a ping, this will repeat itself over a several second period before saving and being sent to the UI, it will the indefinitely repeat as needed until the DDos is turned off. This was the 2nd prototype which was then tested on a small network with 1 device being used to listen to the traffic and having 3 other devices pings each other to check that all the data is stored correctly when multiple devices are pinging each other at the same time.

If the limit is hit when checking for DDos then a change will be visually displayed making it very clear that an attack is occurring, providing the IP address of the source. This was the final prototype for DDoS before it was all put together this was done twice once setting a value in the 2D array to check that it did run as specified and another by running it on the same small network and having more pings coming in than the limit defined.

Once it was all put together it was finally run on a test network and shown to the client as the finished section allowing for any slight changes or tweaks the client may suggest. All data is stored in a csv file as this was agreed upon as the easiest way to transfer for the UI.

2.2.2 Network Scans

The network scans aim is to find the device on a network which ports of them are open, if it is open, it will then find out the version of the and name of what is on that port such as a mailserver or HTTP/S. This was first tested on a single device to make sure that using the nmap python module worked as needed to meet the requirements and work through any issues that may arise to show the client.

The network scan teams next prototype was to find the IP addresses and OS version of all devices on a network which is send to the UI for the diagram along with the open ports, this was done alongside the ports as the nmap module allows for multiple scans to be run together but for testing it was decided that doing them separately due to the nature of the module not being up to date would make it easier to check for bugs and errors. Once both were finished, they were put together and tested on a single device and across a small network thoroughly testing each time to make sure all data is in the CSV as required.

A network diagram to visually show the network to the user was a key requirement for the client, this was done by using traceroute to map out the network and show which devices are connected to one another this along with the device OS and version allowed to show if there was any routers or switches within the network and each device being clearly shown to be connected. This prototype was shown to the client in the weekly meetings to check that is met their requirements and allow for feedback for the following week.

The data from these scans needed to be set out in a way that made it easy for the UI team to understand and use when creating the frontend of the project, as they would need to read specific data from the file for each section so any changes to the format of the file would break the program, several prototypes were created

until a final format was decided upon. Mock data was used at the start to show each possible format and then it was tested to make sure that the file was formatted correctly.

3 RESULTS

The results of the Network Health Monitoring App are broken down into the following 3 sections: Design (UI), Functionality and Usability.

These 3 sections will be described below in relation to the results of the final product.

3.1 DESIGN (UI)

The overall design, layout and theme in the final product is simplistic and user-friendly. The user interface across all windows in the application uses blue as the main background colour of the app, orange for buttons within the app and a mix of white and black text which gives contrast between the background and button colours. An example of this design can be seen in the Welcome Screen of the app below in Figure 1.

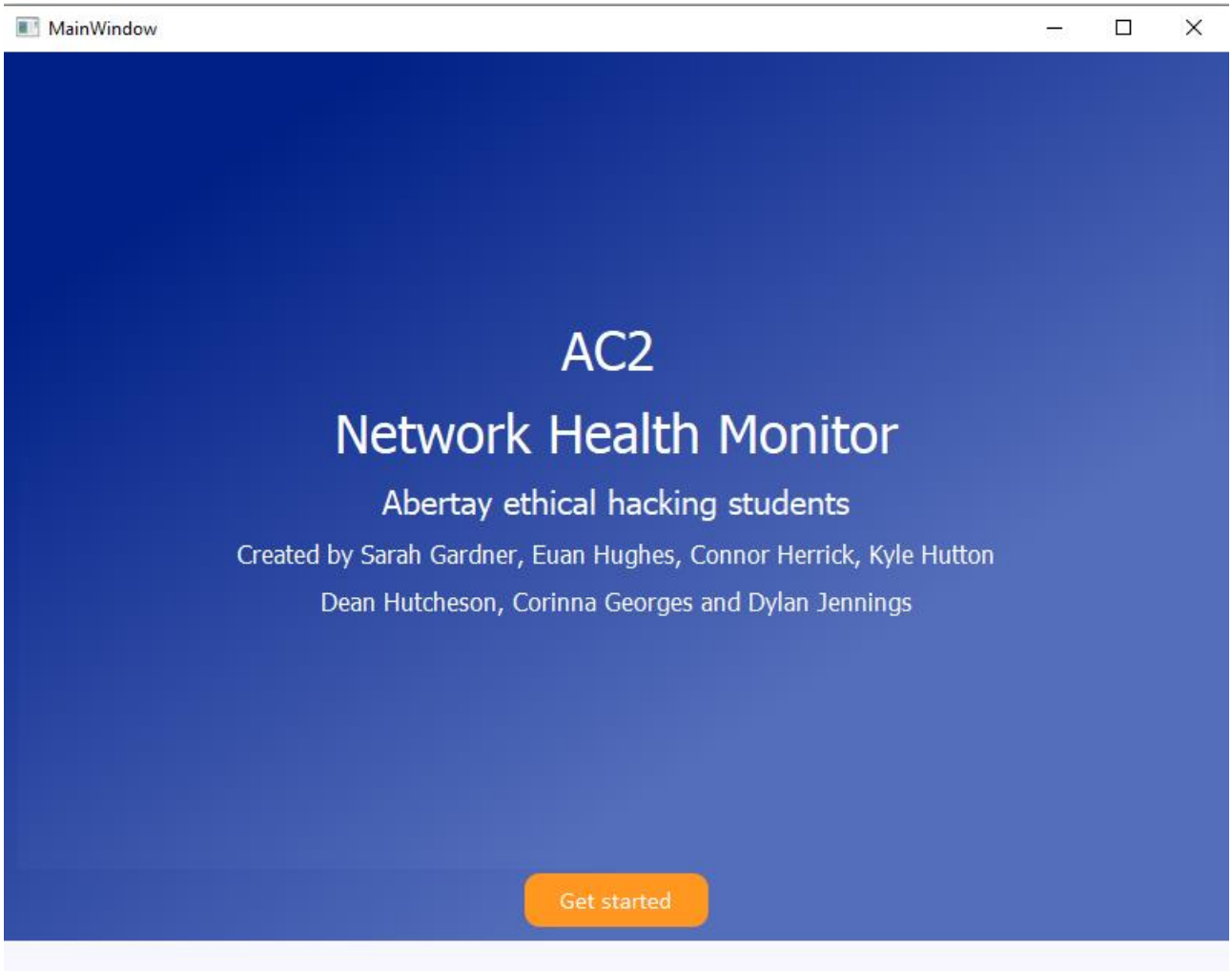


Figure 1

As seen below in Figure 2, the Input Screen follows the same simple design and offers clear and easy to understand user instructions. This window of the application has three fields that takes in user input for the target IP address which can also be range of IP addresses or an entire subnet, the interface in which the user wants to run the scan on such as a wireless or ethernet interface and the host device IP address. After the required information is input, the scan will then execute after the user clicks the “start scan” button.

