Network Guardian

TEAM *PENTAGON*

Alexandra Cherry,

Declan Woodham,

Owen Nelson,

Stewart Anderson,

Velislav Velchev

CMP311: Professional Project Development & Delivery

Version Date: 21/04/2020

VERSION HISTORY

Version Number	Implemented By	Revision Date	Approved By	Approval Date	Description of Change
1.0	Stewart Anderson	25/02/2020	Declan Woodham	25/02/2020	Created the Structure of White Paper
1.1	Stewart Anderson	03/03/2020	Declan Woodham	03/03/2020	Started on Creating Graphic User Interface section
1.2	Stewart Anderson	10/03/2020	Declan Woodham	03/03/2020	Completed the aim
1.3	Stewart Anderson	14/04/2020	Declan Woodham	15/04/2020	Added small descriptions for flask, jinja and PyWebView
1.4	Stewart Anderson	15/04/2020	Declan Woodham	15/04/2020	Added stuff to execution of the plan, Future work and issues section
1.5	Declan Woodham	15/04/2020	Stewart Anderson	15/04/2020	Modified Future Work, Testing section
1.6	Declan Woodham	16/04/2020	Stewart Anderson	16/04/2020	Finished Plugin System section
1.7	Owen Nelson	17/04/2020	Declan Woodham	18/04/2020	NMAP
1.8	Declan Woodham	17/04/2020	Stewart Anderson	18/04/2020	Added Legal Guideline Section
1.9	Stewart Anderson	17/04/2020	Declan Woodham	18/04/2020	Execution of the Plan and more to added to future work (Software, Team management andDevelopment Process)
2.0	Declan Woodham	18/04/2020	Stewart Anderson	19/04/2020	Added Application Packing and Cross Compatibility Section
2.1	Alexandra Cherry	18/04/2020	Declan Woodham	19/04/2020	Testing
2.2	Declan Woodham	19/04/2020	Stewart Anderson	20/04/2020	Finished Report Generation Section
2.3	Declan Woodham	19/04/2020	Stewart Anderson	20/04/2020	Started Evaluation of Enumeration Techniques Section
2.4	VelislavVelchev	20/04/2020	Declan Woodham	20/04/2020	Completed Background Section
2.5	Declan Woodham	20/04/2020	Stewart Anderson	21/04/2020	Documented Security Vulnerability and improvements
2.6	Stewart Anderson	20/04/2020	Declan Woodham	21/04/2020	Abstract and completed Evaluation section
2.7	Declan Woodham	20/04/2020	Stewart Anderson	21/04/2020	Added and Finished Ensuring Quality Sections
2.8	Declan Woodham	21/04/2020	Stewart Anderson	21/04/2020	Added to Abstract Section

2.9	Declan Woodham	21/04/2020	Stewart Anderson	21/04/2020	Formatted Tables, and Text. Updated fields.
3.0	Team Pentagon	21/04/2020	Team Pentagon	21/04/2020	Completed Document

TABLE OF CONTENTS

1 Abstract

Network Administrators, and Pen-testers are when developing network assessments often face the same routine, following the same repetitive steps to achieve their goal. Network Assessments are a vital step in protecting network security, in aim to reduce risk, save potential money lost through theft or damages, and to maintain reputation.

Although the goal of a Network Administrator and a Pen tester may differ, the procedure and the process of developing the Network Assessment will always share the same key step. Enumeration is often the first, vital step taken and is the process of gathering information from various sources to allow the network assessment to progress. Information such as the network configuration, available host machines, and information about the device's hardware and software.

As mentioned, the process although being vital, is often repetitive, and time consuming, opening the risk for human error, with key steps forgotten, or the risk of a vulnerability lying dormant on the network for longer.

Dr Ethan Bayne of the University of Abertay Dundee requested the creation of a cross platform software tool which would in turn automate the enumeration stage, and further provide an added educational value, teaching students at the University the logic behind the creation of network assessment enumeration techniques.

The purpose of this paper is to document the processes which the members of Team Pentagon took to produce the tool, including the evaluation of the need for the tool, the design choices of the software, and the methods the team took to test the software and maintain a high standard of quality.

Network Guardian can help carry out the enumeration stage of a network assessment, gathering and storing information about a network in a structured report. The information can then be easily read and analysed by network administrators or penetration testers. The software reduces the risk of human error and the time taken to carry out the enumeration stage. The reports created can be exported into HTML files to easily distribute the information, without requiring the use of Network Guardian to read them. The software features user programmable plugins allowing end users the ability to fine tune Network Guardian so that reports can be tailored to suit a specific network assessment or requirements, which is unique in the current market. A plugin that Network Guardian uniquely features, is a Network Visualisation plugin which displays a map of the network, allowing the end user to easily picture the layout of a network.

The team followed a SCRUM methodology, this allowed the team to split the project into smaller segments and distribute them to different team members, so that tasks could be carried out at the same time, improving the overall productivity. This methodology involved the client throughout the development of the product, which meant that changes could be made quickly and easily (SECTION 6.3.2 Changes to the Requirements Specification). This methodology entailed regular meetings with the team, ensuring everyone was productive and understood the task at hand. A Gantt chart was also used to keep track of the different tasks being completed and needing done, this was useful as it's easy to visualise and display tasks that can be done at the same time, or tasks that require something to be completed first.

Through reflection, testing and careful evaluation of the final product Network Guardian, the team believes that the tool is provides an asset to the both Network Administrators and Pen testers. Improving their workflow speed, while also reducing risks.

Key Words: Network Administrator, Pen-testing, Enumeration, Educational Value, ISO 25010, SCRUM

2 Introduction

2.1 Background

The University of Abertay Dundee (UAD) under the request of Dr Ethan Bayne, assigned the brief for the creation of a network assessment tool which would provide various important information about any given network. The end users of the software are aimed towards network administrators, security penetration testers and students of UAD.

A Network Assessment is a detailed report and analysis of any network about its infrastructure, security, the performance, potential improvements and the current state of the network.

Network administrators and others who work in digital security may face issues when carrying out their roles regarding the security of the network as the pre-existing, off the shelf products can require a high start-up cost (both monetarily and timewise), are not as user friendly or flexible as they require, nor do they provide the educational value that our client is requesting.

This could create a major issue for security professionals as they are faced with either paying for an expensive product or taking the time to create scripts in order to do their job effectively which may result in turning important tasks (e.g. initial assessment and enumeration) into something tedious.

Enumeration is a vital first step in assessing networks as it is the process of extracting information about the network (e.g. hosts, connected devices and user information) from multiple sources which provides a more detailed view of the network.

Unlike these tools, Network Guardian is free, open source, and provides better functionality at the same time. And because of that, they are able to properly assess the specific network in a quick manner and therefore are able to improve the overall security of it if needed in a quicker and more efficient way than doing it manually or having to buy an expensive tool to do the job.

By using Network Guardian, network administrators and security professionals can create their own plugins or use the standard ones in Network Guardian. This makes it useful, as it provides the ability to fine tune the tool towards the specific network requirements while keeping the costs low.

When compared with the competition, Network Guardian goes above and beyond in terms of both functionality and user experience which makes it stand out in the current market with its features.

2.2 Aim

The aim of the project is to produce a piece of software that can carry out, unattended, the enumeration stage of a typical white box penetration test. The software will be a beneficial tool to Network Administrators and Penetration Testers, as well as provide an educational value in the form of user programmable plugins which students can develop.

The aim is to complete the project following a plan over the course of 10 weeks, with a 3-week buffer to ensure that the project is completed on time. This buffer allows for any problems or issues that may arise to be addressed and sorted out.

2.3 General Information

Software Name	Network Guardian
Developed by	**Team Pentagon** *Alexandra Cherry* *Declan Woodham* *Owen Nelson* *Stewart Anderson* *Velislav Velchev*
Subject Specialist	Dr Ethan Bayne
Supported Platforms	Windows, Linux, Mac OS X
Final Release Date	04/21/2020

3 Procedure

3.1 Creation of Network Guardian Software

3.1.1 Plugin System

One of the early decisions when developing the plan for the software, was to include the user programmable plugin system to allow end users of the software to integrate their own network assessment practices, procedures and tools into the software. This was to provide an additional educational value, teaching end users and students at the University of Abertay how to develop their own scripted network assessment tools, and to further provide a far greater ability to meet the requirements of the end users' networks.

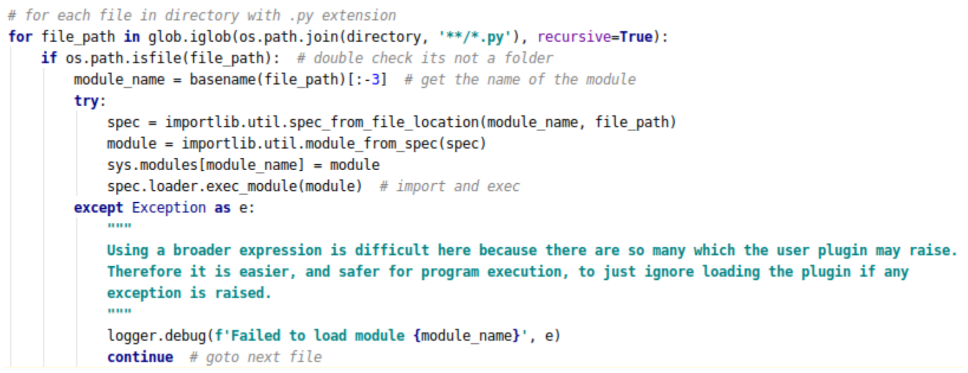
The plugin system itself during development endured numerous iterations during the development of the plugins to improve the way it worked, and to make it more flexible. Due to the agile iterative nature of the development, this was easy and during our meetings, updates to the system were relayed from the framework developer, to the plugin developers.

3.1.1.1 Loading Plugins from a File

Plugin loading was one of the first issues during the development, Python has several methods of utilizing python code from files as it is what it uses itself to load code from its various standard libraries. The difficulty with using these methods however is that it was difficult to load the files from a directory outside of the working environment.

An early version of the system used methods found online but was found to be unstable across different operating systems. Alterations could have been made to use different functions based on the running operating system, but instead different measures were investigated.

With more research into the Python architecture, and with some reverse engineering of the package installer included with Python. A library of functions included was found which satisfied the needs, allowing Network Guardian to load files from any directory path with recursion.



```python
# for each file in directory with .py extension
for file_path in glob.iglob(os.path.join(directory, '**/*.py'), recursive=True):
    if os.path.isfile(file_path):  # double check its not a folder
        module_name = basename(file_path)[:-3]  # get the name of the module
        try:
            spec = importlib.util.spec_from_file_location(module_name, file_path)
            module = importlib.util.module_from_spec(spec)
            sys.modules[module_name] = module
            spec.loader.exec_module(module)  # import and exec
        except Exception as e:
            """
            Using a broader expression is difficult here because there are so many which the user plugin may raise.
            Therefore it is easier, and safer for program execution, to just ignore loading the plugin if any
            exception is raised.
            """
            logger.debug(f'Failed to load module {module_name}', e)
            continue  # goto next file
```

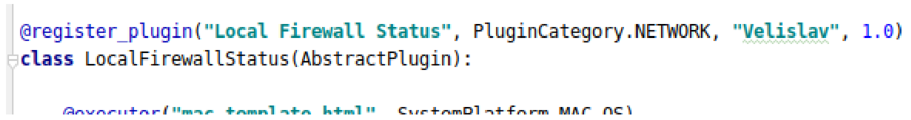
Figure 1: Python code used to load plugins from a directory recursively

3.1.1.2 Registering Plugins

The next conscious design decision was how the software was going to identify and create an instance of the loaded plugins. At first, a decision was made to include a file where the plugins would have to be registered before, they could be activated. This however would be a manual process and following guidance from the Usability Factor in ISO 25010, this would not satisfy the quality indicator requirements the team wanted to meet.

The final method which was used to load the plugins comes in the form of an annotation. Annotations in python provide a way to initiate or call classes and functions at the initiation of the software, rather than when they are called. This was then implemented to register the loaded plugin files into memory by creating an instance with the required parameters, and then adding the instance into an array where the software can call them to generate report information.

An example of the register plugin annotation.



```
@register_plugin("Local Firewall Status", PluginCategory.NETWORK, "Velislav", 1.0)
class LocalFirewallStatus(AbstractPlugin):
```

Figure 2: Register Plugin Annotation in use

Developers of plugins, in and outside the Network Guardian team simply import and call the “register_plugin” annotation at the top of their plugin class and place the python file in the “Plugins” directory for it to work.

3.1.1.3 Plugin Architecture

To ensure that plugins would work and be accepted into the software, an abstract base class called “AbstractPlugin” was created. An abstract class is a class which cannot be instantiated itself, its aim for our purposes is the provide a set of abstract functions which can be implemented by the plugins which derive from it. This type of architecture is a form of object orientated programming and allows Network Guardian to validate plugins when loading and executing.

The abstract plugin is also responsible for storing the plugin metadata, such as name, version, description, and template files in memory.

3.1.1.4 Plugin Executors

During the early iterations of the Network Guardian software, the abstract plugin class provided a single function called “process” which the plugins developers would override and implement their methods of enumeration. This served its purpose, but issues arose when plugins provided different (or no) support for certain operating systems.

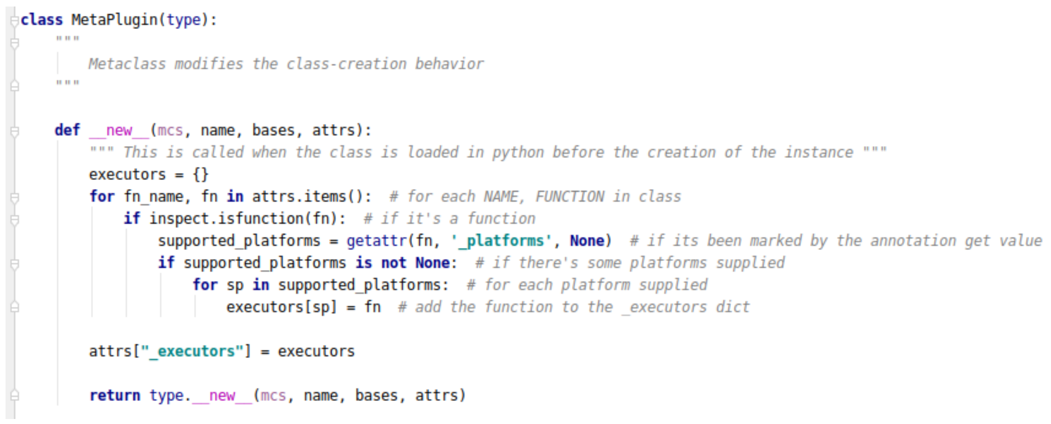
Developers found that they were often repeating boilerplate code used to determine different operating systems, and overall, this design concept was not very pythonic or nice to read.

An alternative method of providing an executor was needed, so the framework developer tested and trialled different methods until the team was satisfied with the proper result. The iterations took place at different meetings, due to the version control system used by the team, it was easy to merge and switch between different systems.

The first method attempted was to simply create three different “process” functions, one for each supported platform. This worked but also added boilerplate code and functions calling functions, as it was found that some plugins would support Linux and Mac OS X with the same code, leading to more repetitiveness, among other things.

The method which was implemented into Network Guardian that made the final cut was utilizing annotations like the “register_plugin” function. With the use of a class meta data, and an annotation, any function within a registered plugin with an executor annotation above it with the correct arguments, will registered as a suitable method of processing the plugin.

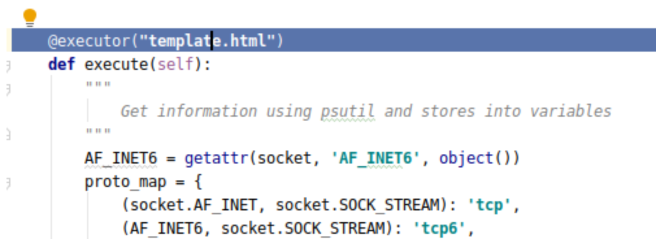
The code below is called right when the plugin is loaded into the python environment, looping through all functions in the class, finding functions marked with a "_platforms" attribute, and then adding it to a dictionary of executors.



```
class MetaPlugin(type):
    """
    Metaclass modifies the class-creation behavior
    """

    def __new__(mcs, name, bases, attrs):
        """ This is called when the class is loaded in python before the creation of the instance """
        executors = {}
        for fn_name, fn in attrs.items():  # for each NAME, FUNCTION in class
            if inspect.isfunction(fn):  # if it's a function
                supported_platforms = getattr(fn, '_platforms', None)  # if its been marked by the annotation get value
                if supported_platforms is not None:  # if there's some platforms supplied
                    for sp in supported_platforms:  # for each platform supplied
                        executors[sp] = fn  # add the function to the _executors dict

        attrs["_executors"] = executors

        return type.__new__(mcs, name, bases, attrs)
```

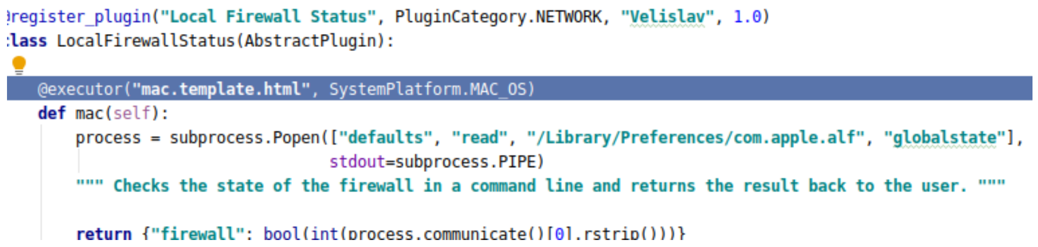
Figure 3: Python code used to register executors within a plugin class

The following code is an example of how a plugin would utilize the plugin executors for specific cases.



```
@executor("template.html")
def execute(self):
    """
    Get information using psutil and stores into variables
    """
    AF_INET6 = getattr(socket, 'AF_INET6', object())
    proto_map = {
        (socket.AF_INET, socket.SOCK_STREAM): 'tcp',
        (AF_INET6, socket.SOCK_STREAM): 'tcp6',
```

Figure 4: Executor specifying all operating systems by default (no platform argument specified)



```
@register_plugin("Local Firewall Status", PluginCategory.NETWORK, "Velislav", 1.0)
class LocalFirewallStatus(AbstractPlugin):

    @executor("mac.template.html", SystemPlatform.MAC_OS)
    def mac(self):
        process = subprocess.Popen(["defaults", "read", "/Library/Preferences/com.apple.alf", "globalstate"],
                                   stdout=subprocess.PIPE)
        """ Checks the state of the firewall in a command line and returns the result back to the user. """

        return {"firewall": bool(int(process.communicate()[0].rstrip()))}
```

Figure 5: Executor specifying specific operating system

When a plugin is called to generate information for a report, the software calls the dictionary with the running operating system as a key, and the subsequent data for that specific operating system is returned. We believe this preferred method of registering an executor further satisfies the Maintainability and Usability factors of ISO 25010.

Further documentation has been provided specifically in the form of a plugin development guide and can be seen in Appendix C to assist plugin developers attempting to create there first plugin.

3.1.2 Report Generation

Report Generation was a particularly difficult element of the software to develop due to several constraints and requirements that had to be met.

One of the main requirements of the report generation is that it had to be able to handle and support a wide range of data outputs to fully support the requirements of different plugins, and future plugins that end users may wish to implement.

Plugins may require outputting any types of data, from bytes, to images, to just simple plaintext, it therefore needed to be flexible enough to meet the developer's needs.

3.1.2.1 Report Storage

Reports by default are stored in the Network Guardian folder created on initialization of the software. When a report is created, the software stores a serialized object of the Report which is stored to the disk.

When deciding on the method of storing, several options were considered but the final decision was to use serialization. Serialization is the process of converting a running object or data structure in code, such as a class or an array to a stream of bytes so it can be written to a file.

Python’s standard library includes a module called Pickle, which is used for serialization and deserialization of objects. The advantage of using serialization is that it requires little code, and therefore was implemented with ease. The use of pickling however carries one serious side effect, and disadvantage which comes in the form of a security vulnerability.

The vulnerability is made possible due to a weakness in Python’s pickle library. The attack takes form when an attacker crafts a malicious report file. For example, an attacker could serialize a similar class to the exported report object and change the methods to malicious ones such as a reverse shell, or malware dropper, then if an unsuspecting user of the software loads the report from the attacker, the exploit will be executed.

Though the exploit requires a legitimate user to run the modified report file, and could potentially require a form of social engineering, i.e. the attacker sending the file over email asking the user to view the results, the fact the vulnerability exists is dangerous. Further discussion on this exploit, as well as two counter measures can be found in section 6.3.4.