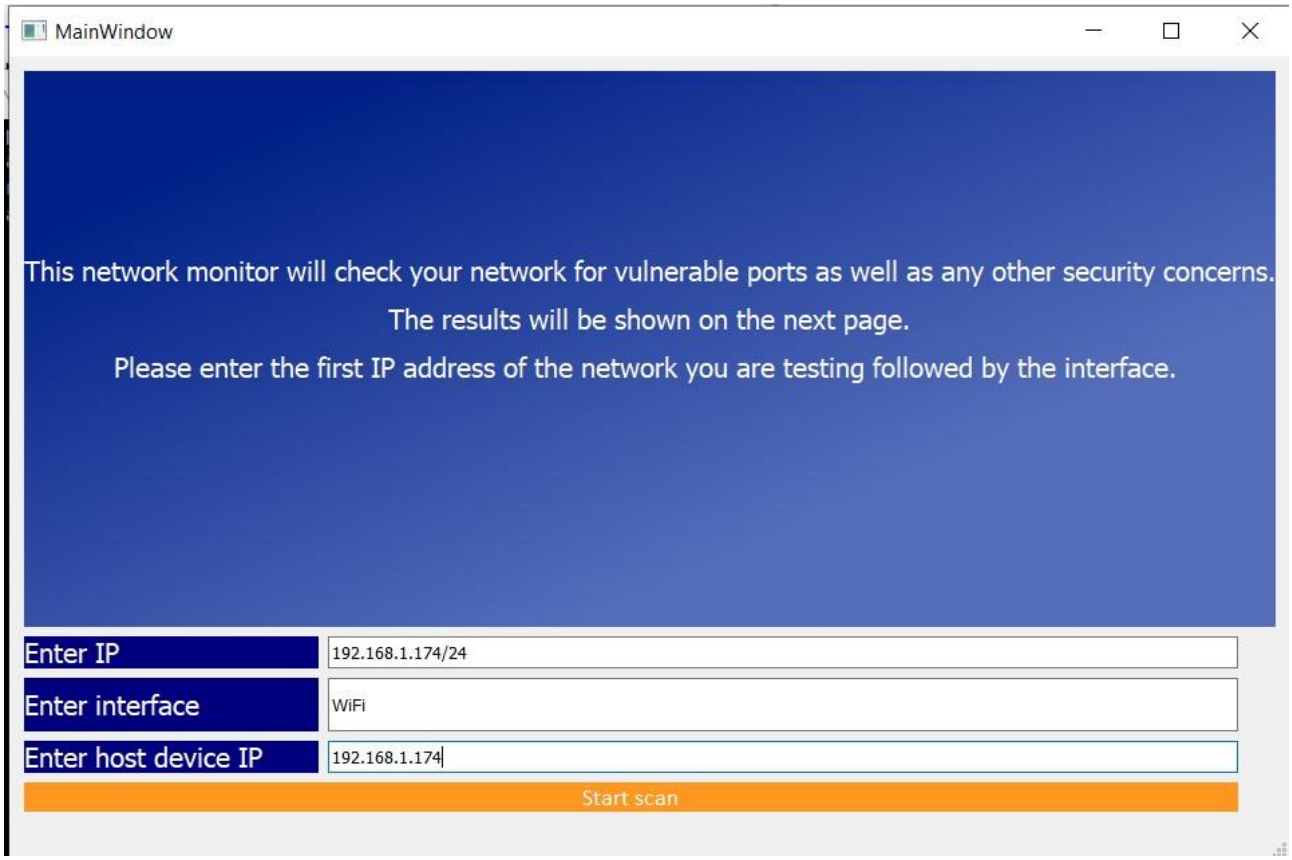


Figure 2

After the health scan is complete, the user is then directed into the Main Window of the application which contains a series of smaller windows with all the relevant information detected via the scan. The 4 sections in this window are a network traffic graph which is colour coded in red and blue for outgoing and incoming traffic, a topology of the target network with vulnerable devices displayed in the colour red, a device information field and security warnings field. This is displayed on-screen in an easy-to-understand layout with the device information separated into individual fields for hostname, ip address, service information and open ports. The security warnings field outputs how many devices are detected on the network and how many of those devices are vulnerable based off open ports that are detected through the initial scan. Further security analysis can be performed by pressing the “start DOS detection” button found at the bottom of the user interface. This will detect if any devices in the network are experiencing a DOS attack and will be displayed in the security warnings field. Figure 3 below shows the output of the complete Main Window.

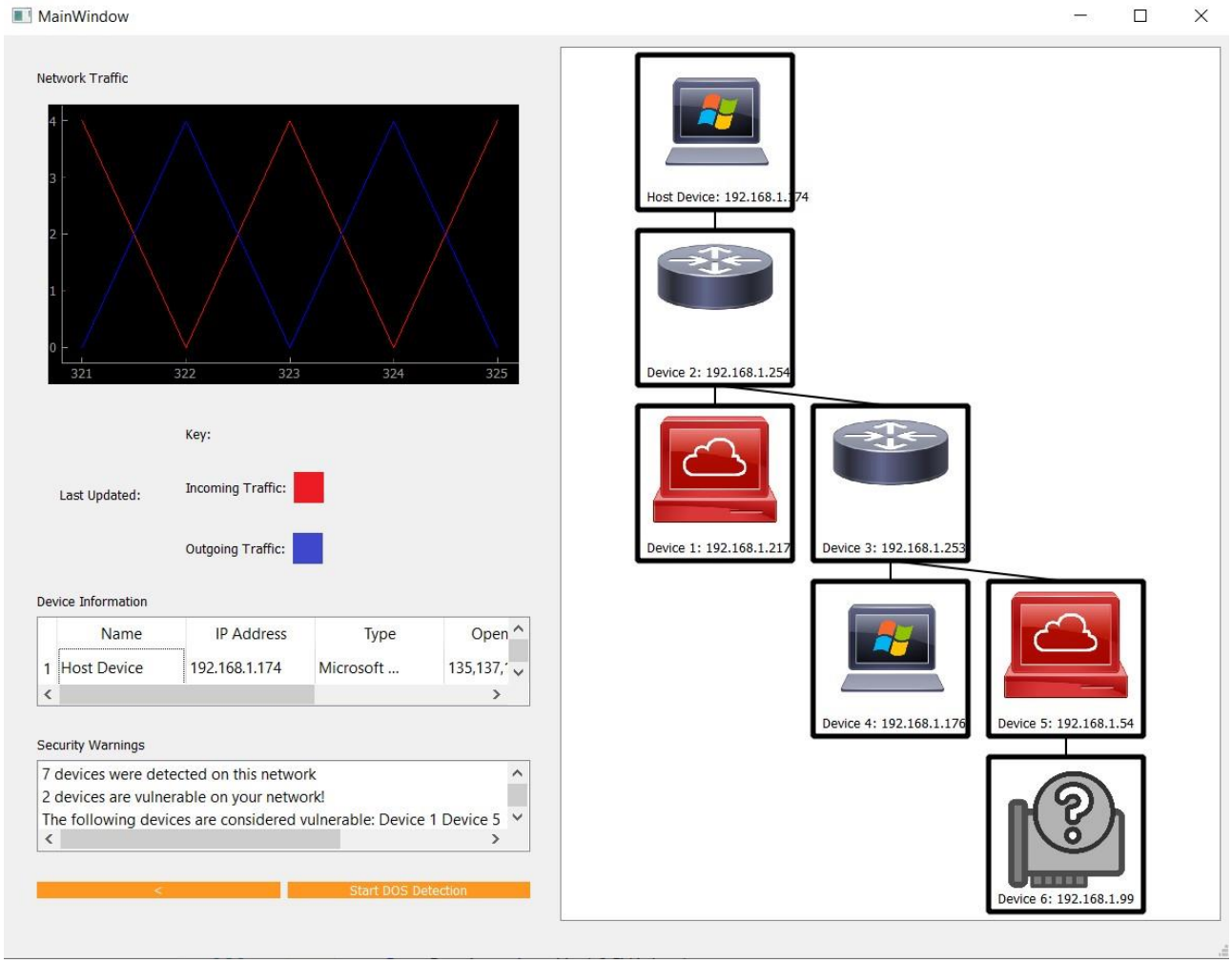


Figure 3

The last window in the application is the Device Window seen below in Figure 4. This can be accessed by clicking on a device in the device information field in the previous window and this will display a more in-depth analysis on the chosen device. There are 4 sections in this window displaying network traffic, device information, open ports, and service name and lastly an icon to provide an added visual element displaying the device type such as a Windows or Linux machine.

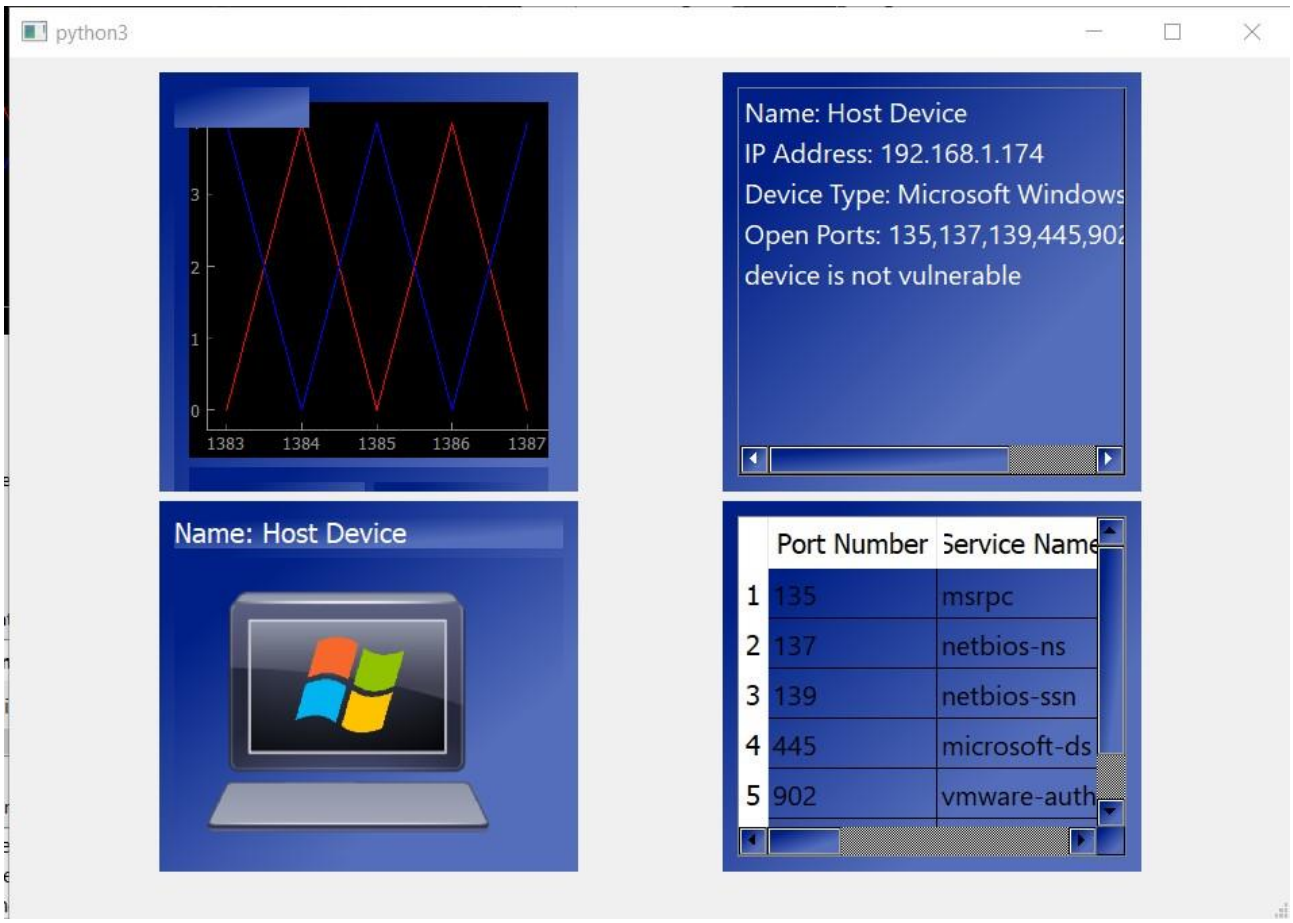

python3
3
2
1
0
1383
1384
1385
1386
1387
Name: Host Device
IP Address: 192.168.1.174
Device Type: Microsoft Windows
Open Ports: 135,137,139,445,902
device is not vulnerable
Name: Host Device
Port Number
Service Name
1 135 msrpc
2 137 netbios-ns
3 139 netbios-ssn
4 445 microsoft-ds
5 902 vmware-auth