3.1.2.2 Exporting to HTML

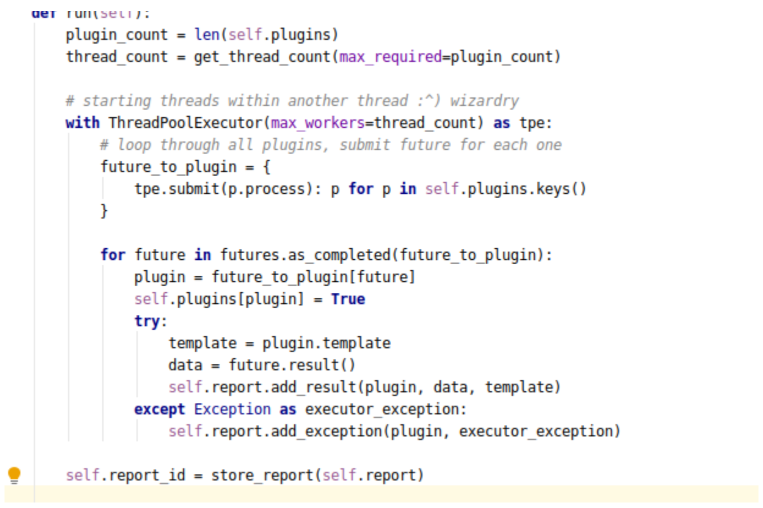
Network Guardian further provides the capability to export the generated report files to HTML so that they can be viewed in a web browser. This functionality was implemented to simplify the viewing of reports, as this feature would allow users of the software to send the generated reports to others without the program. This feature could save time when writing reports in a network assessment paper, as it could be simply included in the appendices.

This feature was implemented with ease, due to the way plugins render the reports as they are already in HTML.

3.1.2.3 Multithreading

To further increase the efficiency and speed of execution, Network Guardian automatically runs plugins when executed within a multithreaded pool. Multithreading allows instructions to be run in parallel, for up to as many logical and virtual CPU cores stored within the system. The software further includes the capability to set the maximum threads allowed if the end user requires, to free up and reduce CPU usage during report generation.

This feature dramatically increases the speed of scanning as often certain enumeration techniques can take considerable time, running them in parallel allows the software to complete multiple tasks at once. This complements the Performance Efficiency requirements in our quality models.



```
def run(self):
    plugin_count = len(self.plugins)
    thread_count = get_thread_count(max_required=plugin_count)

    # starting threads within another thread :^) wizardry
    with ThreadPoolExecutor(max_workers=thread_count) as tpe:
        # loop through all plugins, submit future for each one
        future_to_plugin = {
            tpe.submit(p.process): p for p in self.plugins.keys()
        }

        for future in futures.as_completed(future_to_plugin):
            plugin = future_to_plugin[future]
            self.plugins[plugin] = True
            try:
                template = plugin.template
                data = future.result()
                self.report.add_result(plugin, data, template)
            except Exception as executor_exception:
                self.report.add_exception(plugin, executor_exception)

    self.report_id = store_report(self.report)
```

Figure 6: Code used to run plugin executors with multithreading

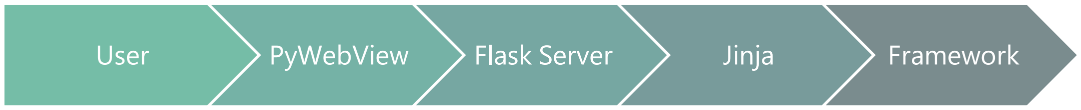
3.1.3 User Interface

The development of the user interface consisted of three key stages, Planning and Design, Creation of the Model View Controller and Integration.

3.1.3.1 User Interface Framework

During the project planning and proposal phase of the development of Network Guardian, research was completed regarding the user interface framework which would be used to render the user interface.

The way the user interface works within Network Guardian relies on two key third party python libraries.



Jinja

Jinja is a python library which was used for its template inheritance functionality. This allowed a base to be created with all the common features and elements within the application, such as the header and navigation bar and therefore *blocks* were used to implement the different aspects of the software, essentially the content of each page.

Jinja is further used in plugins, where Jinja is used to render the output data into the plugin template file.

Flask

Flask is a micro web framework written in Python, which was used to display and host Network Guardian. Flask was used as it is very lightweight, easy to use and works in conjunction with Jinja. Flask is essentially a lightweight web server which hosts all the template files generated with Jinja.

3.1.3.2 Model View Controller

Network Guardian using Flask allows the use of a Model View Controller (MVC) design pattern. An MVC is a software design pattern which aims to link the functions called within the GUI to the functions within the software. Network Guardian uses Flask to provide the interaction with the MVC and then all subsequent actions are processed by the framework.

3.1.3.3 Encapsulating the GUI within a Window

PyWebView is a python library, that allows you to display HTML content within its own GUI application, this is also cross-platform, which is perfect for our application. PyWebView was selected as it is lightweight and can encapsulates the flask server which is being run to host Network Guardian.

3.1.3.4 GUI Creation

When creating the Graphics User Interface (GUI), initial designs were created using wireframes. These were then developed further, and each aspect had a function or use to prevent 'gold plating' and improve efficiency and time management. The wireframes can be seen in Appendix D.

The initial pages were then created in pure HTML only and then the integration with jinja and the mode view controller began.

Once the pages were linked, features such as 'breadcrumbs' were added. And then when the plugins were completed, the GUI was then implemented with the framework. As the GUI was integrated, features such as a 'Settings' page was then added, to make the application more realistic and usable. The GUI was cleaned up by changing the icon images, button shapes and sizes.

3.2 Plugin Development

Plugin development took place in parallel with the development of the software, and user interface to increase the efficiency of the development. As the plugin system was developed by one member of the team, the team members developing the plugin's worked on the plugin list in order of easiest to most difficult, this allowed members of the team to be comfortable with the system and be aware of its capabilities and constraints.

3.2.1 Plugin Selection

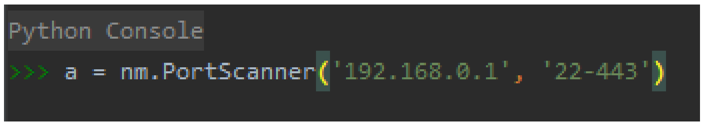
The plugins and their method of enumeration implemented into the software were decided on during the project proposal planning stage before development began on the software. Throughout implementation the team followed guidance from two highly regarded security testing materials, the Open Source Security Testing Methodology Manual and the Penetration Testing Methodologies and Standards. A further evaluation of the implemented can be found in section 5.2.

3.2.2 NMAP

One of the most significant plugins implemented into the Network Guardian software the numerous NMAP Plugins. This utilizes a third party library to integrate the standard NMAP functions into python.

The first stage of setting up the plugin was downloading NMAP from nmap.org, so that it was on the system ready to use, then in python "import NMAP" was used to import the NMAP library - this was the extent of the set up required and meant that it was now ready to be used.

During initial testing, the code below was used as this would create a NMAP scan with the parameters set it the brackets. This meant that it would scan the current machine on the ports 22-443 and then copy these results into the variable "a". The results were already well laid out, so it was easy to navigate through and find out all the information about the scan.



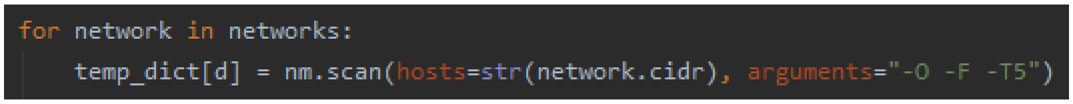
```
Python Console
>>> a = nm.PortScanner('192.168.0.1', '22-443')
```

Figure 7: Code to initiate NMAP Port scan in Python

As there was not a lot of documentation surrounding the library and how to use NMAP in python, it did take some time to learn how to use different parameters and how this information would be stored. After dedicating some time to understanding this however it was quite easy to carry out scans and store this information.

The plugins main objective however was to scan whole networks, not just a single machine. The first thing needed for this crucial advancement was a function that could gather the network addresses. A function was created to get the network addresses from each network interface, which then runs a validation check to eliminate of all the addresses that were irrelevant (such as loopback addresses or link local addresses). It would then take these approved addresses and calculate CIDR Addresses.

These CIDR addresses are then passed into the main code where they would be looped through and scanned, with each result being stored into a dictionary. The code "temp_dict[d] = nm.scan(hosts=str(network.cidr), arguments= " -O -F -T5") was used, which would carry out a scan of the current CIDR with arguments O,F and T5. -O meant that NMAP would try to guess the operating system of the system, -F meant it would scan the top 100 ports and T5 meant it would scan very quickly.



```
for network in networks:
    temp_dict[d] = nm.scan(hosts=str(network.cidr), arguments="-O -F -T5")
```

Figure 8 Main NMAP code used to scan networks

These arguments were used for an efficient quick scan so that it would not take too long to scan the entire network. This information was then put into a template using jinja so it could loop through the dictionary for each network scan.

3.2.3 Network Visualization

The Network Visualization plugin was an addition and modification to the original brief which was requested by the client in Week 7. The Network Visualization plugin was a complex tool to implement because it required both the combination of multiple technologies, and decent pre-existing networking knowledge.

To provide the functionality that the client required, three steps had to be taken place before the formatting could even begin. The first step is host discovery which was taken care of with NMAP, by gathering the available network interfaces on the machine and calculating the subnet range CIDR, NMAP was used to find all available hosts.

After finding the available hosts, NMAP would then do a more in-depth scan of the available hosts, attempting to verify the Operating system and device information, gathering a list of open ports, and producing a vulnerability scan.

Finally, after that was complete, a trace route was completed to verify the location of the device on the network, in relation to the internet, the routers, and machine it was scanned on. This was done by looking at the hops.

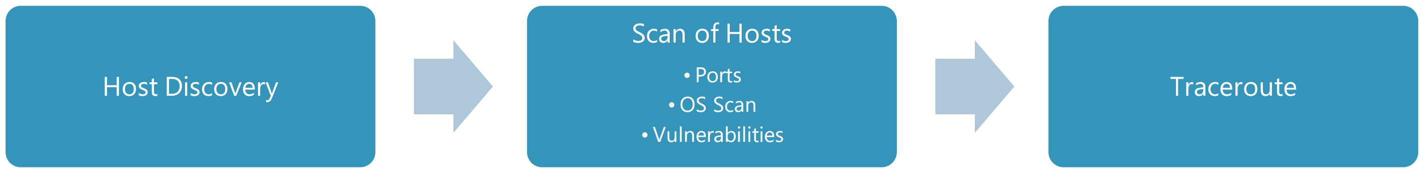


Figure 9: Process of Providing Information for Network Visualization

To render and provide the actual graphical representation of the network, another technology was implemented. D3 is a JavaScript library which is used to produce dynamic and interactive data.

As HTML and JavaScript is how plugins are rendered within the GUI, this was chosen as the desired choice of rendering the data. Originally the team planned to stitch image files together to produce a static image, however the team felt this could be improved on. D3 although initially looking easy to implement, took considerable time as the documentation was quite weak.

3.3 Application Packaging and Cross Compatibility

Network Guardian, being a cross compatible software package needed to be ran on all major operating systems, therefore careful consideration had to made when developing areas of the software to ensure it would run on all systems.

3.3.1 Packaging

Freezing or Application Packaging, is a term coined to a process in python where the running environment, I.e. the python binaries, as well as the python code is compiled into a single binary which can be ran on any machine without the need to install Python.

The advantage of distributing the software in this format is the application "just work", even if the user doesn't already have the correct (or any) version of Python installed. Besides, end-user software should always be in an executable format. Files ending in ".py" are for software engineers, and following ISO 25010, this would not meet the requirements as it would fail the portability and usability factors in our quality models.

The capability to do this, increasingly improves the portability of the software, as users can easily place the binary on a removable memory format such as a USB stick, and then go to any machine to complete an assessment.

However, several issues can occur when freezing, this is due to errors occurring with certain parts of code, especially with third party libraries which do not support the process. Several freezing utilities were tested with the software including "bbFreeze", "py2exe", and "cxFreeze", until the team decided on using PyInstaller which was found to provide no errors during execution on any of the desired platforms. It also provided the added ability to add metadata such as a description, and icons to the executables.

With the use of PyInstaller, the latest version of Network Guardian has been compiled into three separate executable binaries, one for each platform supported by the software.

Executable Name:	**Platform**
Network Guardian.exe	Windows
Network Guardian.dmg	Mac OS X
Network Guardian.bin	X86 Linux

Table 1: Packed Executables Filename's and Platform

3.4 Ensuring Quality of Development

Several key development strategies were devised during planning to ensure that the product developed by Team Pentagon would be a high standard of quality.

3.4.1 Version Control

GitHub with Git was used as desired choice of version control as it provided a free online host of the code, allowing the developers to work on the software from anywhere, with any suitable device. GitHub also provided several other notable features which were used to assist with the development, such as the code review stage, and contribution statistics.

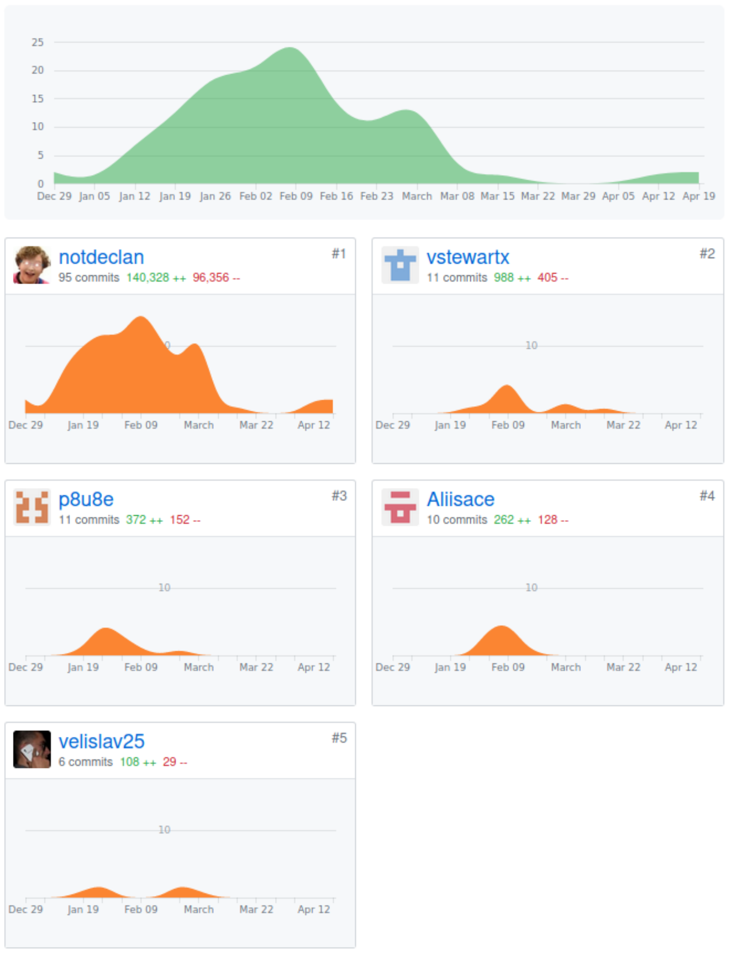


Figure 10: Screenshot from GitHub Insights displaying developers' statistics

3.4.1.1 Branch Strategy

A Branching Strategy was devised before beginning development to both simplify and improve the development workflow. A Branch provides a way of sectioning off a part of development, this is to allow developers to work on parts separately without interfering with their code among other things. It also allows separate versions of the software to be available at one time.

During the development of Network Guardian, the team devised two main branches, Master, and Development. The sole purpose of the two branches was to separate the running, most up to date stable version of the software which the client could use at any point from the potentially bug ridden, and unstable Development branch.

The development branch then had offshoots of itself, for separate features and bug fixes. Separating into separate branches also provides the capability to revert the changes of the branch at a future date if issues arise. For example, if a branch which implemented a certain feature happened to break the application, or create bugs, it can simply be removed from the software with the click of a button.

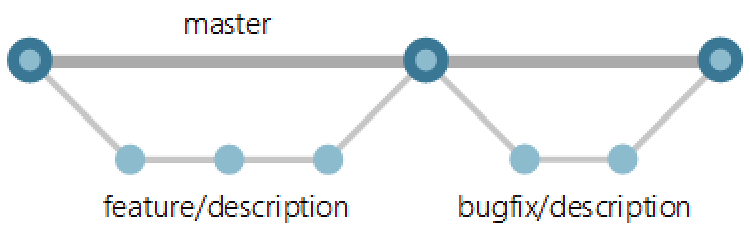


Figure 11: Branching example diagram (Microsoft, 2015)

When code inside the Development branch is stable, and suitable for a release, it was merged into the Master branch.

3.4.2 Following Python Coding Convention Standards

Throughout the development of Network Guardian, the developers ensured to follow the strict code style conventions included in Python's official enhancement proposals. PEP 8 focuses on all aspects of the development of a Python application, setting standards in a full stack manner, from code indentation, to naming conventions, and more.

The developers followed the coding convention standards to overall improve and maintain consistency throughout the application, so that in future the project could be adapted and improved upon by alternate developers.

3.4.3 Code Review

Utilizing our Version Control System, a code review stage was implemented into the projects agile methodology.

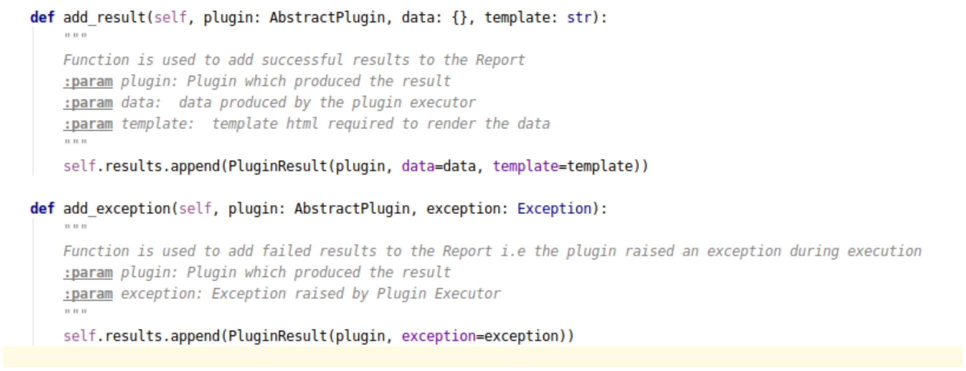
A code review would take place between any merge request from a development off-branch into both the development branch, and the master branch. The main aim of the code review stage was to prevent any poorly written code from being implemented into the project. As well as this, would improve the future proofing of the code, as poorly written code is considerably harder to work on in the future, especially by alternate developers.

As laid out in the project proposal, the client was given access to the master branch and had the capability to use the software from early stages of development. This is partly the reason why it was key to ensure that only working code was implemented into the master branch.

During the code review stage, the teams head developer would critique code in the pull request, noting areas which were considered inappropriate for implementation. Various things were critiqued, such as the code conventions, reliability, and readability, as well as if the code had all the required and relevant unit testing. Various updated pull requests and merge requests will then entail until the head developer, and the team was happy to implement the changes.

3.4.4 Code Commenting

All code implemented into the software was commented to the Python Official comment convention standards to improve the readability of the code for future developers. This besides making the software easier to update and improve, will assist the plugin developers into understanding how the software works, to further improve the efficiency and the capability of their plugins.

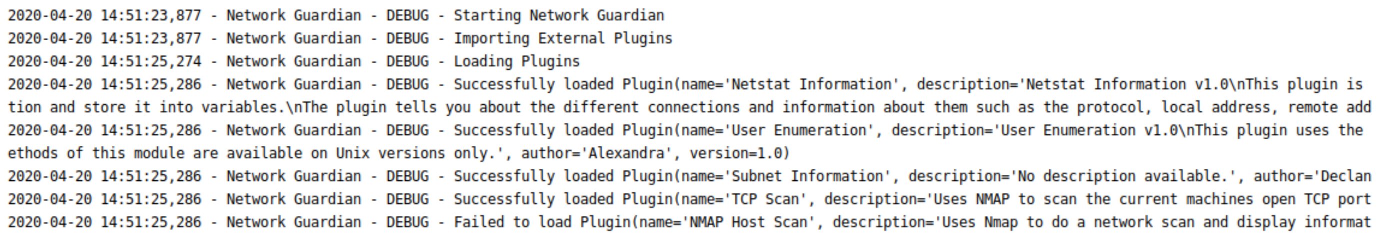


```
def add_result(self, plugin: AbstractPlugin, data: {}, template: str):
    """
    Function is used to add successful results to the Report
    :param plugin: Plugin which produced the result
    :param data:  data produced by the plugin executor
    :param template:  template html required to render the data
    """
    self.results.append(PluginResult(plugin, data=data, template=template))

def add_exception(self, plugin: AbstractPlugin, exception: Exception):
    """
    Function is used to add failed results to the Report i.e the plugin raised an exception during execution
    :param plugin: Plugin which produced the result
    :param exception: Exception raised by Plugin Executor
    """
    self.results.append(PluginResult(plugin, exception=exception))
```

Figure 12: Example of code commenting

3.4.5 Logging

Logging was implemented throughout the software using python's inbuilt library "logging". Logging is a key step in debugging the software when things are not working as expected. Utilizing the inbuilt command line interface, users of the software can change the logging verbosity to view debugging messages, and regular running messages.