Figure 4

3.1 FUNCTIONALITY

Due to the simplistic design of the application and clear and concise instructions on how to use it, this makes the tool very functional and fit for purpose. This was proven during the testing phases of the project that demonstrated that all functions within the application performed and output results as expected. All backend code runs correctly and efficiently, and error-handling has been correctly utilized in case the user enters invalid arguments/information into the input fields. In an instance where a user enters in valid information, but the target IP is unreachable from the host device, the network map will appear blank and the functions within the app will not work.

Functionality is lacking in the case if a target device/network is sitting behind a firewall. Due to the firewall blocking packets, it prevents the application from reaching anything beyond that point. This will be explained in more depth in the future work section of the report.

3.2 USABILITY

The usability of the application is satisfactory. This can be seen in the overall design of the user interface which is clean and simple in its layout and theme, free of any spelling errors throughout all windows of the application and clear in its instructions on how to use the tool and what the purpose of the application is. This makes the application more universal as it is user-friendly enough to accommodate users of all different technical backgrounds. The prerequisite is that the user must know the target ip, interface and host ip address. Apart from those three input requirements, the rest of the application only requires the user to interact with buttons which will automate all scans and output. This makes the overall usability simple. All testers on the project are happy with the overall functionality and usability of the final product and although there is room for improvements, the targets have been met and an adequate network health monitoring tool has been produced for the client.

4 DISCUSSION

4.1 GENERAL DISCUSSION

Created a GUI-based application using PyQt5 library that allows users to monitor network traffic and identify suspicious activity. The use of a graphical user interface (GUI) can improve the usability of the system, making it more accessible to a wider range of users. This can help to increase the adoption and usage of the system, leading to more accurate and comprehensive network health monitoring.

The project involves several different Python files that work together to provide features and functionality. The different files work with each other starting the application and launching the different components to display the results. Having the application in a modular way can provide several benefits for network health monitoring, including Scalability, Code Reusability, Maintainability and Collaboration. Modular scripts can greatly improve the effectiveness and efficiency of network health monitoring systems, as well as make it easier to develop, maintain, and collaborate on the codebase.

The findings of the project are relevant to other work in this area, as network monitoring is an important aspect of cybersecurity. By identifying suspicious activity and potential threats, network monitoring tools can help organizations improve their security posture and protect against cyber-attacks. The project may contribute to the development of better network monitoring tools and techniques and can help raise awareness about the importance of cybersecurity.

Overall, the benefits of this network health monitoring system can include improved network performance, reduced downtime, and increased security, which can lead to a positive return on investment for the client. By addressing the client's main points of inefficient network monitoring and potential security threats, this system can provide a valuable solution to meet their needs. There is room for improvements, however these will be discussed in the future work section.

By highlighting abnormal network activity, displaying vulnerable open ports, classifying devices on the network, and displaying all discovered devices, the project has achieved its goals. By following the quality baseline, the team was able to successfully achieve each marker, which are:

1. All scripts and coding must run, be combined, and show results, Coding errors must remain below 5% in total.
2. Must run, simplistic layout and simple results. Code/spelling errors to be less than 2%.
3. Identify 90% of clients' IPs in their network.
4. Displays all open ports.
5. Display device found but lack any detail and how many devices.
6. Higher than usual traffic is displayed on screen, very basic, lacking in any detail.
7. No errors with running software, or the projects end product on client system.
8. Spelling errors to be less than 10% in total for each document, email to client less than 2%.

4.2 CONCLUSIONS

Identified several benefits the application solution can provide for network health monitoring needs:

1. Scalability: The modular script is designed to be scalable, allowing to monitor large networks without compromising performance.
2. Automation: The script automates the monitoring process, reducing the time and effort required to manually monitor network health.
3. Customization: The solution can be customized to meet specific monitoring requirements and can be easily integrated with existing systems.
4. Real-time Alerts: The script can send real-time alerts to notify of any potential network health issues, enabling them to quickly address any problems.
5. Cost-effective: The solution is cost-effective compared to traditional network monitoring tools, without compromising functionality, easy integration with existing monitoring tools and importantly reduced downtime and associated costs.

It is crucial for the client to use this solution to ensure the health and reliability of their network. If they do not use it, they risk facing downtime, loss of productivity, damage to reputation, and costly repairs.

In summary, the modular scripts developed for network health monitoring are a valuable solution for the client. The benefits include improved network performance, reduced downtime, quick issue identification and resolution, customizable alerting and reporting, and easy integration with existing monitoring tools. It is crucial for the client to implement this solution to avoid network issues and ensure business continuity.

4.3 FUTURE WORK OR COUNTERMEASURES

If given more time and resources, there are several areas that could be further developed and improved upon. These potential next steps could include:

1. Incorporating machine learning algorithms for more advanced anomaly detection and predictive maintenance.
2. Implementing real-time visualization of network performance metrics for better monitoring and analysis.
3. Integrating the system with other network management tools for a more comprehensive solution.
4. Conducting further testing and validation to ensure the system is robust and reliable in various network environments.
5. Improved network investigation probing for network gathering behind firewalls.
6. Unknown and unreachable devices are placed in a separate table for the user to view.
7. Detection of other attacks, Denial of Service, other methods than ping packets being used.
8. If the device is vulnerable, give the user an explanation as to why.

It is important to prioritize ongoing maintenance and updates to ensure the system remains secure and up to date with the latest threats and vulnerabilities. Regular security audits and penetration testing can also help identify any weaknesses or areas for improvement. Adding; Intrusion detection and prevention, segmenting a network can help limit the spread of malware or other malicious activity, using encryption, strong authentication, access control, and additionally, user education and awareness training can help mitigate risks associated with human error and social engineering attacks.

4.4 CALL TO ACTION

The application offers the following benefits to the organization:

- Real-time monitoring of network health and performance.
- Early detection of potential issues and proactive measures to prevent downtime.
- Customizable alerts and notifications to keep IT team informed.
- Modular and scalable design to fit organization's specific needs.
- User-friendly interface for easy navigation and understanding of network data.

To ensure the organization can make an informed decision about the network health monitoring application, a free 14-day trial version of the product is offered to test its capabilities and see how it can benefit the organization, which includes all features of the product. During the trial, the organization's IT team will have access to the product support team to answer any questions and guide them through the product.

Also provided is comprehensive training to ensure that the IT team can maximize the benefits of the application. The training includes in-person or online sessions, instructional materials, and best practices to ensure that the team can manage the application effectively.

For more information, to schedule a demo or request a free trial, please contact 2201262@abertay.ac.uk or visit www.AC2.uk.
Looking forward to working to enhance organizations network's health and performance.

5 User Manual

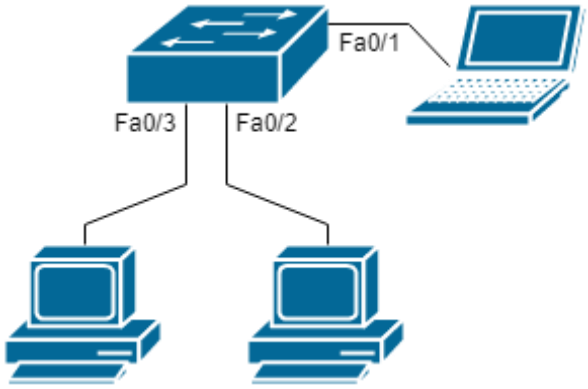
5.1 Audience and Purpose

A user of the network health monitoring application would be expected to have fair technical experience in the field of cyber security, as per the brief given to the Project Team. However, the overall application is intended to be easy to use and the graphical interfaces have been designed to be as user friendly as possible. Regardless, this manual aims to give an in-depth guide to how the different parts of the application can be used to assess a network's health.

5.2 Requirements

The health monitoring application was written for and tested with Windows 10 computers. It may work on different Windows versions but will definitely not work on any other operating systems.

An initial network setup must be done before anything else. To function effectively the application requires copies of every packet being sent across the network being assessed. A network switch with SPAN (Switch Port Analyzer) functionality is recommended, alternatives to this are possible, but this method has been tested and confirmed to work.



For the network shown above, an additional device must be plugged into the switch's COM port. A tool such as PuTTY can then be used to make a serial connection to the switch. Once logged into a Cisco terminal, the setup commands for a Cisco Catalyst 2900XL/3500XL switch (the model used during testing) are as follows:

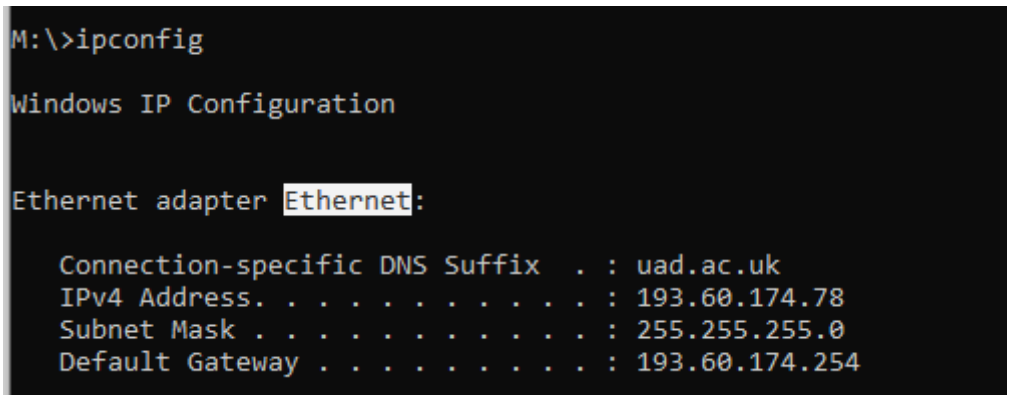
```
configure terminal
interface Fa0/1
port monitor Fa0/2
port monitor Fa0/3
```

Additional information can be found on Cisco's website.

https://www.cisco.com/c/en/us/support/docs/switches/catalyst-6500-series-switches/10570-41.html#anc13

These commands will cause the switch to start making a copy of packets going in or out of the Fa0/2 and Fa0/5 ports and begin sending them to Fa0/1. Once this is the case the denial of service detection will be fully functional.

After connecting a network interface of the machine hosting the network health monitor to the switch's SPAN port, the software can be set up. Make note of the interface's name, in this case it was "Ethernet". Interface names on Windows can be shown by running ipconfig.