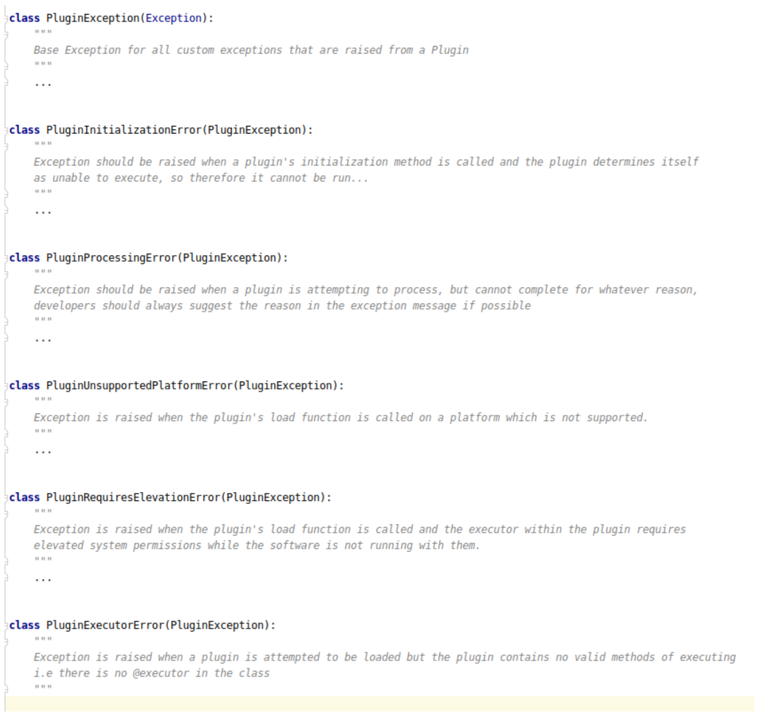
```
2020-04-20 14:51:23,877 - Network Guardian - DEBUG - Starting Network Guardian
2020-04-20 14:51:23,877 - Network Guardian - DEBUG - Importing External Plugins
2020-04-20 14:51:25,274 - Network Guardian - DEBUG - Loading Plugins
2020-04-20 14:51:25,286 - Network Guardian - DEBUG - Successfully loaded Plugin(name='Netstat Information', description='Netstat Information v1.0\nThis plugin is
tion and store it into variables.\nThe plugin tells you about the different connections and information about them such as the protocol, local address, remote add
2020-04-20 14:51:25,286 - Network Guardian - DEBUG - Successfully loaded Plugin(name='User Enumeration', description='User Enumeration v1.0\nThis plugin uses the
ethods of this module are available on Unix versions only.', author='Alexandra', version=1.0)
2020-04-20 14:51:25,286 - Network Guardian - DEBUG - Successfully loaded Plugin(name='Subnet Information', description='No description available.', author='Declan
2020-04-20 14:51:25,286 - Network Guardian - DEBUG - Successfully loaded Plugin(name='TCP Scan', description='Uses NMAP to scan the current machines open TCP port
2020-04-20 14:51:25,286 - Network Guardian - DEBUG - Failed to load Plugin(name='NMAP Host Scan', description='Uses Nmap to do a network scan and display informat
```

Figure 13: Example of logging during program start-up with logging level set to debug mode

3.4.6 Strong Exception Handling

Exception Handling is the process of responding to the occurrence of exceptions during the execution of software. Handling exceptions prevents the software from crashing or causing unexpected behaviour. Implementing custom exceptions further improve the reliability, while also providing more detail into what caused the exception.

During the implementation of Network Guardian, six different custom exceptions were added, handled and commented.



```
class PluginException(Exception):
    """
    Base Exception for all custom exceptions that are raised from a Plugin
    """
    ...


class PluginInitializationError(PluginException):
    """
    Exception should be raised when a plugin's initialization method is called and the plugin determines itself
    as unable to execute, so therefore it cannot be run...
    """
    ...


class PluginProcessingError(PluginException):
    """
    Exception should be raised when a plugin is attempting to process, but cannot complete for whatever reason,
    developers should always suggest the reason in the exception message if possible
    """
    ...


class PluginUnsupportedPlatformError(PluginException):
    """
    Exception is raised when the plugin's load function is called on a platform which is not supported.
    """
    ...


class PluginRequiresElevationError(PluginException):
    """
    Exception is raised when the plugin's load function is called and the executor within the plugin requires
    elevated system permissions while the software is not running with them.
    """
    ...


class PluginExecutorError(PluginException):
    """
    Exception is raised when a plugin is attempted to be loaded but the plugin contains no valid methods of executing
    i.e there is no @executor in the class
    """
```

Figure 14: Custom Exception Handling classes for Plugins

3.5 Following Legal Guidelines

3.5.1 Third Party Libraries

As noted in the original proposal, to ensure that Network Guardian, the client, and the development team would not face any legal implications throughout the creation of the product, careful consideration was taken when deciding on the usage of third-party libraries.

Open Source Libraries provide quick and easy access to vast amounts of quality code, when developing software, it is often not needed to write everything from scratch, because someone has often already done it. Why reinvent the wheel if you don't have to? This not only speeds up the development process, but in a team where not all the developers are totally competent, third party libraries allow the use of difficult to implement code without developing it themselves.

When utilizing third party libraries there are often caveat's however, as most open source libraries come with a required license which needs to be followed. The licenses state how the code can be used, modified and implemented in commercial software, and can have serious legal implication's if not followed correctly.

Therefore, throughout the development the Network Guardian team paid careful consideration when choosing libraries and followed all the stated requirements in the licenses. Alongside the software, the licenses and copyrights for all third-party libraries used were included as required.

All included third party libraries with their respective copyright tags can be found in Appendix G, alongside the specific license it requires. Furthermore the entire license file is included with the source code as required by some of the licenses.

4 Testing

To ensure that the software Team Pentagon has developed works, the team set out to create a solid method that could replicate the real-life use cases of our product, Network Guardian. The team set out to do this by creating several test network environments, where the set characteristics of the network could be accurately controlled.

This allowed the team to see if Network Guardian would produce accurate results over several machines with different operating systems.

4.1 Creating Test Environment

Several options were considered for the test environment of Network Guardian including emulation with "CORE Emulator" and virtualisation using 'VMware's vSphere', however these options did not fully meet our requirements for building a test network for Network Guardian. The team settled on creating a test environment by building virtual machines and manually connecting them in a virtual network consisting of multiple virtual desktops, servers and routers across 5 subnets (see diagram 4.1a: Network Diagram on the next page).

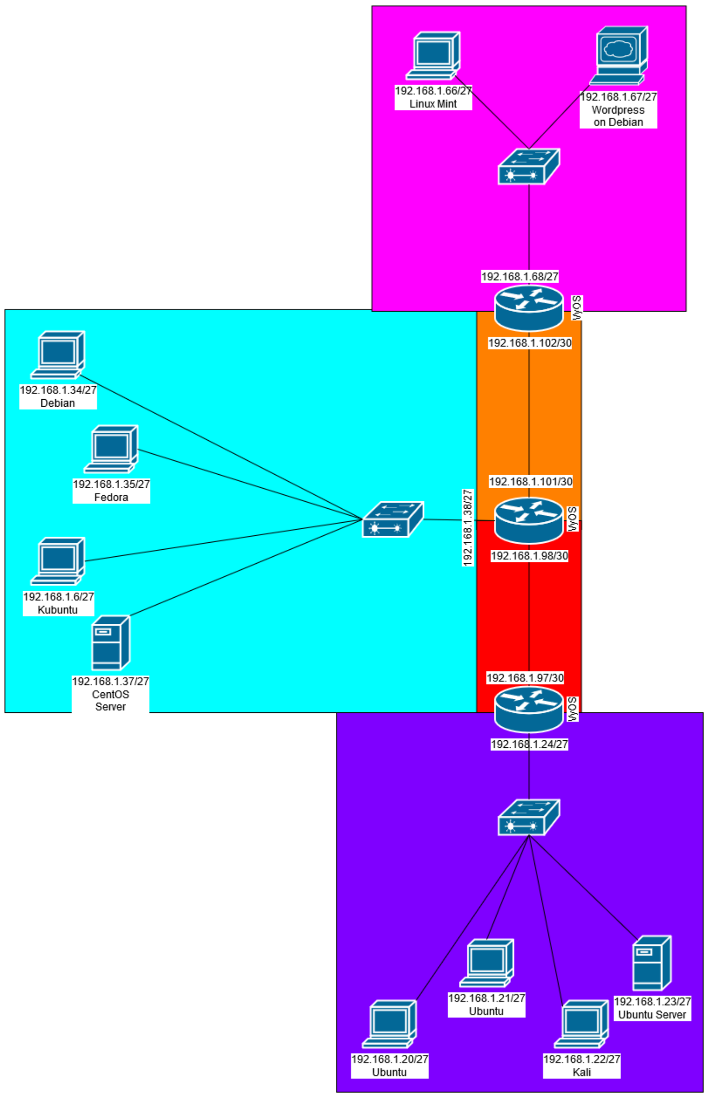
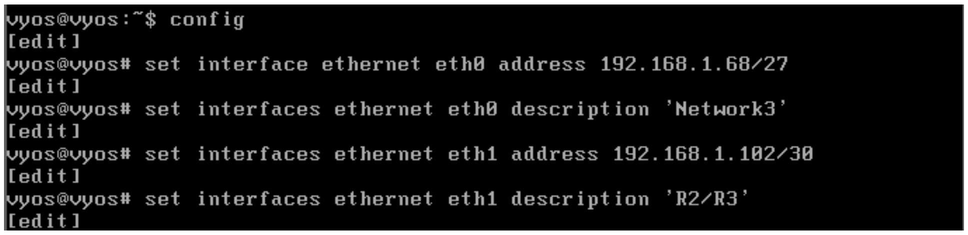


Figure 15: Network Guardian Test Environment Network Diagram

The test network was built by downloading installation disc image file of multiple operating systems and building virtual machines in 'VMWare Workstation'. The virtual machines were built using the 'New Virtual Machine Wizard' in VMWare Workstation and selecting the required installation disc image file of the operating system (see Appendix I for screenshots of the virtual machine set up).

The hosts consisted of Windows 10 & a variety of Linux distros including eight based on Debian (four Ubuntu (including one Kubuntu), one Kali, two Debian and one Linux Mint) and two rpm-based (Fedora and CentOS) to provide a more thorough test of Network Guardian on different Linux distros.

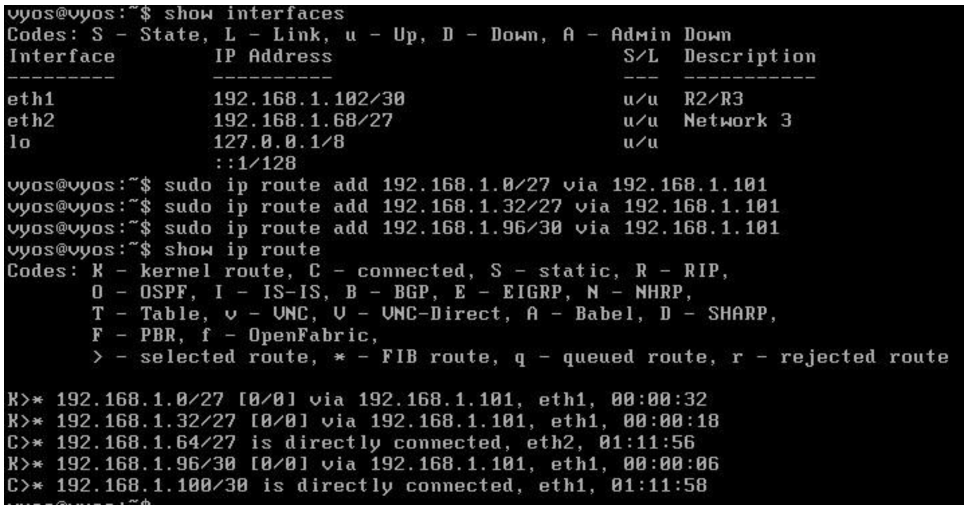
The hosts were then configured into subnets connected to three routers (running VyOS) by running a script which set the IP address, default route, and added IP routes to the hosts (see figure 14 for the network diagram and figures 15-16 for configuration of Router 3 and Appendix J for Router 1 & Router 2 configuration).



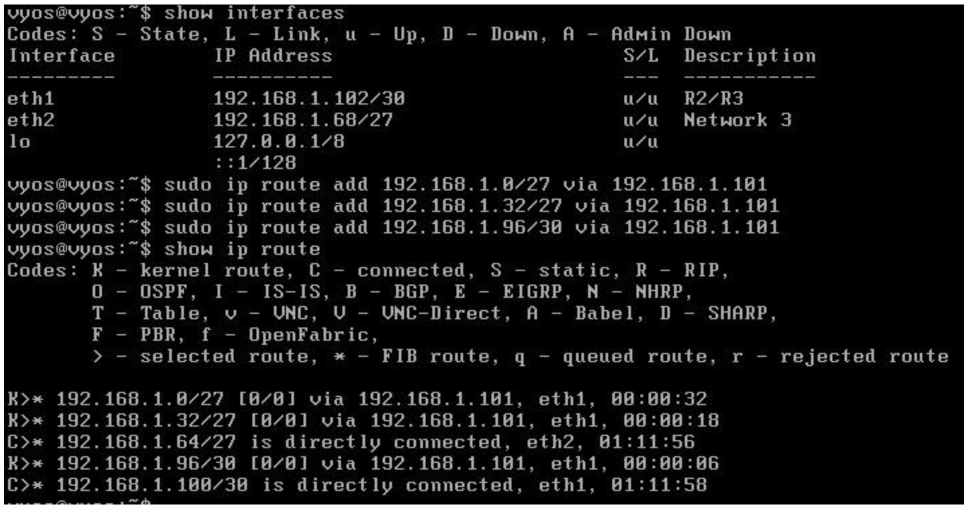
```
vyos@vyos:~$ config
[edit]
vyos@vyos# set interface ethernet eth0 address 192.168.1.68/27
[edit]
vyos@vyos# set interfaces ethernet eth0 description 'Network3'
[edit]
vyos@vyos# set interfaces ethernet eth1 address 192.168.1.102/30
[edit]
vyos@vyos# set interfaces ethernet eth1 description 'R2/R3'
[edit]
```

```
vyos@vyos:~$ config
[edit]
vyos@vyos# set interface ethernet eth0 address 192.168.1.68/27
[edit]
vyos@vyos# set interfaces ethernet eth0 description 'Network3'
[edit]
vyos@vyos# set interfaces ethernet eth1 address 192.168.1.102/30
[edit]
vyos@vyos# set interfaces ethernet eth1 description 'R2/R3'
[edit]
```

Figure 16: Configuration of Interfaces of Router 3



```
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                          S/L  Description
---------        ----------                          ---  -----------
eth1             192.168.1.102/30                    u/u  R2/R3
eth2             192.168.1.68/27                     u/u  Network 3
lo               127.0.0.1/8                         u/u
                 ::1/128
vyos@vyos:~$ sudo ip route add 192.168.1.0/27 via 192.168.1.101
vyos@vyos:~$ sudo ip route add 192.168.1.32/27 via 192.168.1.101
vyos@vyos:~$ sudo ip route add 192.168.1.96/30 via 192.168.1.101
vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

K>* 192.168.1.0/27 [0/0] via 192.168.1.101, eth1, 00:00:32
K>* 192.168.1.32/27 [0/0] via 192.168.1.101, eth1, 00:00:18
C>* 192.168.1.64/27 is directly connected, eth2, 01:11:56
K>* 192.168.1.96/30 [0/0] via 192.168.1.101, eth1, 00:00:06
C>* 192.168.1.100/30 is directly connected, eth1, 01:11:58
```



```
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth1             192.168.1.102/30                  u/u  R2/R3
eth2             192.168.1.68/27                   u/u  Network 3
lo               127.0.0.1/8                       u/u
                 ::1/128
vyos@vyos:~$ sudo ip route add 192.168.1.0/27 via 192.168.1.101
vyos@vyos:~$ sudo ip route add 192.168.1.32/27 via 192.168.1.101
vyos@vyos:~$ sudo ip route add 192.168.1.96/30 via 192.168.1.101
vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

K>* 192.168.1.0/27 [0/0] via 192.168.1.101, eth1, 00:00:32
K>* 192.168.1.32/27 [0/0] via 192.168.1.101, eth1, 00:00:18
C>* 192.168.1.64/27 is directly connected, eth2, 01:11:56
K>* 192.168.1.96/30 [0/0] via 192.168.1.101, eth1, 00:00:06
C>* 192.168.1.100/30 is directly connected, eth1, 01:11:58
```

Figure 17: Interfaces of Router 3 + IP routes

While this setup was more complicated and resource heavy (in terms of both time and computer processing) than other options explored, it enabled more detailed and thorough testing of Network Guardian due to the ease of modifying machine settings.

These connections were repeatedly tested by running another script that pinged all IP addresses on the test network.

A script was written to open & close different ports before each test run to enable a variety of different conditions in the test network.

The full subnet information, as well as information relating to all the virtual machines can be found in Appendix H.

4.2 Testing Strategy

After the test environment was created Network Guardian was run on several configurations of the network, with different machines up and connected in a variety of configurations on different Operating Systems.

After the test environment was created Network Guardian was run on several configurations of the network, with different machines up and connected in a variety of configurations on different Operating Systems.

5 Discussion

5.1 Discussion of Testing Results

The results of our testing concluded that our software could accurately provide the team with the information we expected it to produce; information that the team believes will increase the productivity of Network Administrators and Security Penetration testers.

The results produced from the software were accurate when compared to other methods of obtaining the same information from other sources – e.g. the list of open ports in the Network Guardian matches the open ports list from running the netstat command on the terminal.

5.2 Evaluation of Implemented Enumeration Techniques and Benefits

An evaluation of the implemented enumeration techniques is a key step in ensuring that the developed product is of use to the end users, and notably the client. As highlighted in the background section, the main aim and requirement of the product is to assist in the enumeration stage of a network assessment, therefore it is important to evaluate the usefulness of the plugins included within the product to validate the products usefulness.

In the following sub sections, the usefulness the included plugins will be quantified against why or why not they may be required within a network assessment, and finally concluding on whether when used together a network administrator or pen-tester could utilize the tools to assist in their workflow.

5.2.1 Check Internet Connectivity

The state of whether a device has available internet access has a significant impact on the need of security measures in place. When a device connected to the internet, it then becomes vulnerable from external attacks.

Although checking internet connectivity is an easy step, the ability to automate the feature and to include it in the report, saves time when completing an assessment, and therefore provides benefit.

5.2.2 Local Firewall Status

The Local Firewall Status plugin automates the process of checking whether firewall is enabled on Windows, and Mac OS X by querying the local system and checking if the default firewalls are enabled. Due to different firewalls obviously having different interfaces it would be difficult to determine if every firewall software is enabled, therefore during our implementation we only supported these two.

Despite only checking these two firewalls, the ability to quickly check whether they are enabled might be enough for most network administrators. Furthermore, due to the ability to add plugins, Network Administrators could modify the existing plugins too add capability for their specific firewall software.

5.2.3 NetStat Information

NetStat is a command line utility found across all the operating systems supported by Network Guardian, used to examine real-time network connections. The purpose is to provide a list of network connections and their respective statistics, as well as information such as the protocol, the local and remote addresses, and the process name.

NetStat information is specifically useful for a network administrator which is attempting to verify the running applications connected to the internet. Having

5.2.4 Network Interface Information

Having all the required information and statistics about the Network Interface devices installed on the local machine helps give the end users of the software information about all the potential areas of exploration during the assessment.

5.2.5 Network Visualization

The network visualization plugin provides a great benefit to the end users of the software. The ability to create a pseudo network diagram on the fly can dramatically improve the understanding of a network. The included capability to highlight the potentially vulnerable machines also saves time and has the possibility of reducing the time a vulnerability on a network lays dormant.

6 Evaluation

6.1 Execution of the plan

At the start of the project it was already agreed that we were going to meet up twice a week and during this time we were going to work on the project for several hours, ensuring we were all on track and working on the project. The first couple of weeks were about setting up our developing environments and carrying out research. We set up PyCharm, GitHub and Microsoft Teams, this would help the rest of the development to run smoothly.