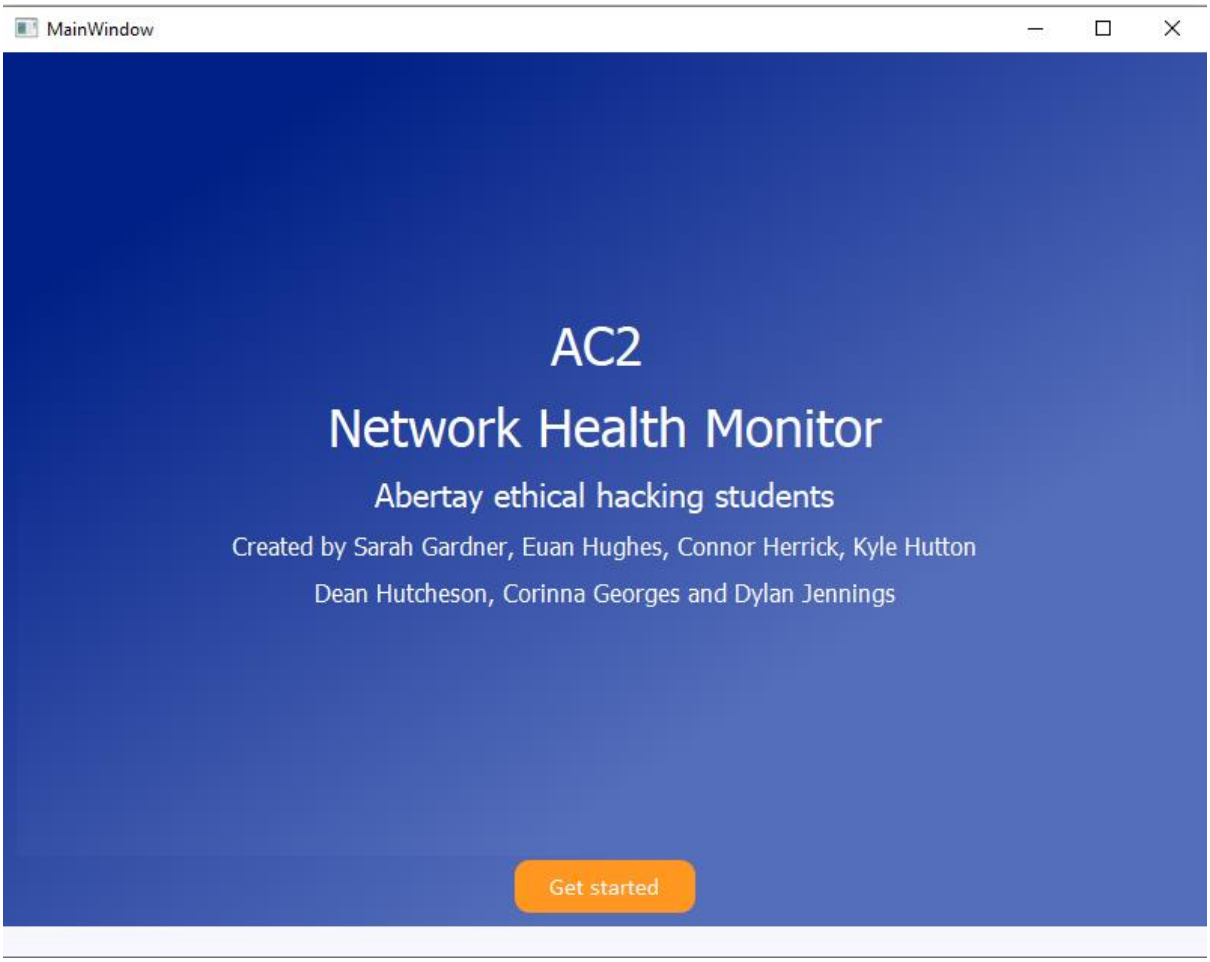
The application is written in Python 3 and will require Python version 3.11.3 to be installed. And before running, there are prerequisite Python modules that will need to be installed.

Name	Version	Link
PyQt5	5.15.9	https://www.riverbankcomputing.com/software/pyqt/
python-nmap	0.7.1	https://xael.org/pages/python-nmap-en.html
pyshark	0.6	https://github.com/KimiNewt/pyshark/
PyQtGraph	0.13.3	https://www.pyqtgraph.org/

Once the application has started, knowledge of the host machine's IP address and the network's subnets is required to create an accurate network map and fully detect every device connected to the switch.

5.3 INSTRUCTIONS

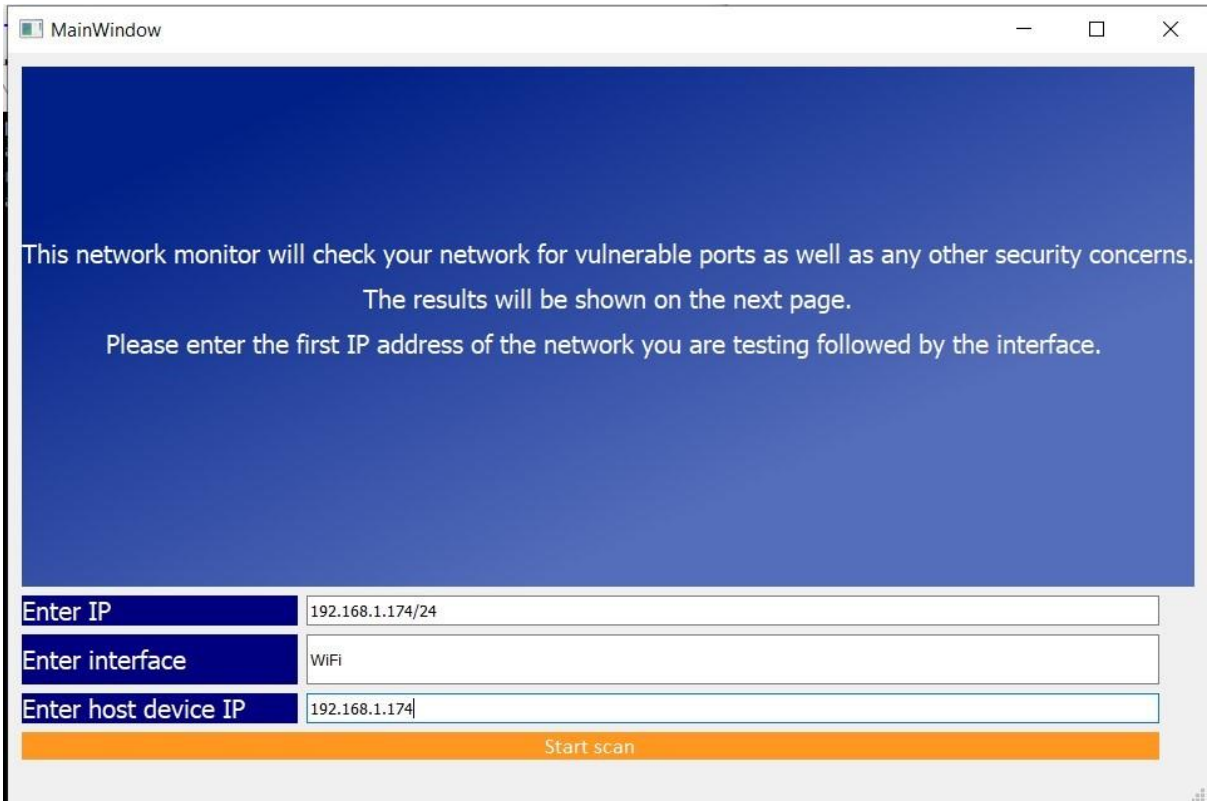
After the application successfully launches it will display the introduction window as shown below.