The execution of our project plan was mostly on track up until week 11, this is due to the COVID-19 situation. At this point, the product testing, whitepaper and proposal presentation still needed to be completed. The testing was assigned to Alexandra, as she had been working on the testing environment for the previous 7 to 8 weeks. Different sections of the whitepaper were then assigned to everyone, specifically if they had worked on that area, for example the creation Graphic User Interface was assigned to Stewart and the Framework section was assigned to Declan. The presentation was then carried out by the team members that made themselves available, these being Stewart, Declan and Velislav.

The execution was delayed around week 10/11 not only due to the COVID-19 pandemic, but also because the testing environment was still not completed or usable. This delayed everything else as the whitepaper and presentation required testing results, to allow the team to support and back their product. Multiple reasons were given to try and excuse the fact the testing was not completed such as a frozen laptop or the team member was busy. This was dealt with through out the project but was harder to work around due to the COVID-19 pandemic as the team could no longer meet up to ensure work was getting done. Multiple team members became lazy and disengaged making it difficult to complete the project to the highest standard.

6.1.1 Software Quality

The software quality measures implemented in section 3.4, were carefully considered before beginning the development and were regularly reviewed to ensure that the measures were in place.

6.1.2 ISO 20510 Compliance

As mentioned in previous sections above, Team Pentagon throughout development strived to follow guidance from the international standard ISO 25010, a compliance which aims to improve the quality of systems and software by providing which provides quality models. The compliance provides a product quality model composed of eight characteristics that relate to static properties of software.

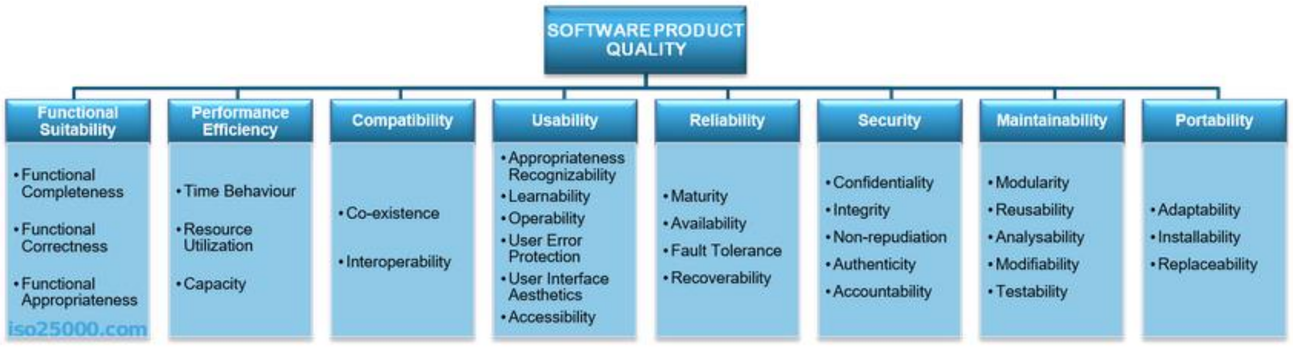


Figure 18: Eight quality characteristics specified in ISO 20510 (ISO 25000, n.d.)

6.2 Issues

6.2.1 Issues with Team members

There were a couple of issues that team Pentagon faced during the development of Network Guardian, one of these issues was following the Gantt chart and completing some of the tasks on time, such as the Testing Environment as well as starting the whitepaper, some of the time lost was made up on other areas of the project such as plugins and the graphics user interface (GUI) as most of the plugins were straightforward and once the initial GUI was created, tweaks and changes that were needed were made throughout.

An obvious issue that the team faced was continuing the development through the COVID-19 pandemic. This prevented the team from meeting two to three times a week to carry out work together. These sessions were when the team were most productive and efficient as everyone could work together, and we knew what everyone was doing. This ensured that everyone was carrying out what they were supposed to and working on Network Guardian.

6.2.2 Changes to the specification

During week 7, a meeting with Dr Ethan Bayne (Client) was held to update him on the progress of the project and he then requested that a plugin that visualises the network was implemented into the software. Due to the plugin design of the software, this was not too hard to implement, the only issue was finding time and carrying out the research to do this.

6.3 Future Work

6.3.1 Software

There is plenty of room for further development of Network Guardian. The initial improvements would be to include additional plugins, meaning the end user wouldn't have to carry out their own testing when creating their own plugins as this would already have been done before being released to the users.

Additional features could be implemented into Network Guardian, such as a Vulnerability Scoring based off "The Common Vulnerability Scoring System" more commonly known as CVSS. This was discussed in the Proposal, and was found through market research, as the team thought this would be a great addition to the software. This feature would essentially mark different information found with a score, providing an indication of how important or vulnerable certain data is. The higher the score the more the vulnerable the network would be, giving the end user a clear indication of vulnerabilities within the network. We would adapt the CVSS version 3.0 standards and add an "Advisory" category for non-vulnerabilities. The table below displays the Severity level and the characteristics we would implement.

SEVERITY	CHARACTERISTIC
CRITICAL	CVSSv2 score is 10.0.
HIGH	CVSSv2 score is between 7.0 and 9.9.
MEDIUM	CVSSv2 score is between 4.0 and 6.9.
LOW	CVSSv2 score is between 0.1 and 3.9.
ADVISORY	CVSSv2 score is 0 or non-vulnerability.

Table 2: CVSS Advisory Scoring

Another function that would be implemented into Network Guardian would be the capability to export the reports as a PDF and HTML file. The program can currently export the report as an HTML but implemented the ability to export as a PDF will give the user options. This function allows the reports to be conveniently transmitted without requiring Network Guardian to view them.

Another feature to be added to Network Guardian would be an option in the settings to store the root credentials, as this would allow current plugins (Local Firewall Status) and future plugins to work on all operating systems, as these are required for some enumeration techniques. Being able to set the root credentials would further improve the quality of the report and efficiency of the software.

6.3.2 Testing

To improve the testing of Network Guardian, additional and more complex virtual networks should be created to further test the limits and capabilities of the software, to ensure it is efficient and accurate. More complex networks would include routers, servers and more workstations, with firewalls and different ports open and closed, this would further ensure the accuracy of the product.

An alternative testing procedure would be to measure the time taken to carry out the enumeration stage manually and compare it to the use of Network Guardian. This would give a better visual representation on how time efficient the software is and would provide an accurate representation of how much using the software would improve the workflow of the end users.

A final addition to the testing procedure would be to improve the evaluation of the implemented enumeration techniques, by using the software to perform a real network assessment. This would further prove the usefulness. This could also be accompanied by feedback from Network Administrators and Pen testers, to gauge a review of the usefulness from outside the development team which by nature have a bias, and a greater understanding of the software's abilities and constraints.

6.3.3 Development Process

To improve the development process, stricter deadlines and less leniency should be implemented into the team. Everyone in the team should also be encouraged to follow the Gantt chart closely and should not take the buffer time into consideration, as this gave everyone a false sense of security and procrastinated their work.

Rules for developing the software could be implemented into the development process, such as pushing at least one "commit" a day, to assure that everyone is making progress and procrastination is avoided. This involves the use of GitHub which was used throughout the project, so this would not require additional software or knowledge as this was the method used to merge everyone's work.

6.3.4 Security

Currently as described in section 3.1.2.1 the process which the software loads, and stores report files utilizes a process called Serialization. As explained earlier, Serialization is the process of translating data structures and object states to a format which can be stored to a file, or in a memory buffer. This, although being a very efficient method of loading data in and out of software, imposes a substantial security vulnerability to the users if used incorrectly.

Although not implemented into the software due to time constraints, several mitigation techniques have been devised to eradicate the vulnerability.

The first potential mitigation technique would be to change from serialization to a document-oriented (NoSQL) database approach. One of the deciding factors for using Serialization as the preferred method of storing report data was because it allows for developers of user programmable plugins to store any data type, whether it be raw bytes, images, or simple plaintext.

How a document-oriented database differs from traditional relational database is that there is no structure required across tables. As each plugin requires different outputs, and different data types, traditional structures with columns would add additional, and unneeded code for each plugin as they would have to insert and create the structures on initialization. Implementing a document-orientated database such as MongoDB, would allow similar plugin code to which is currently implemented, as the currently implemented serialized dictionary object could be transformed into a JSON object easily with some additional encoding for data types that need to be stored as bytes.

A second potential mitigation technique would take form of certificate signing the serialized objects before writing them to disk. Signing the reports could prevent malicious users from tampering with the serialized objects, or creating their own, this is because once modified, or created outside of the Network Guardian software, it would no longer be signed with the authorised certificate.

The main issue with this specific approach is that if malicious attackers were to reverse engineer the software to gain a copy of the private key, they could then sign own malicious report files, and the vulnerability would still exist. If this was a project of scale or a commercial product aiming to be sold, server sided signing could potentially be implemented but for the purposes of this project that would be considerably overkill when easier, and simpler alternative security measures could be implemented, such as the NoSQL Approach.

6.3.5 Team Management and Dynamic

The team was extremely productive when working in the same space such as the library. This allowed team members to ask for assistance and ensured that everyone was making progress as we could motivate and encourage each other to work through difficult challenges.

The team found out each other's strengths and weaknesses quickly into the development of Network Guardian, this allowed tasks to be assigned to team members that were more comfortable in specific areas. The team was very fortunate to have a team member that had experience and a good understanding of developing software at a professional level, this allowed everyone in the team to have a better understanding themselves, of how we were going to develop Network Guardian as plenty of questions could be asked.

The use of Microsoft Teams made the team more efficient as it allowed them to work on the White Paper and Presentation at the same time, as well as share ideas, documents and files all in one place without having multiple copies of files that are at different stages. The Microsoft Teams Call function also allowed the team to have meetings when the COVID-19 pandemic occurred, making communication easier, ensuring the work would still be completed for the deadline.

To improve the team management, the project manager should talk to the team members individually more often to gaze how well they think the team is performing and if any changes or actions need to happen. This would ensure everyone is happy with the way the team is being run and if there are any problems, they are dealt with quickly and in a professional manner. Having regular individual meetings will also allow team members to get their ideas across if they have a passive and quiet demeanour, allowing the project to be as good as possible and avoiding built up stress team members may have from not having their ideas explored.

7 Conclusion

As highlighted in various sections, Network Guardian provides benefits in the following areas.

- Increase Work rate
- Reduce User Error
- Provide Informative Reports
- Provide Additional educational value
- Reduce Risks

Network Guardian provides multiple benefits to our client and its users. This is done by increasing the work rate of security personnel and reducing the potential of user errors and risks associated with enumerating a network.

By producing thorough and informative reports on the enumeration of a network, Network Guardian benefits users by providing a detailed breakdown on their network.

Network Guardian provides additional educational value by encouraging users to build custom plugins to help enumerate their network. This helps its users improve both their programming and networking skills simultaneously.

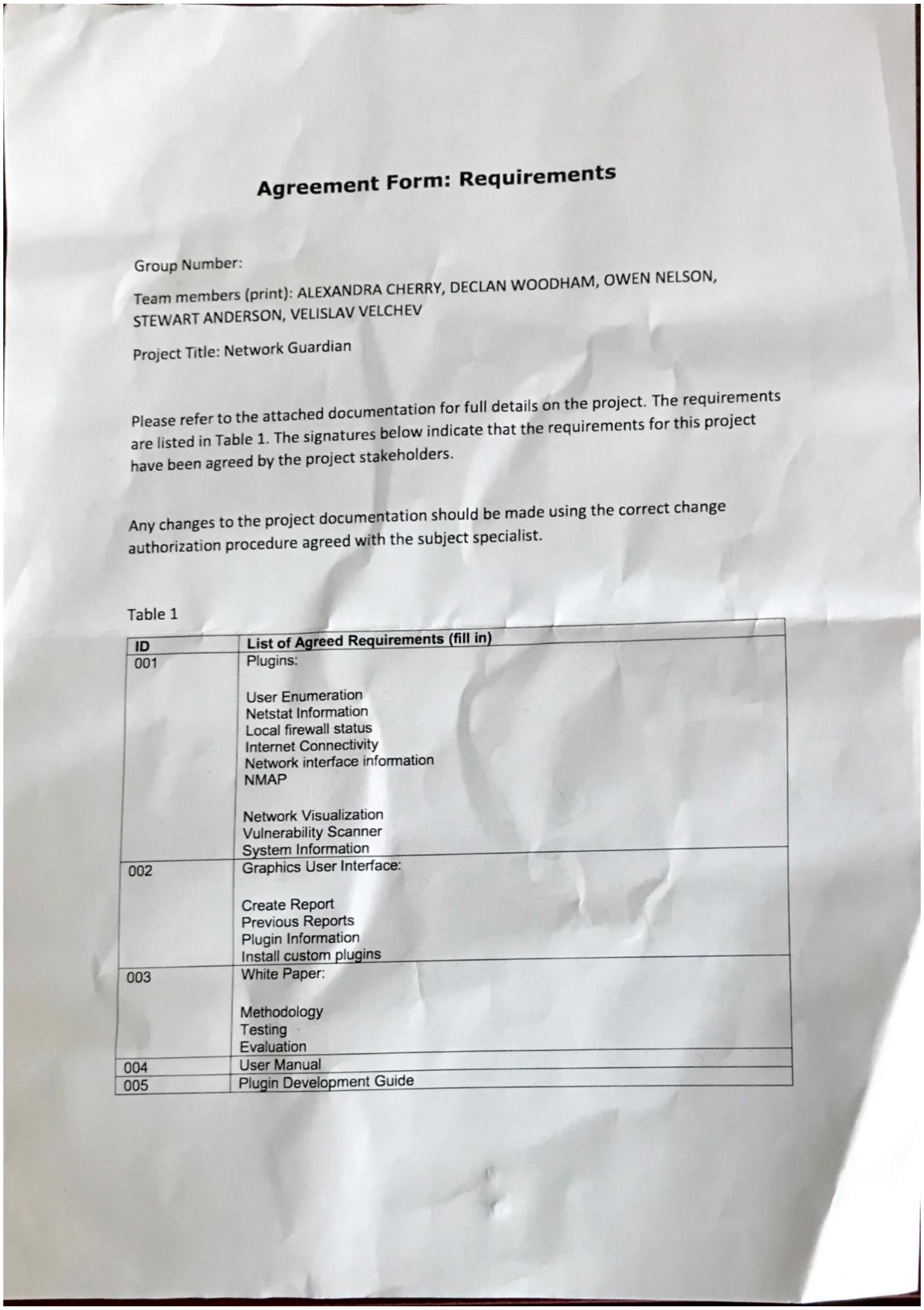
With strong testing procedures the tool has been proven to produce viable, and accurate results, while also providing a high-quality report. As demonstrated, throughout the development, the Network Guardian Team took key steps in ensuring a quality product was delivered, while maintaining and delivering all the required deliverables requested by our client.

8 References

ISO 25000. (n.d.). *ISO 25010*. Retrieved from ISO 25000: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

Microsoft. (2015, 11 15). *Adopt a Git branching strategy*. Retrieved from Microsoft: https://docs.microsoft.com/en-us/azure/devops/repos/git/git-branching-guidance?view=azure-devops

Appendix A: Deliverables and Requirements Specification



Agreement Form: Requirements

Group Number:

Team members (print): ALEXANDRA CHERRY, DECLAN WOODHAM, OWEN NELSON, STEWART ANDERSON, VELISLAV VELCHEV

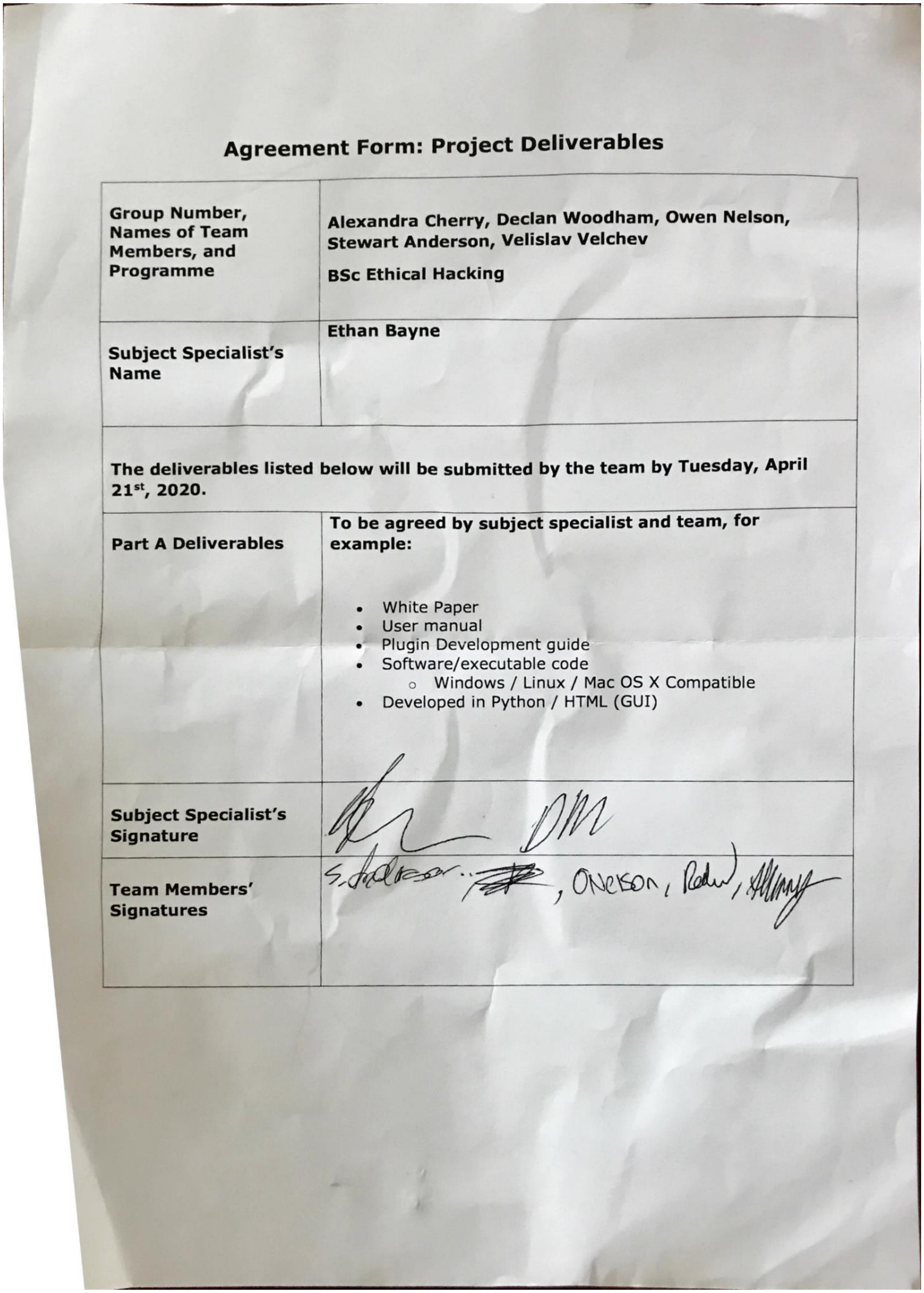
Project Title: Network Guardian

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 authorization procedure agreed with the subject specialist.

Table 1

ID	List of Agreed Requirements (fill in)
001	Plugins: User Enumeration Netstat Information Local firewall status Internet Connectivity Network interface information NMAP Network Visualization Vulnerability Scanner System Information
002	Graphics User Interface: Create Report Previous Reports Plugin Information Install custom plugins
003	White Paper: Methodology Testing Evaluation
004	User Manual
005	Plugin Development Guide



Agreement Form: Project Deliverables

Group Number, Names of Team Members, and Programme	**Alexandra Cherry, Declan Woodham, Owen Nelson, Stewart Anderson, Velislav Velchev** **BSc Ethical Hacking**
Subject Specialist's Name	**Ethan Bayne**
The deliverables listed below will be submitted by the team by Tuesday, April 21st, 2020.	
Part A Deliverables	**To be agreed by subject specialist and team, for example:** • White Paper • User manual • Plugin Development guide • Software/executable code ○ Windows / Linux / Mac OS X Compatible • Developed in Python / HTML (GUI)
Subject Specialist's Signature	[illegible] [illegible]
Team Members' Signatures	[illegible], [illegible], ONelson, [illegible], [illegible]

Appendix B: User Guide

TEAM: Pentagon
Alexandra Cherry,
Declan Woodham,
Owen Nelson,
Stewart Anderson,
Velislav Velchev

USER GUIDE

Version Number: 1.0
Version Date: 18/02/2020

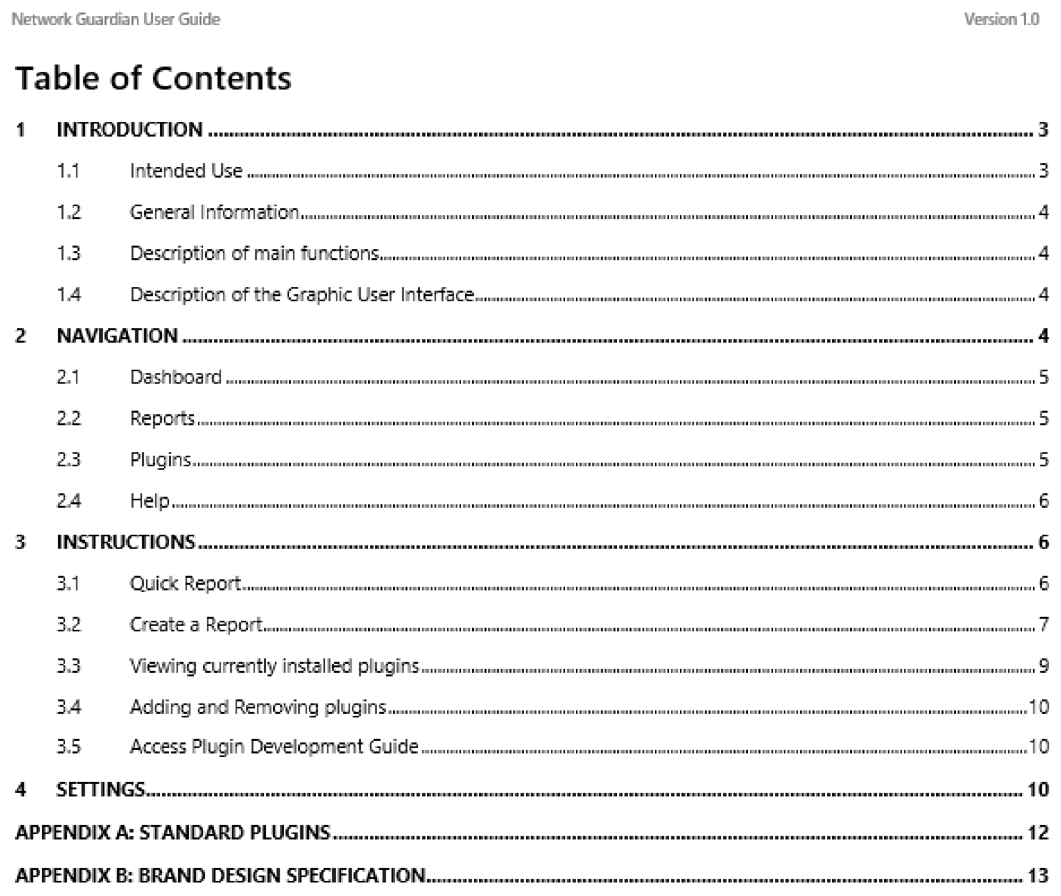
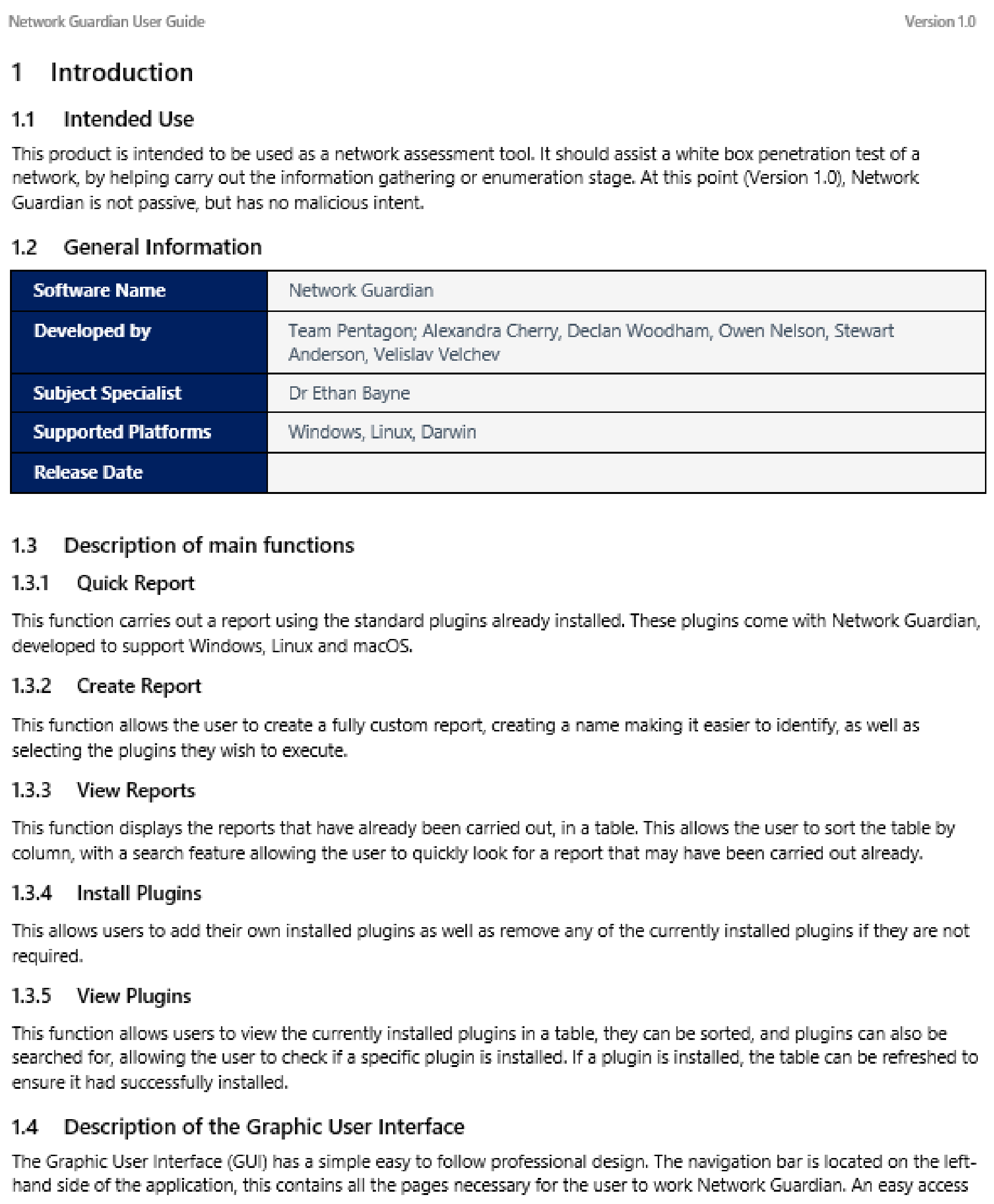


Table of Contents



1 Introduction

1.1 Intended Use

This product is intended to be used as a network assessment tool. It should assist a white box penetration test of a network, by helping carry out the information gathering or enumeration stage. At this point (Version 1.0), Network Guardian is not passive, but has no malicious intent.

1.2 General Information

Software Name	Network Guardian
Developed by	Team Pentagon; Alexandra Cherry, Declan Woodham, Owen Nelson, Stewart Anderson, Velislav Velchev
Subject Specialist	Dr Ethan Bayne
Supported Platforms	Windows, Linux, Darwin
Release Date	

1.3 Description of main functions

1.3.1 Quick Report

This function carries out a report using the standard plugins already installed. These plugins come with Network Guardian, developed to support Windows, Linux and macOS.

1.3.2 Create Report

This function allows the user to create a fully custom report, creating a name making it easier to identify, as well as selecting the plugins they wish to execute.

1.3.3 View Reports

This function displays the reports that have already been carried out, in a table. This allows the user to sort the table by column, with a search feature allowing the user to quickly look for a report that may have been carried out already.

1.3.4 Install Plugins

This allows users to add their own installed plugins as well as remove any of the currently installed plugins if they are not required.

1.3.5 View Plugins

This function allows users to view the currently installed plugins in a table, they can be sorted, and plugins can also be searched for, allowing the user to check if a specific plugin is installed. If a plugin is installed, the table can be refreshed to ensure it had successfully installed.

1.4 Description of the Graphic User Interface

The Graphic User Interface (GUI) has a simple easy to follow professional design. The navigation bar is located on the left-hand side of the application, this contains all the pages necessary for the user to work Network Guardian. An easy access

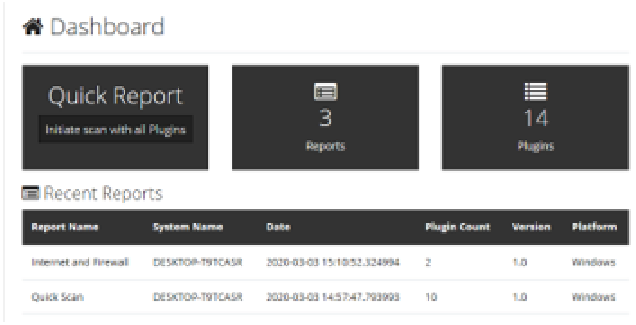
'Create Report' button is located at the top right of every page, so a user can always start a new report, there is also a settings icon where the user can change settings, as described in section *5. Settings.*

2 Navigation

To navigate through Network Guardian, there is a navigation bar on the left-hand side of the application, as described in section *2.4 Description of the Graphic User Interface.*

2.1 Dashboard

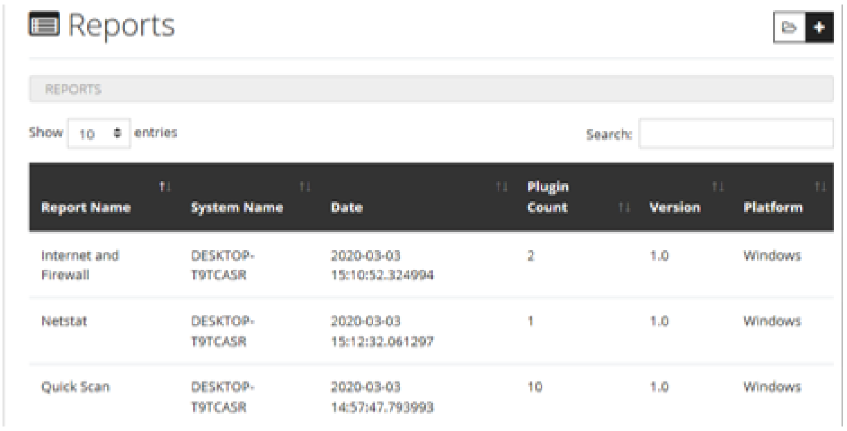
Once the program is run, the software opens and displays the 'Dashboard'. This shows the 'Quick Report' function, the number of reports completed, and the number of plugins currently installed. There is also a table displaying the most recent reports that are completed. To go back to the dashboard, use the navigation bar on the left-hand side, and click 'Dashboard'.



Report Name	System Name	Date	Plugin Count	Version	Platform
Internet and Firewall	DESKTOP-T9TCASR	2020-03-03 15:10:52.324994	2	1.0	Windows
Quick Scan	DESKTOP-T9TCASR	2020-03-03 14:57:47.793993	10	1.0	Windows

2.2 Reports

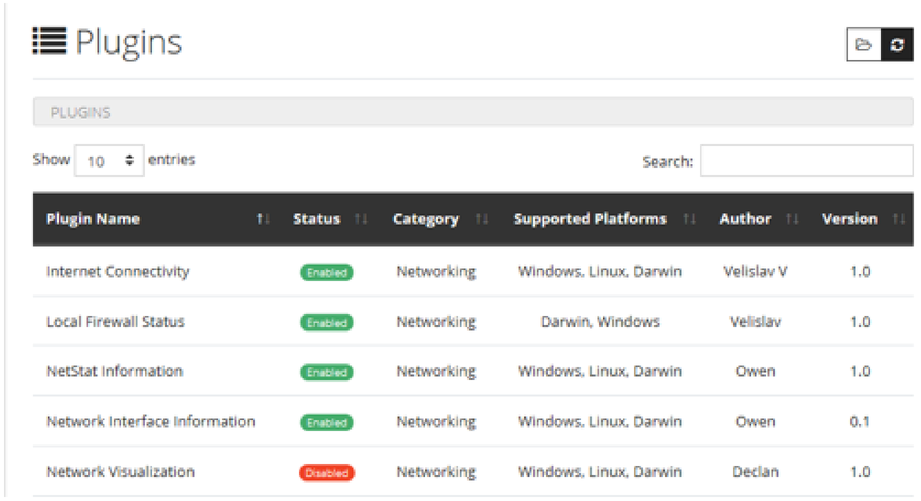
To the reports page, click the 'Reports' button located in the navigation bar. This will display a table with all the reports that have been carried out, they are found in the 'reports' directory. These can be sorted within the table, there is also a search function that can be used to look for a specific kind of report, from 'Report Name' to the 'Platform' the report was carried out on.



Report Name	System Name	Date	Plugin Count	Version	Platform
Internet and Firewall	DESKTOP-T9TCASR	2020-03-03 15:10:52.324994	2	1.0	Windows
Netstat	DESKTOP-T9TCASR	2020-03-03 15:12:32.061297	1	1.0	Windows
Quick Scan	DESKTOP-T9TCASR	2020-03-03 14:57:47.793993	10	1.0	Windows

2.3 Plugins

To access the plugins page, click the 'Plugins' button located in the navigation bar. This will display a page with a table showing all the currently installed plugins, found in the plugins directory of Network Guardian. These plugins can be sorted within the table, there is also a search function, where specific plugins can be located, whether it searches the 'Plugin Name', 'Author' or 'Supported Platform'.



Plugin Name	Status	Category	Supported Platforms	Author	Version
Internet Connectivity	Enabled	Networking	Windows, Linux, Darwin	Velislav V	1.0
Local Firewall Status	Enabled	Networking	Darwin, Windows	Velislav	1.0
NetStat Information	Enabled	Networking	Windows, Linux, Darwin	Owen	1.0
Network Interface Information	Enabled	Networking	Windows, Linux, Darwin	Owen	0.1
Network Visualization	Disabled	Networking	Windows, Linux, Darwin	Declan	1.0

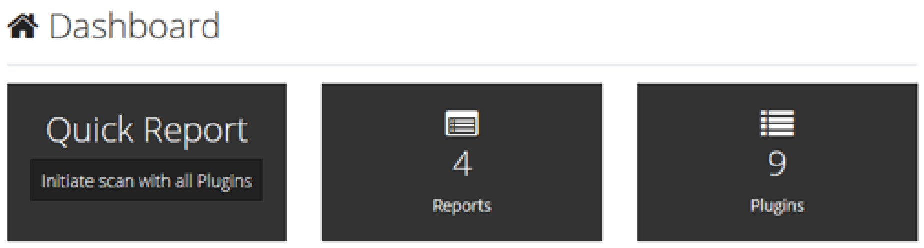
2.4 Help

To access the help page, click the 'Help' button located in the navigation bar. This will display an explanation of Network Guardian and the development team. The 'User Guide' and 'Plugin Development Guide' can also be accessed from this page.

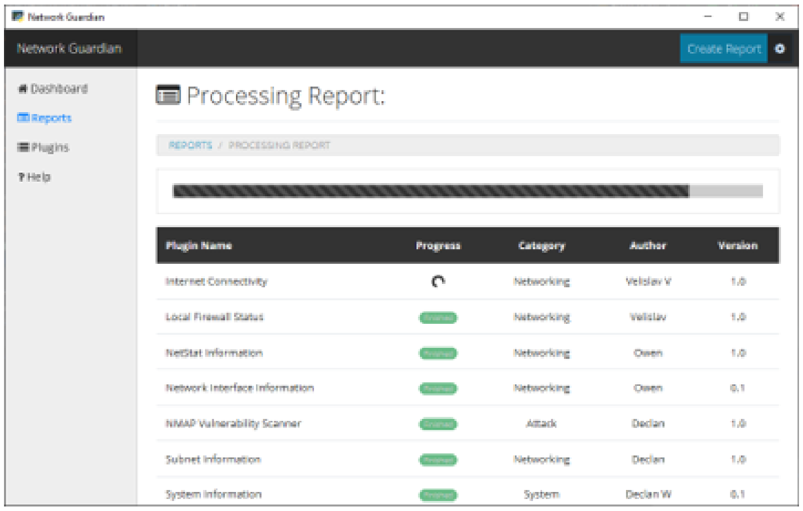
3 Instructions

3.1 Quick Report

There is button located on the 'Dashboard' page, 'Initiate scan with all plugins'. Once this is clicked, a report is carried out using all the currently installed plugins.



The scanning page is then displayed showing a list of the plugins being ran, as well as a progress bar.

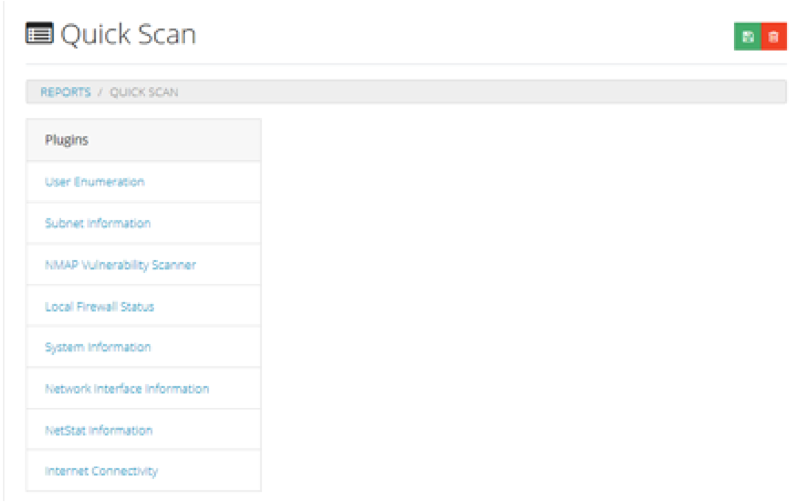


Processing Report:

REPORTS / PROCESSING REPORT

Plugin Name	Progress	Category	Author	Version
Internet Connectivity		Networking	Velislav V	1.0
Local Firewall Status	Finished	Networking	Velislav	1.0
NetStat Information	Finished	Networking	Owen	1.0
Network Interface Information	Finished	Networking	Owen	0.1
NMAP Vulnerability Scanner	Finished	Attack	Declan	1.0
Subnet Information	Finished	Networking	Declan	1.0
System Information	Finished	System	Declan W	0.1

Once the report is completed a results page is then displayed, which shows all the plugins executed. There is also an option to delete the report or export it as an HTML file.



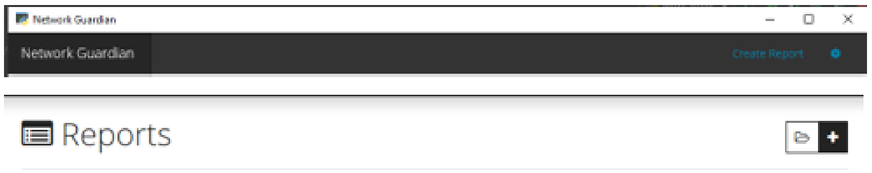
Quick Scan

REPORTS / QUICK SCAN

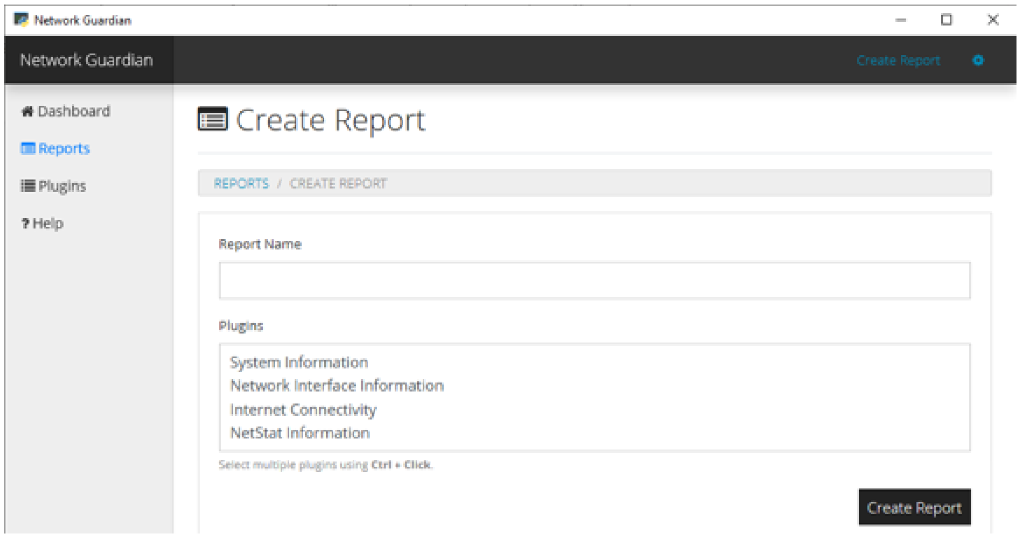
Plugins
User Enumeration
Subnet Information
NMAP Vulnerability Scanner
Local Firewall Status
System Information
Network Interface Information
NetStat Information
Internet Connectivity

3.2 Create a Report

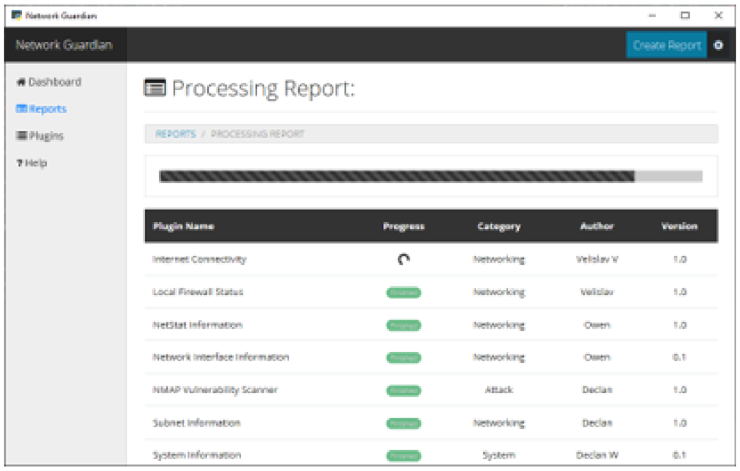
At the top right of every page there is a 'Create Report' button, there is also a button (Plus icon) located on the 'Reports' page.