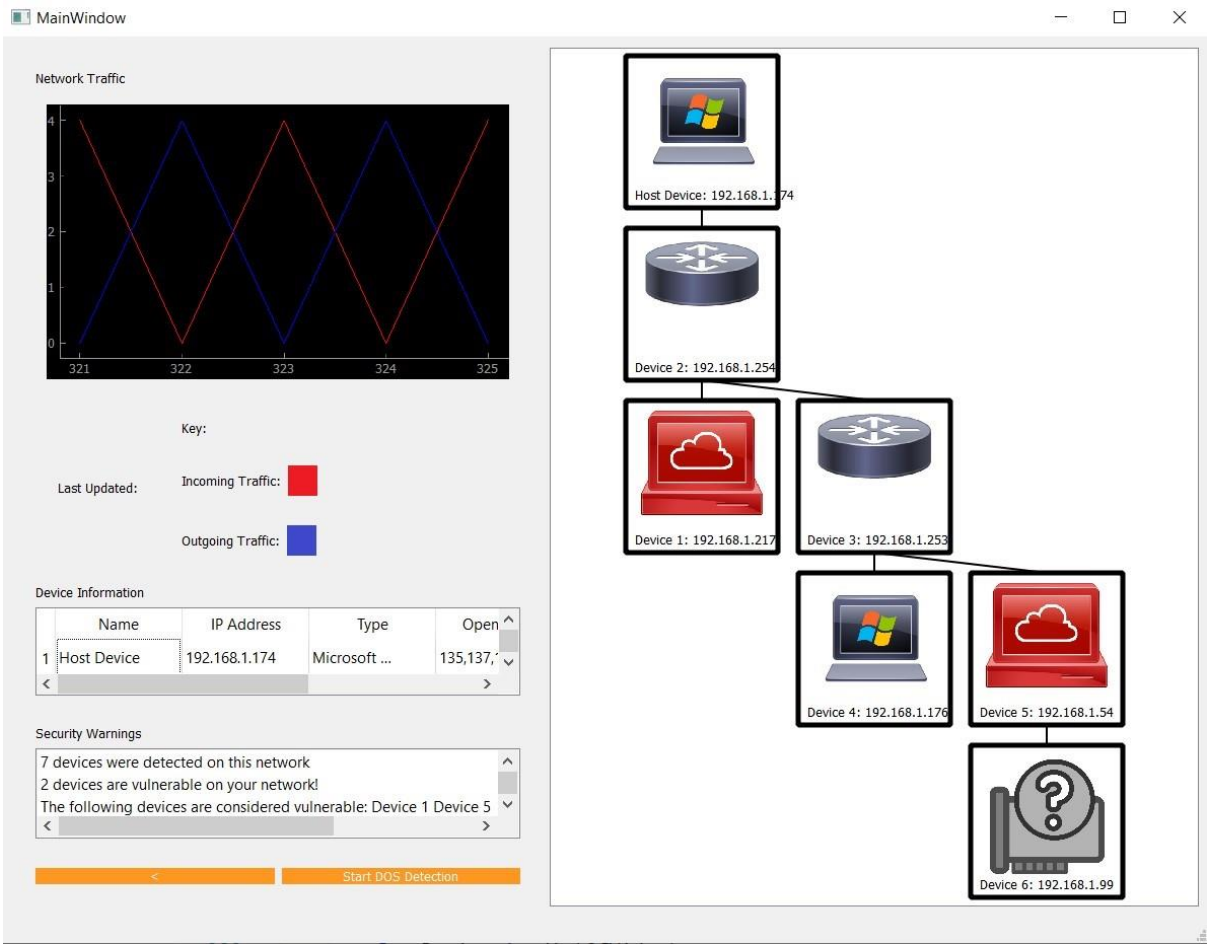
After clicking “Get started” a user will be brought to the network scanning screen and must enter an IP address or address range. This string will be put straight into an Nmap command so can be in multiple formats.

192.168.0.10	A single device
10.0.0.46 10.0.0.47	Two specific devices
169.254.10.4-32	From 169.254.10.4 to 169.254.10.32
172.10.54.0/30	Devices in this subnet

Additionally, a network interface must be specified. This would be the one used by host machine to connect to the switch.



To begin mapping the devices the user can click “Start scan”.