Once this is clicked the user is then taken to a page displaying a form. This form has the following inputs; Name of Report and Plugins. The plugins are a multi-select where the user must select the plugins they wish to execute using *'ctrl + click'*.



Once the desired plugins are selected the user must then click 'Create Report' at the bottom of the form. The user will then be taken to a page showing the progress of the report as well as a table of the selected plugins.



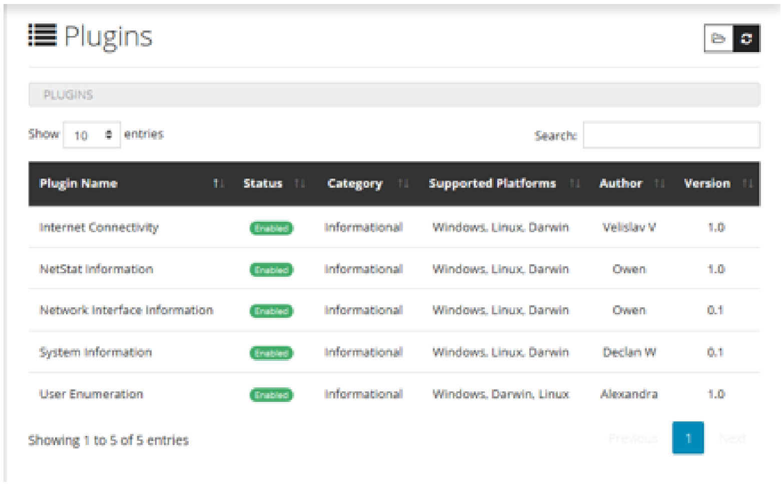
Processing Report:

Plugin Name	Progress	Category	Author	Version
Internet Connectivity		Networking	Velislav V	1.0
Local Firewall Status	[illegible]	Networking	Velislav	1.0
NetStat Information	[illegible]	Networking	Owen	1.0
Network Interface Information	[illegible]	Networking	Owen	0.1
NMAP Vulnerability Scanner	[illegible]	Attack	Declan	1.0
Subnet Information	[illegible]	Networking	Declan	1.0
System Information	[illegible]	System	Declan W	0.1

Once the report is completed the results are displayed to the user, these can be viewed again by going to the reports page and selecting the appropriate report.

3.3 Viewing currently installed plugins

To view the currently installed plugins, click 'Plugins' located in the navigation bar on the far left. This will take the user to the 'Plugins' page.



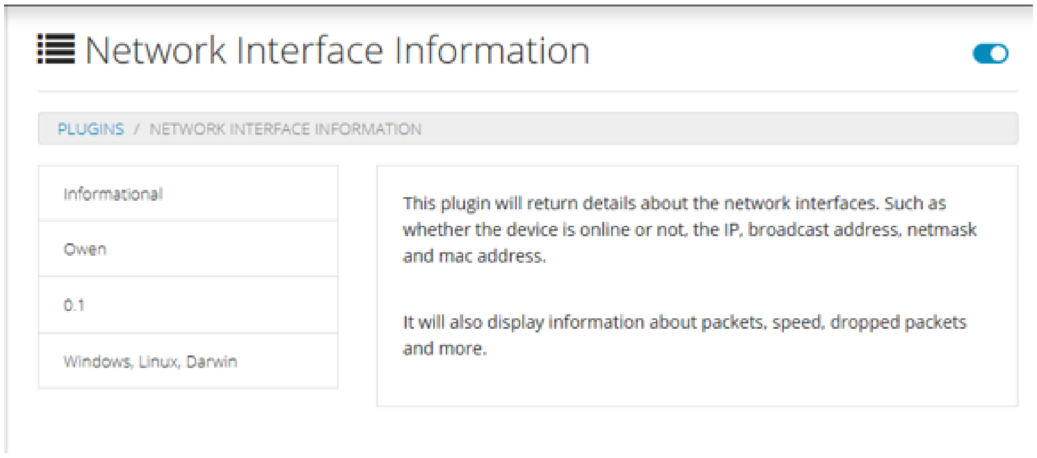
Plugins

Plugin Name	Status	Category	Supported Platforms	Author	Version
Internet Connectivity	Enabled	Informational	Windows, Linux, Darwin	Velislav V	1.0
NetStat Information	Enabled	Informational	Windows, Linux, Darwin	Owen	1.0
Network Interface Information	Enabled	Informational	Windows, Linux, Darwin	Owen	0.1
System Information	Enabled	Informational	Windows, Linux, Darwin	Declan W	0.1
User Enumeration	Enabled	Informational	Windows, Darwin, Linux	Alexandra	1.0

Showing 1 to 5 of 5 entries

The page displays a table listing all the currently installed plugins and if they are *enabled* and *supported by the current operating system*. The *category* the plugin is in, is also displayed as well as the *author* and *version* of the plugin.

The user can select a plugin and they are taken to a page where more information, for example the description, is displayed to the user.


Network Interface Information
PLUGINS / NETWORK INTERFACE INFORMATION

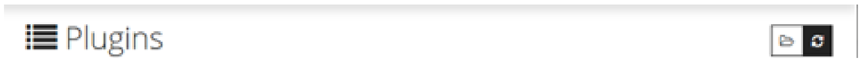
Informational
Owen
0.1
Windows, Linux, Darwin

This plugin will return details about the network interfaces. Such as whether the device is online or not, the IP, broadcast address, netmask and mac address.

It will also display information about packets, speed, dropped packets and more.

3.4 Adding and Removing plugins

To add or remove a plugin, go to the 'plugins' directory for Network Guardian, this can be located by clicking the folder icon on the 'Plugins' page.



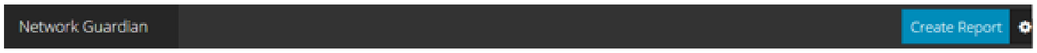
From there the user can add and remove plugins as they wish. To update the table, click the 'refresh' icon next to the folder icon and the table will update, displaying the modified plugins and if they are enabled.

3.5 Access Plugin Development Guide

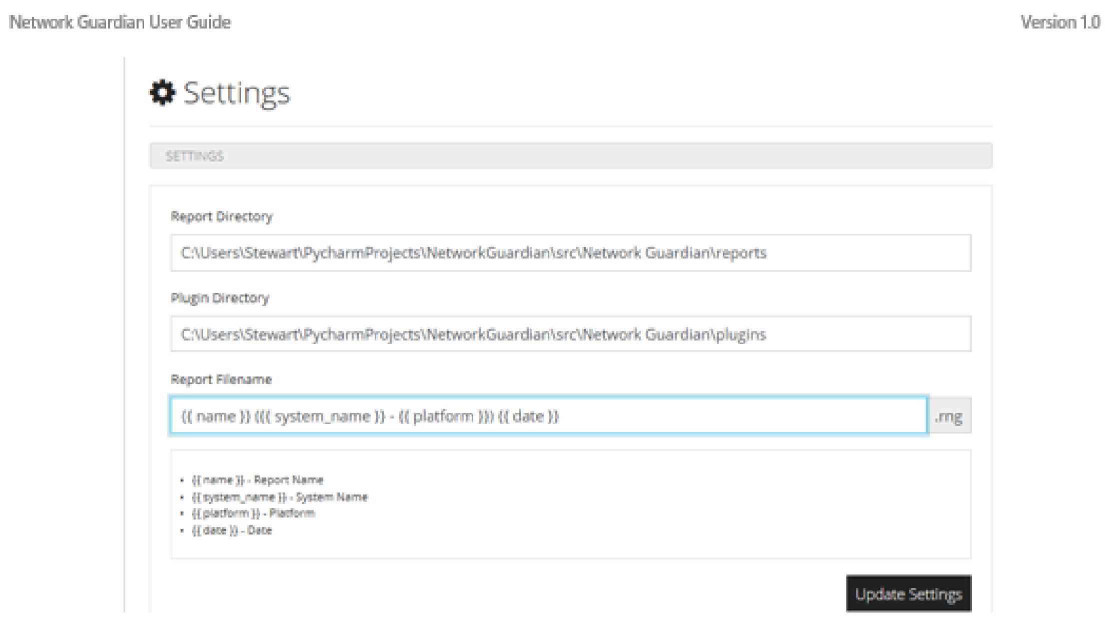
To access the 'Plugin Development Guide' go to the 'Help' page using the navigation bar. From here select the link with the label 'Plugin Development Guide'.

4 Settings

To access the 'Settings' page, click the cog icon at the top right of the page, next to 'Create Report'.



The settings page allows the user to change the directory for reports as well as plugins. The user can also alter the default name for reports. This is only recommended if you are an advanced user.



Network Guardian User Guide Version 1.0

Settings

SETTINGS

Report Directory

C:\Users\Stewart\PycharmProjects\NetworkGuardian\src\Network Guardian\reports

Plugin Directory

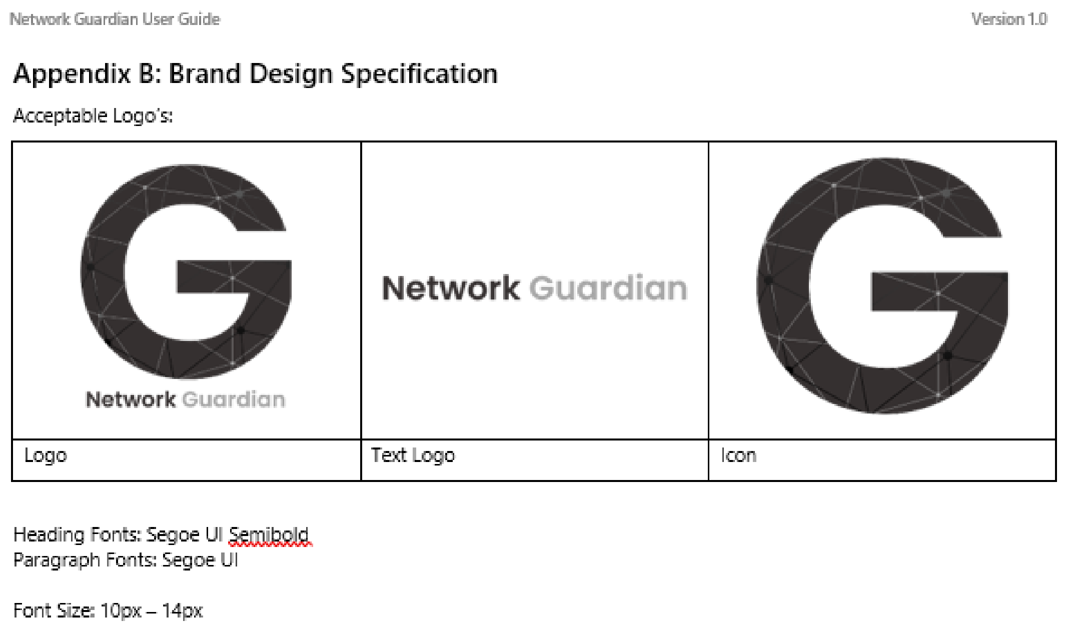
C:\Users\Stewart\PycharmProjects\NetworkGuardian\src\Network Guardian\plugins

Report Filename

{{ name }} ({{ system_name }} - {{ platform }}) {{ date }} .rng

- {{ name }} - Report Name
- {{ system_name }} - System Name
- {{ platform }} - Platform
- {{ date }} - Date

Update Settings



Network Guardian User Guide Version 1.0

Appendix B: Brand Design Specification

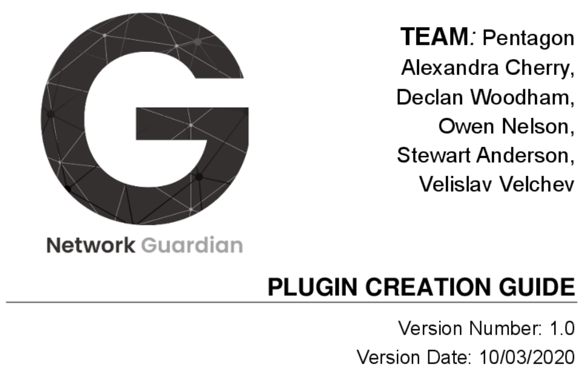
Acceptable Logo's:

Network Guardian	Network Guardian	
Logo	Text Logo	Icon

Heading Fonts: Segoe UI Semibold
Paragraph Fonts: Segoe UI

Font Size: 10px – 14px

Appendix C: Plugin Development Guide

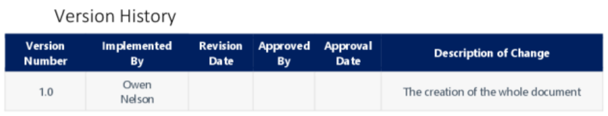


TEAM: Pentagon
Alexandra Cherry,
Declan Woodham,
Owen Nelson,
Stewart Anderson,
Velislav Velchev

Network Guardian

PLUGIN CREATION GUIDE

Version Number: 1.0
Version Date: 10/03/2020

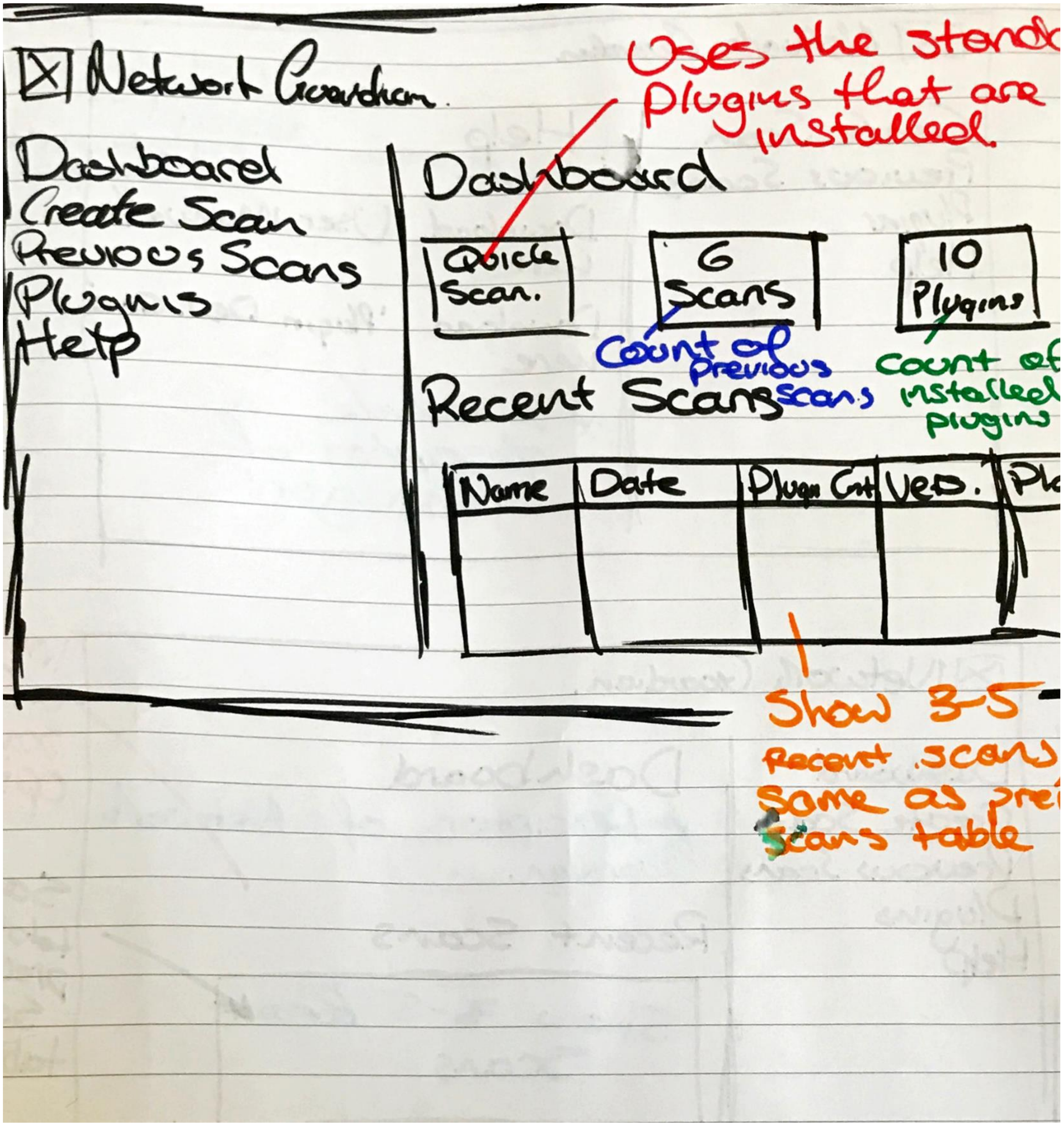


Version History

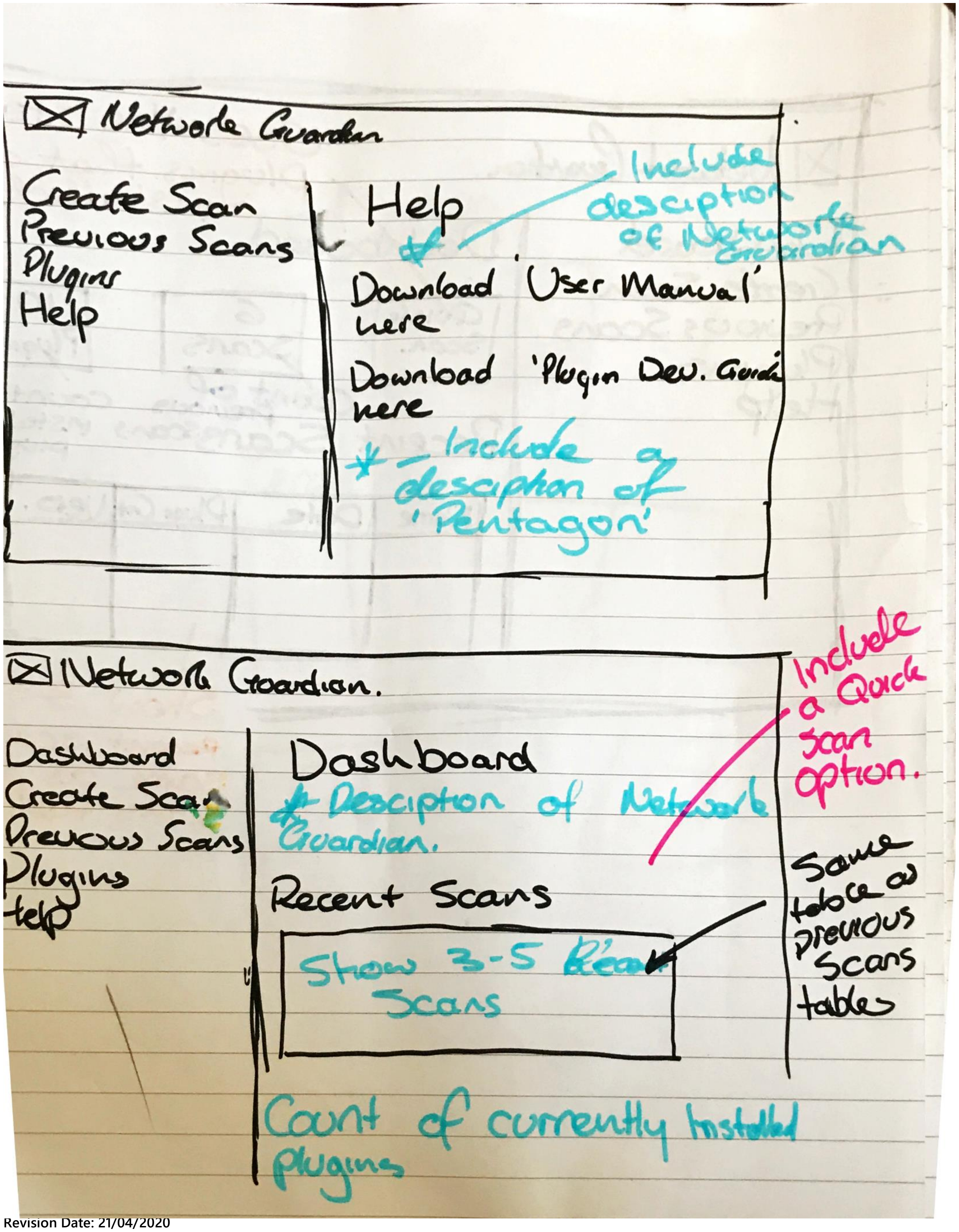
Version Number	Implemented By	Revision Date	Approved By	Approval Date	Description of Change
1.0	Owen Nelson				The creation of the whole document

Appendix D: Graphics User Interface Wireframe Designs

Wireframe 1



Wireframe 2

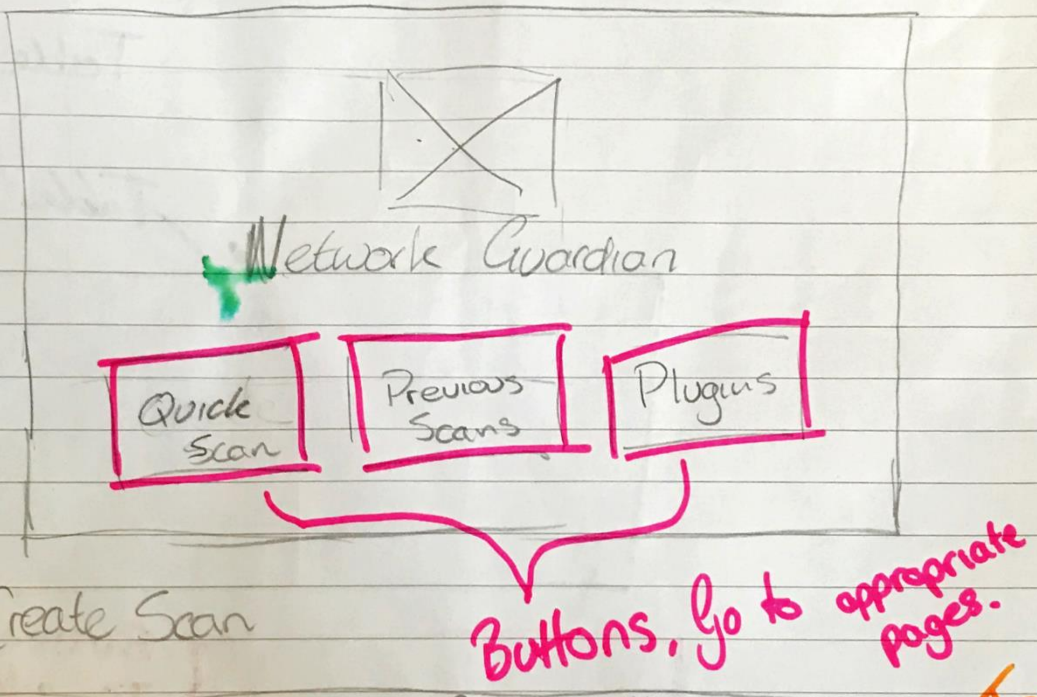

Network Guardian
Create Scan
Previous Scans
Plugins
Help
Help
Include description of Network Guardian
Download 'User Manual' here
Download 'Plugin Dev. Guide' here
Include a description of 'Pentagon'
Network Guardian.
Dashboard
Create Scan
Previous Scans
Plugins
Help
Dashboard
Description of Network Guardian.
Include a Quick Scan option.
Recent Scans
Show 3-5 Recent Scans
Same table as previous Scans tables
Count of currently Installed plugins

Wireframe 3

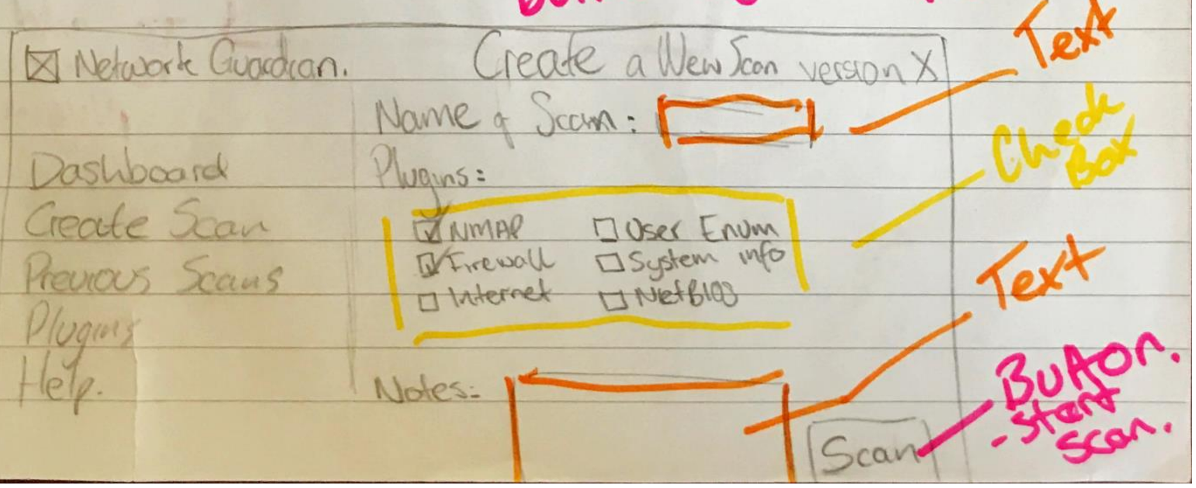
Network Guardian

- Quick Scan (Dashboard)
- Previous Scans · Name of · Scan, Date, version.
- Plugins
  - ~~Select Plugins~~ Description of Plugin.
  - Add Plugins - Supported Platfor
- Create Scan

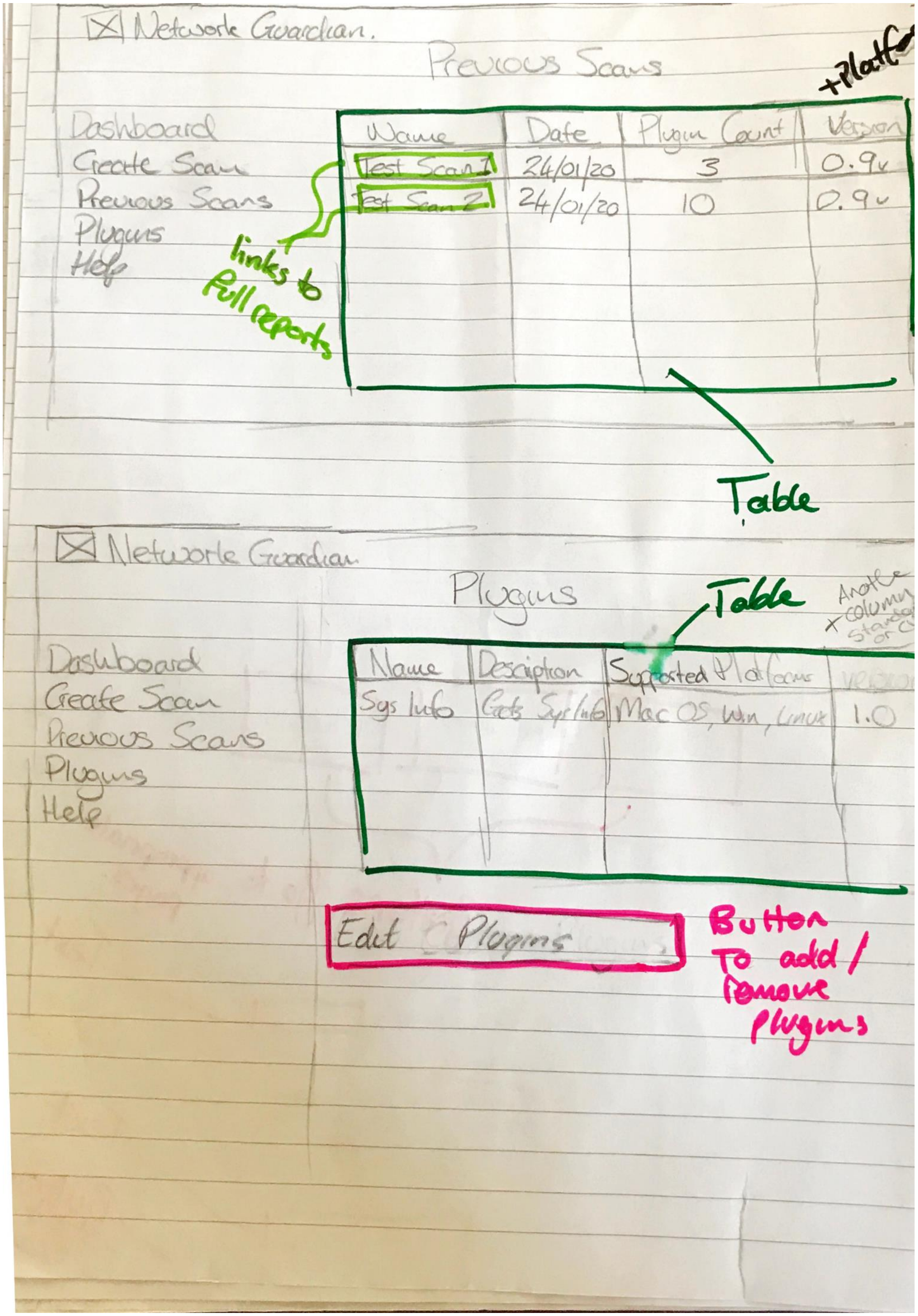
Initial Launch



Create Scan



Wireframe 4


Network Guardian.
Previous Scans
+Platf
Dashboard
Create Scan
Previous Scans
Plugins
Help
Name
Date
Plugin Count
Version
Test Scan 1
24/01/20
3
0.9v
Test Scan 2
24/01/20
10
0.9v
links to full reports
Table
Network Guardian
Plugins
Table
Another column
Dashboard
Create Scan
Previous Scans
Plugins
Help
Name
Description
Supported Platforms
Version
Sys Info
Gets Sys Info
Mac OS, Win, Linux
1.0
Edit Plugins
Button to add / remove plugins

Appendix E: Brand Design Specification

The following brand design specification was created to ensure uniformity and between all documentation and software produced under the Network Guardian brand.

Acceptable Logo Usage:

Network Guardian

Network Guardian

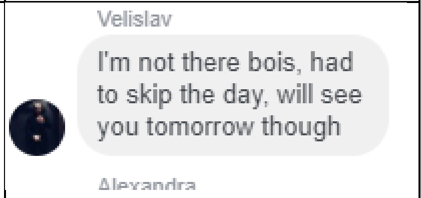
Logo Text Logo Icon

Heading Fonts: Segoe UI Semi Bold
Paragraph Fonts: Segoe UI
Font Size: 10px – 14px

Appendix F: Minutes

Lecture 13/01/2020 - Week 1 – Author: Declan

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes	
Declan Woodham	Yes	
Owen Nelson	No	Unknown, did not respond in group chat
Velislav Velchev	No	 Velislav I'm not there bois, had to skip the day, will see you tomorrow though
Stewart Anderson	Yes	

Minutes

Just learning about requirements and submissions involved in the module

Class Meeting 17/01/2020 - Week 1 – Author: Declan

Attendance

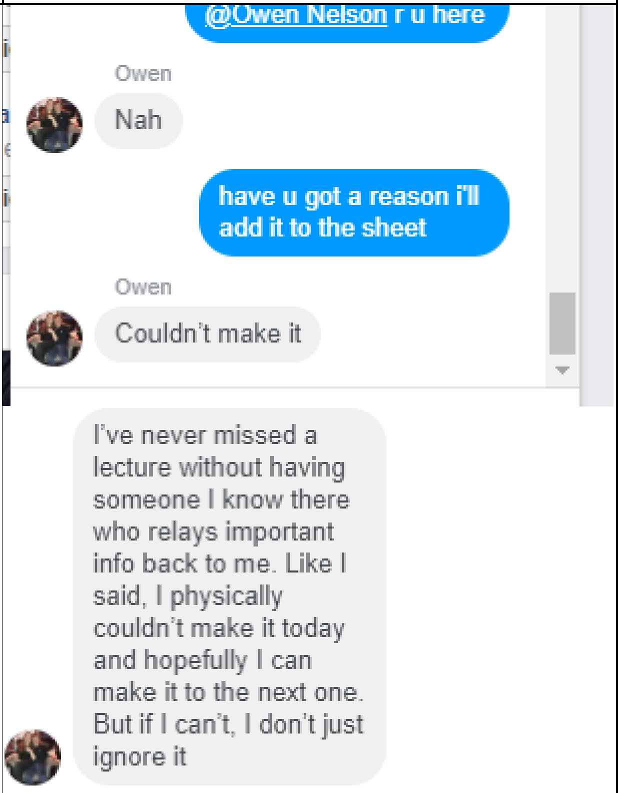
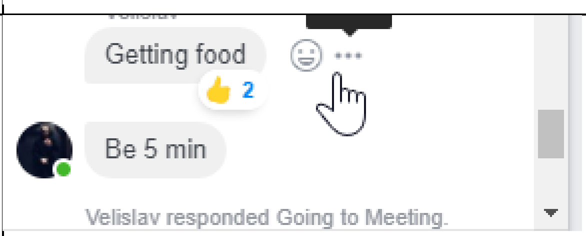
Name	Attendance	Reason if given
Alexandra Cherry	Yes	
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	Late	
Stewart Anderson	Yes	

Minutes

- Delegated jobs
  - Alexandra, Owen, Velislav – Plugin Development
  - Stewart – GUI Design
  - Declan – Framework
- Group asked to install PyCharm and register for GitHub accounts before the next meeting
- Booked room in library for next meeting
  - Wednesday 22nd January 14:00 – 16:00 Room 2.03 in the library
  - Friday 24th January 13:00 – 16:00 Room 2.03 in the library

Lecture 20/01/2020 - Week 2 – Author: Declan

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes	
Declan Woodham	Yes	
Owen Nelson	No	 @Owen Nelson r u here / Owen: Nah / have u got a reason i'll add it to the sheet / Owen: Couldn't make it / I've never missed a lecture without having someone I know there who relays important info back to me. Like I said, I physically couldn't make it today and hopefully I can make it to the next one. But if I can't, I don't just ignore it
Velislav Velchev	Arrived 35 minutes late.	 Getting food / Be 5 min / Velislav responded Going to Meeting.
Stewart Anderson	Yes	

Minutes

- Need to do the deliverables sheet ASAP
- Individual Contribution
  - 4 pages of A4 normal text e.g 10-11 point font size
- 4 Abertay Attributes
  - Intellectual
  - Personal
  - Professional
  - Active Citizen
- Create a personal logbook of reflections

Team Meeting 22/01/2020 - Week 2 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes (Late)	Meeting
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	Yes	
Stewart Anderson	Yes	

Minutes

- Set up PyCharm on everyone's workstations
- Assigned initial Plugins to team members
    - Network Interface Information – Owen
    - Internet Connectivity check – Velislav

Team Meeting 24/01/2020 - Week 2 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes	
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	Yes	
Stewart Anderson	Yes	

Minutes

- Velislav completed 'Internet Connectivity' plugin
- Stewart created potential GUI designs on paper
- Owen almost completed 'Network Interface' plugin, needs to finish the template.

Team Meeting 28/01/2020 - Week 3 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes	
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	Yes	
Stewart Anderson	Yes	

Minutes

- Stewart must set up remote access to home workstation, as working from USB doesn't work due to the required admin privileges needed for development.
- Stewart basically completed initial GUI designs
- Owen completed the 'Network Interface' plugin.
- Set up pycharm on Alexandra's laptop
- Alexandra started user enumeration

Team Meeting 31/01/2020 - Week 3 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes	
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	Yes	
Stewart Anderson	Yes (20 minutes late)	Work

Minutes

- Stewart started HTML development on the GUI
- Alexandra made progress on user enumeration plugin
- Velislav continuing with firewall plugin
- Owen started Netstat plugin
- Declan working on report handling system

Team Meeting 04/02/2020 - Week 4 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes	
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	Yes (40 minutes late)	
Stewart Anderson	Yes	

Minutes

- continuing with GUI, added create scan page, quick scan button and scanning page - Stewart
- Progress on User enumeration – Alexandra
- Finished Netstat and reworked network interface – Owen
- Assisted Alexandra and Velislav – Declan
- Finished testing Mac Firewall plugin, started testing windows – Velislav

Team Meeting 11/02/2020 - Week 5 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes	
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	No	
Stewart Anderson	Yes	

Minutes

- Started NMAP plugin – Owen
- Implemented GUI navigation and some dynamic elements such as plugins – Stewart
- Worked on Jinja template for user enumeration plugin – Alexandra
- Implemented export HTML option and worked on framework– Declan
- Absent – Velislav

Team Meeting 14/02/2020 - Week 5 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes (50 minutes late)	
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	Yes (35 minutes late)	
Stewart Anderson	Yes	

Minutes

- GUI improvements, jinja2 url_for function, added breadcrumbs, made page titles more dynamic. - Stewart
- Progress on NMAP scan – Owen
- Finished Firewall status for windows – Velislav
- A lot more work – Declan
- Completed user enumeration for windows and linux – Alexandra

Team Meeting 18/02/2020 - Week 6 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes	
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	Yes	
Stewart Anderson	Yes	

Minutes

- Started on the 'User Guide' - Stewart
- Completed Firewall, Linux unsupported – Velislav
- Completed user enumeration, started creating a virtual network for testing – Alexandra
- Continuing with NMAP – Owen
- A lot of work on integrating GUI and Framework – Declan

Team Meeting 20/02/2020 - Week 6 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes (late 20 minutes)	
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	Yes	
Stewart Anderson	Yes	

Minutes

- Continuing with User Guide – Stewart
- Creating full plugin descriptions – Velislav
- Continuing with NMAP – Owen
- Finished Pywebview Integration - Declan
- Creating test features; network emulation – Alexandra

Team Meeting 21/02/2020 - Week 6 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes (20 minutes late)	Meeting overran
Declan Woodham	Yes	
Owen Nelson	Yes (90 minutes late)	vets
Velislav Velchev	Yes (75 minutes late)	work
Stewart Anderson	Yes (90 minutes late)	work

Minutes

Warning for members being constantly late and lack of work - Stewart	

- User guide – Stewart
- Descriptions of plugins – velislav
- NMAP – owen
- Improved plugin loading – Declan
- Setting up network emulation – Alexandra

Team Meeting 24/02/2020 - Week 7 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes	
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	Yes	
Stewart Anderson	Yes (arrived 30 mins later)	

Minutes

- Completed User Guide for Functions already completed, waiting for software completion to update screenshots and settings section – Stewart
- Integrated firewall plugin, worked on network visualization and added exception handling - Declan
- Progress on everything – Owen, Alexandra

Team Meeting 25/02/2020 - Week 7 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes (70 minutes later)	Went to lab
Declan Woodham	Yes (60 minutes later)	Went to lab
Owen Nelson	Yes	
Velislav Velchev	Yes (80 minutes later)	Went to lab
Stewart Anderson	Yes	

Minutes

- Created the structure of the White paper – Stewart
- Worked on network visualization, helped owen – declan
- Worked on NMAP, information is now in the dictionary – Owen
- Working on Virtual Network for testing – Alexandra

Team Meeting 03/03/2020 - Week 8 – Author: Stewart

Attendance

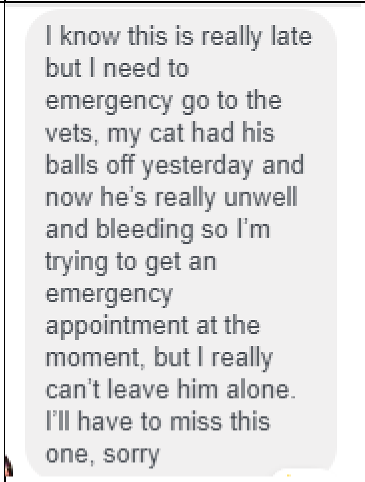
Name	Attendance	Reason if given
Alexandra Cherry	Yes	
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	Yes (45 minutes late)	
Stewart Anderson	Yes	