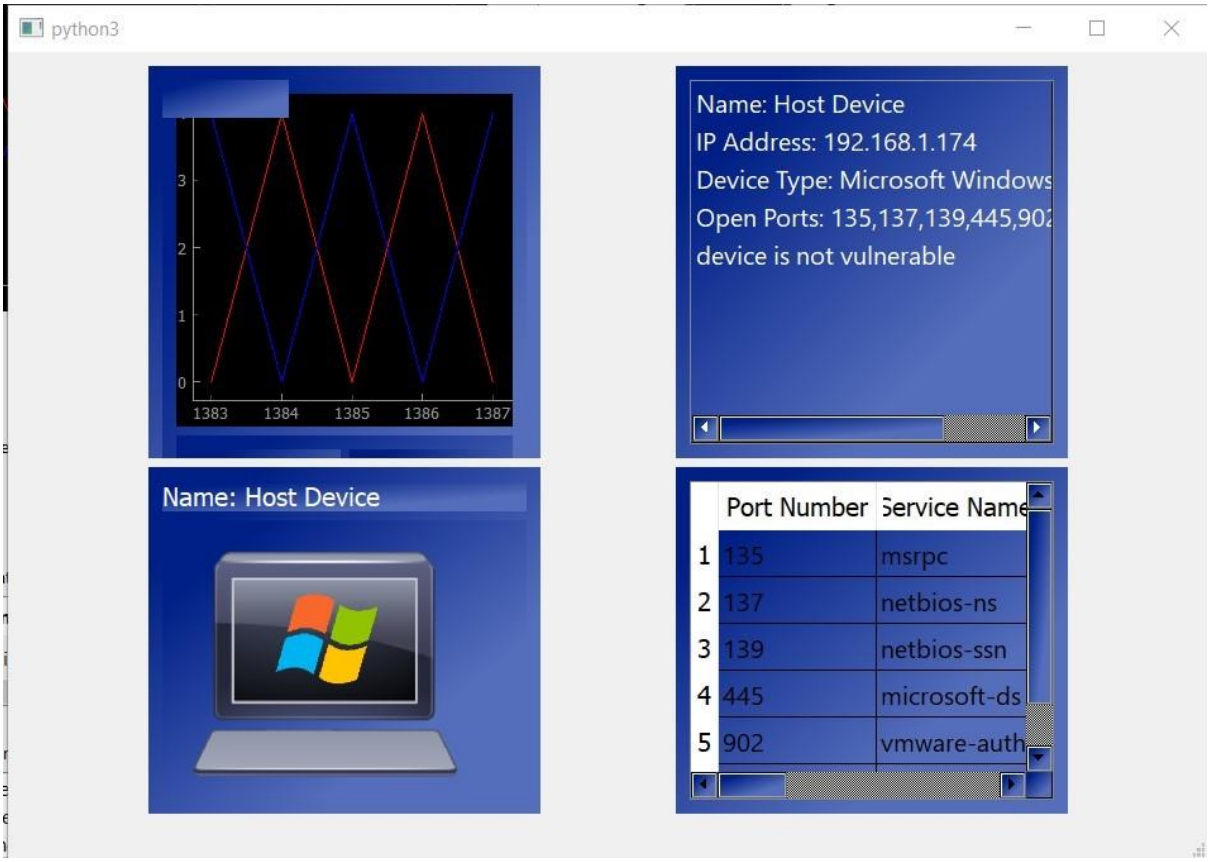


After the scan completes the user should be shown a visual network map with nodes representing the different devices alongside some accompanying information. At this point the application may show any detected issues regarding the network’s health. In the case above, two devices have been identified as

vulnerable and thus are coloured red. Furthermore, this can be seen in the “Security Warnings” section in the bottom left.

To begin monitoring for denial of service attacks the “Start DOS Detection” button may be clicked. Once clicked, this will start showing data in the network traffic graph which may give a visual hint that a denial of service attack is underway. However, if the application detects an influx of packets an alert will appear in the “Security Warnings” section.

Further detail about a device can be seen by clicking on said device’s icon which will cause a new window to pop up.



5.4 Frequently Asked Questions

Q: How long does the scanning stage take?

A: For a small network of ten devices this would take a couple of minutes. But it should be considered that some complex devices will take significantly longer to scan.

Q: Will the network map be updated periodically?

A: No, the only way to get an updated view of the network is to press the "<" button which will go back to the network scanning screen so you can rescan the network manually.

Q: Why is the application crashing during the scanning phase?

A: This issue may be related to how the devices are being mapped. Specifically in the case that a device is discovered with Nmap but is unreachable using standard pings. If this happens the application will have a device without a network path leading to it and will be unable to place it onto the map. The only way around this issue is to figure out which devices are blocking standard ICMP ping requests and unblock them.

Q: Why isn't the DOS detection showing any network traffic?

A: TShark has a known bug causing the program not to capture packets if its graphical counterpart Wireshark hasn't been opened. Therefore, until the developers have patched this bug, Wireshark must also be installed and run separately. Only then will the application's denial of service functionality work.

5.5 Contact Information

Any questions regarding the use of this software can be directed to Sarah

6 REFERENCES

GOV.UK. (2022). Cyber Security Breaches Survey 2022. [online] Available at: https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2022/cyber-security-breaches-survey-2022 [Accessed 28 Apr. 2023].

Froehlich, A. (2021). network analyzer (protocol analyzer or packet analyzer). [online] Networking. Available at: https://www.techtarget.com/searchnetworking/definition/network-analyzer [Accessed 30 Apr. 2023].

Jonathan Arnowitz, M. A. (2010). *Effective Prototyping for Software Makers* (1st ed.). San Francisco: Morgan Kaufmann Publishers.

Phillips, D., Giridhar, C., & Kasampalis, S. (2016). *Python: Master the Art of Design Patterns* (1st ed.). Birmingham: Packt Publishing Ltd.

APPENDICES

APPENDIX A – CLIENT BRIEF

"The developed application must visualise the scanned network and should attempt to classify between clients, servers and network devices from the traffic analysed. If clients connected on the network are sending or requesting abnormal data from other similar clients, they should be highlighted on the network map as being potentially compromised systems. Similarly, if clients have vulnerable application ports open, they should be highlighted as being potentially vulnerable systems."

APPENDIX D - DELIVERABLES & REQUIREMENTS (REQUIRED)

Agreement Form: Project Deliverables

Group Name, Names of Team Members, and Programme	Team Name: AC2 Team Number: EH9 Sarah Gardiner, [Redacted], [Redacted], [Redacted], [Redacted], [Redacted], and [Redacted]
Subject specialist's Name (Client)	Mr. Ross Heenan
The deliverables listed below will be submitted by the team by the due date.	
Part A deliverables	**To be agreed by program specialist and team, for example:** • Working executable code. • User manual. • A network map presented on the GUI consisting of the devices, their types (such as PC, router, server), and containing information about them (such as open ports, and operating system type) • Compiled package including technical GUI. • Results easy to view and understand. • Flexibility to work across multiple types of networks. • Performance benchmark data • Installer downloading prerequisite packages, including Nmap and libraries used within the program.
Subject specialist's (Client) signature	Mr.Ross Heenan
Team members' signatures	Sarah Gardiner, [Redacted], [Redacted], [Redacted], [Redacted], [Redacted], and [Redacted]

Agreement Form: Requirements

Group Name: AC2

Team members (print): Sarah Gardiner, [Redacted], [Redacted], [Redacted], [Redacted], [Redacted] and [Redacted]

Project Title: Network Health Monitoring Application.

Please refer to the attached documentation for full details on the project. The requirements are listed in Table 1. The signatures below indicate that the requirements for this project have been agreed by the project stakeholders.

Any changes to the project documentation should be made using the correct change authorisation procedure agreed with the programme specialist.

Table 1

ID	List of Agreed Requirements (fill in)
	• IP identification • Identify vulnerable open ports. • Identifying devices • Flag any DDOs traffic. • Working code • Program usability • Program compatibility • Program manual

Stakeholders	Signatures	Date
Team members	[Redacted]	16/02/2023
	[Redacted]	16/02/2023
	[Redacted]	16/02/2023
	Sarah Gardiner	16/02/2023
	[Redacted]	16/02/2023
	[Redacted]	16/02/2023
	[Redacted]	16/02/2023

Subject Specialist	Ross Heenan	02/05/23
Client (if applicable)	Ross Heenan	02/05/23