Minutes

- Completed licensing file in Network Guardian – Velislav
- Updated User Guide screenshots, start white paper, creating the graphic user interface section – Stewart
- Working on template for NMAP – Owen
- Completed initial Virtual Network, now creating another one as well as documenting the steps – Alexandra
- Fixed Windows bug for saving reports, sorted out java script files, worked on licensing and worked on command line interface, improved window view code base – Owen

Team Meeting 06/03/2020 - Week 8 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes	
Declan Woodham	Yes	
Owen Nelson	No	 I know this is really late but I need to emergency go to the vets, my cat had his balls off yesterday and now he's really unwell and bleeding so I'm trying to get an emergency appointment at the moment, but I really can't leave him alone. I'll have to miss this one, sorry
Velislav Velchev	Yes	
Stewart Anderson	Yes	

Minutes

- GUI in the whitepaper – Stewart
- Working on Virtual Network for testing – Alexandra
- Working on the Network Visualization Map plugin - Declan
- Emailed Andrea (module tutor) to get clearer idea of the white paper, trying to organise a meeting to discuss the project

Team Meeting 10/03/2020 - Week 10 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes (50 minutes late)	
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	Yes	
Stewart Anderson	Yes	

Minutes

- Testing discussion

Team Meeting 13/03/2020 - Week 10 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes (30 minutes late)	
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	Yes (30 minutes late)	
Stewart Anderson	Yes	

Minutes

- Discussed with client (Ethan) what he would like in the whitepaper
- Client requested that vulnerabilities were highlighted in a way to stand out
- Structured the white paper
- Alexandra and Velislav working on the test networks
- Declan working on implementing a function that highlights vulnerabilities
- Owen working on plugin development guide
- Stewart working on the whitepaper

Team Meeting 16/03/2020 - Week 11 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes	
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	Yes (45 minutes late)	
Stewart Anderson	Yes	

Minutes

- Owen completed plugin guide
- Velislav downloaded VMs
- Declan and Stewart worked on Whitepaper
- Alexandra working on Virtual Network for testing
- Next meeting 19/03/2020, room 4.05 @ 2pm-5pm

VIRTUAL Team Meeting 04/04/2020 - Week 13 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	Yes	
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	Yes	
Stewart Anderson	Yes	

Minutes

- Assigned sections of the whitepaper
  - Stewart – Execution of the plan, Issues and Future Work
  - Velislav – Background
  - Declan – Creation of Network Guardian
  - Alexandra – Testing strategy and Creation of the test environment
  - Owen - Development of NMAP plugin

VIRTUAL Team Meeting 13/04/2020 - Week 15 – Author: Stewart

Attendance

Name	Attendance	Reason if given
Alexandra Cherry	No	Just said unavailable
Declan Woodham	Yes	
Owen Nelson	No	No reason
Velislav Velchev	Yes	
Stewart Anderson	Yes	

Minutes

- Stewart, Declan and Velislav working on the presentation

VIRTUAL Team Meeting 21/04/2020 - Week 16 – Author: Velislav

Attendance

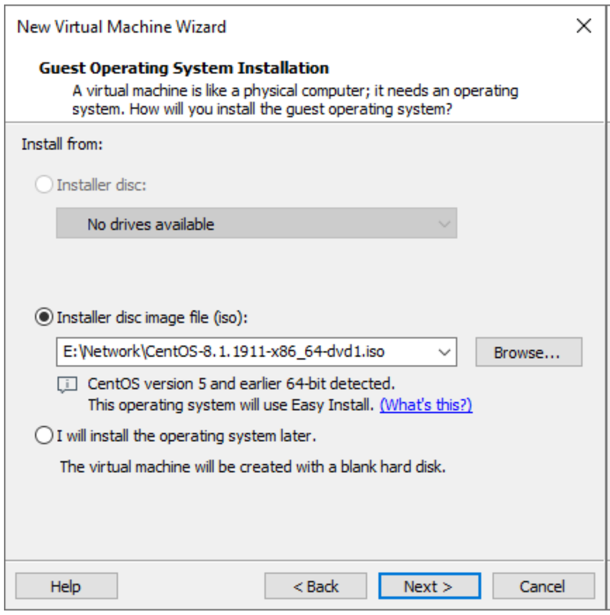
Name	Attendance	Reason if given
Alexandra Cherry	Yes	
Declan Woodham	Yes	
Owen Nelson	Yes	
Velislav Velchev	Yes (15min)	Gone to shop
Stewart Anderson	Yes	

Minutes

- Velislav – Completed Background on 20/04/2020, working on building .img file for Mac Os X.
- Alexandra -
- Stewart -
- Declan - Helped Velislav with IMG file.
- Owen -

Appendix G: Third Party Library Licenses

Appendix H: Virtual Machine Set-up Using New Virtual Machine Wizard in VMWare Workstation



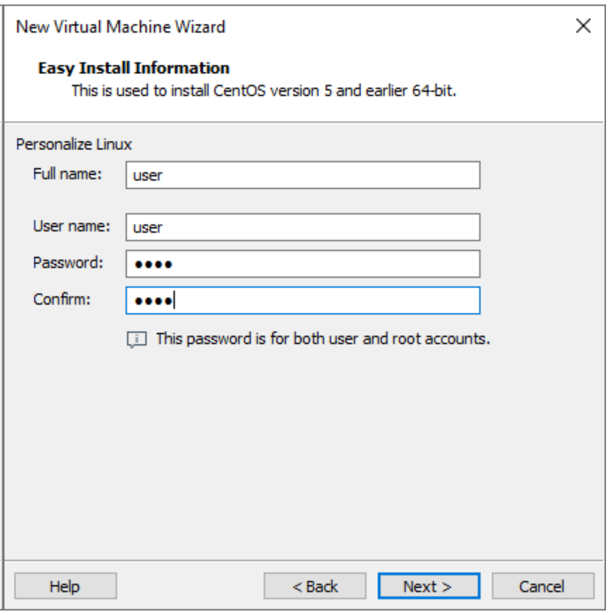
Guest Operating System Installation
A virtual machine is like a physical computer; it needs an operating system. How will you install the guest operating system?

Install from:

○ Installer disc:
No drives available

● Installer disc image file (iso):
E:\Network\CentOS-8.1.1911-x86_64-dvd1.iso
CentOS version 5 and earlier 64-bit detected.
This operating system will use Easy Install. (What's this?)

○ I will install the operating system later.
The virtual machine will be created with a blank hard disk.

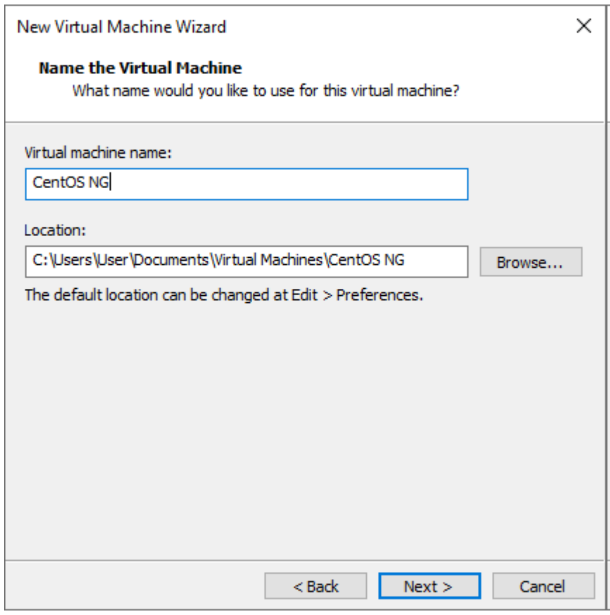


Easy Install Information
This is used to install CentOS version 5 and earlier 64-bit.

Personalize Linux

Full name: user
User name: user
Password: ••••
Confirm: ••••

This password is for both user and root accounts.

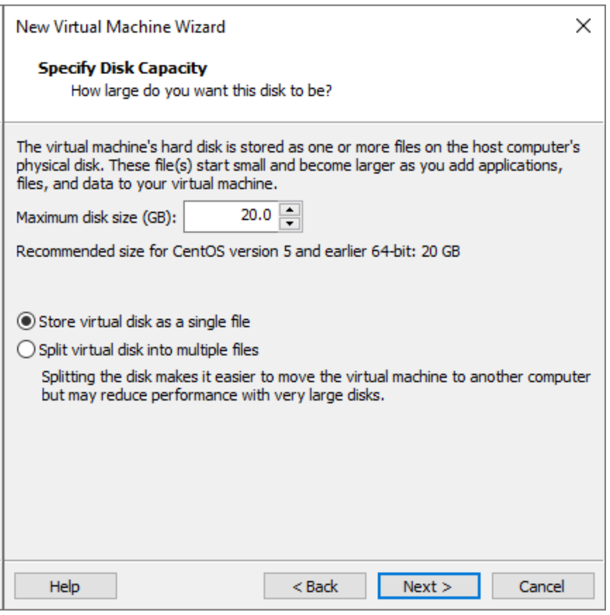


Name the Virtual Machine
What name would you like to use for this virtual machine?

Virtual machine name:
CentOS NG

Location:
C:\Users\User\Documents\Virtual Machines\CentOS NG

The default location can be changed at Edit > Preferences.



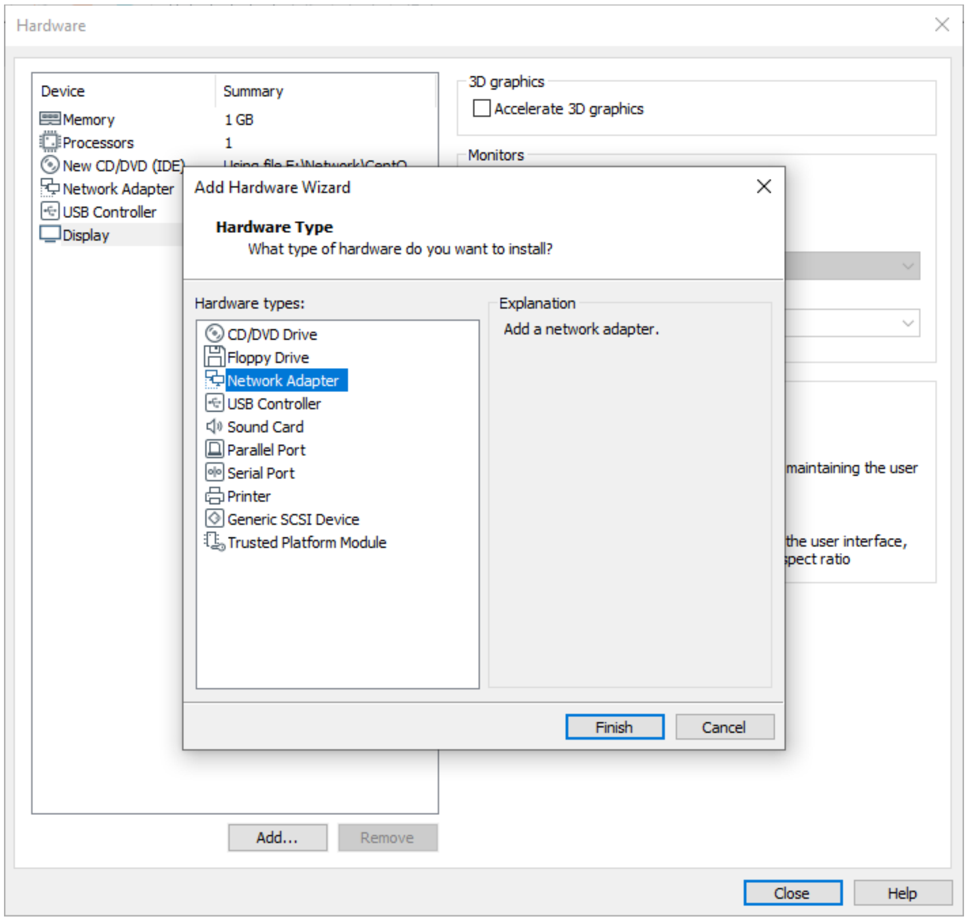
Specify Disk Capacity
How large do you want this disk to be?

The virtual machine's hard disk is stored as one or more files on the host computer's physical disk. These file(s) start small and become larger as you add applications, files, and data to your virtual machine.

Maximum disk size (GB): 20.0

Recommended size for CentOS version 5 and earlier 64-bit: 20 GB

● Store virtual disk as a single file
○ Split virtual disk into multiple files
Splitting the disk makes it easier to move the virtual machine to another computer but may reduce performance with very large disks.



Hardware

Device	Summary
Memory	1 GB
Processors	1
New CD/DVD (IDE)	Using file E:\Network\CentO
Network Adapter	
USB Controller	
Display	

3D graphics

☐ Accelerate 3D graphics

Monitors

maintaining the user

the user interface,
pect ratio

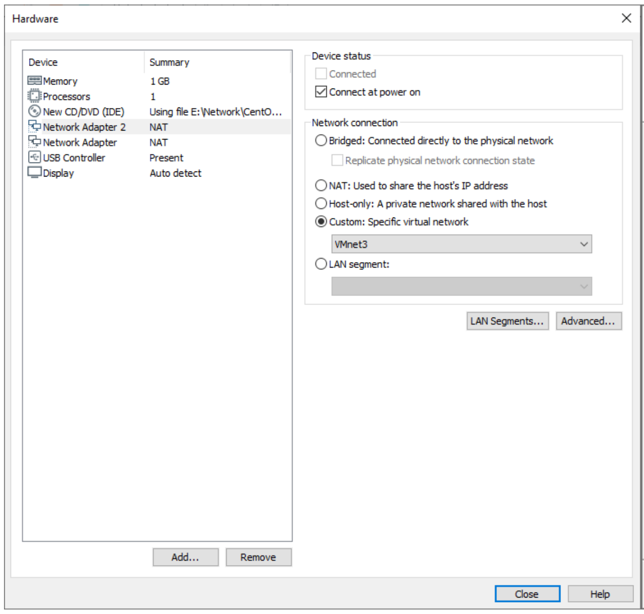
Add Hardware Wizard

Hardware Type
What type of hardware do you want to install?

Hardware types:

- CD/DVD Drive
- Floppy Drive
- Network Adapter
- USB Controller
- Sound Card
- Parallel Port
- Serial Port
- Printer
- Generic SCSI Device
- Trusted Platform Module

Explanation
Add a network adapter.

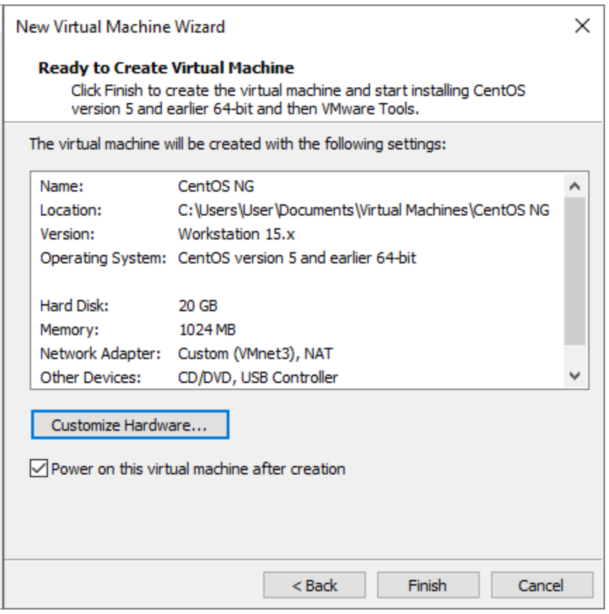


Hardware

Device	Summary
Memory	1 GB
Processors	1
New CD/DVD (IDE)	Using file E:\Network\CentO...
Network Adapter 2	NAT
Network Adapter	NAT
USB Controller	Present
Display	Auto detect

Device status
☐ Connected
☑ Connect at power on

Network connection
○ Bridged: Connected directly to the physical network
☐ Replicate physical network connection state
○ NAT: Used to share the host's IP address
○ Host-only: A private network shared with the host
● Custom: Specific virtual network
VMnet3
○ LAN segment:



New Virtual Machine Wizard

Ready to Create Virtual Machine
Click Finish to create the virtual machine and start installing CentOS version 5 and earlier 64-bit and then VMware Tools.

The virtual machine will be created with the following settings:

Name:	CentOS NG
Location:	C:\Users\User\Documents\Virtual Machines\CentOS NG
Version:	Workstation 15.x
Operating System:	CentOS version 5 and earlier 64-bit
Hard Disk:	20 GB
Memory:	1024 MB
Network Adapter:	Custom (VMnet3), NAT
Other Devices:	CD/DVD, USB Controller

☑ Power on this virtual machine after creation

Figures: Appendix Ha-g: creation of CentOS virtual machine

Appendix I: Testing Environment Subnet Information and Virtual Machine Information

Subnet ID	Subnet Mask	CIDR Notation	First Usable Host	Last Usable Host	Broadcast Address	Usage
192.168.1.0	255.255.255.224	/27	192.168.1.1	192.168.1.30	192.168.1.31	Network 1
192.168.1.32	255.255.255.224	/27	192.168.1.33	192.168.1.62	192.168.1.63	Network 2
192.168.1.64	255.255.255.224	/27	192.168.1.65	192.168.1.94	192.168.1.95	Network 3
192.168.1.96	255.255.255.252	/30	192.168.1.97	192.168.1.98	192.168.1.99	R1/R1
192.168.1.100	255.255.255.252	/30	192.168.1.101	192.168.1.101	192.168.1.102	R2/R3

Table 3: Subnets in use on the test network

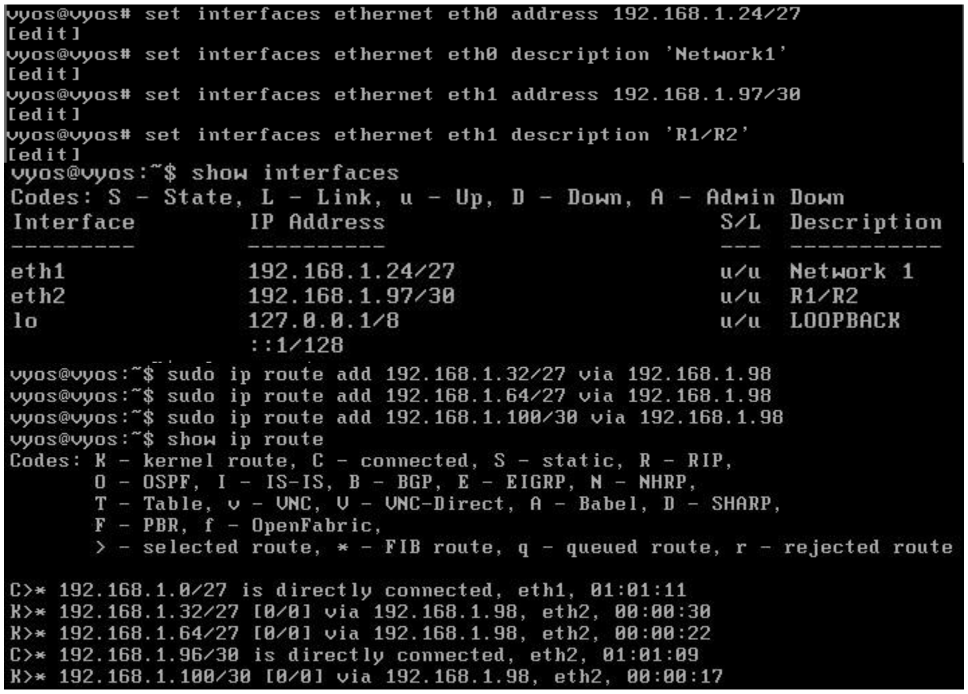
Virtual Machine	username	user password	Network1	R1/R2
Ububtu NG 2	NG	toor	192.168.1.20/27	
Ubuntu NG	NG	toor	192.168.1.21/27	
Kali NG	kali	kali	192.168.1.22/27	
Ubunutu S NG	user	toor	192.168.1.23/27	
Windows 10 NG	IEUser	P4ssword!	192.168.1.25/27	
Router1	vyos	vyos	192.168.1.24/27	192.168.1.97/30

Virtual Machine	username	user password	root password	Network2	R1/R2	R2/R3
Debian NG	user	toor	toor	192.168.1.34/27		
Fedora NG	user	toor		192.168.1.35/27		
Kubuntu	NG	toor		192.168.1.36/27		
CentOS S NG	user	toor	toor	192.168.1.37/27		
Router2	vyos	vyos		192.168.1.38/27	192.168.1.98/30	192.168.1.101/30

Virtual Machine	Username	Password	Network3	R2/R3
Mint NG	user	toor	192.168.1.66/27	
Wordpress	bitnami	bitnami	192.168.1.67/27	
Router3	vyos	vyos	192.168.1.68/27	192.168.1.102/30

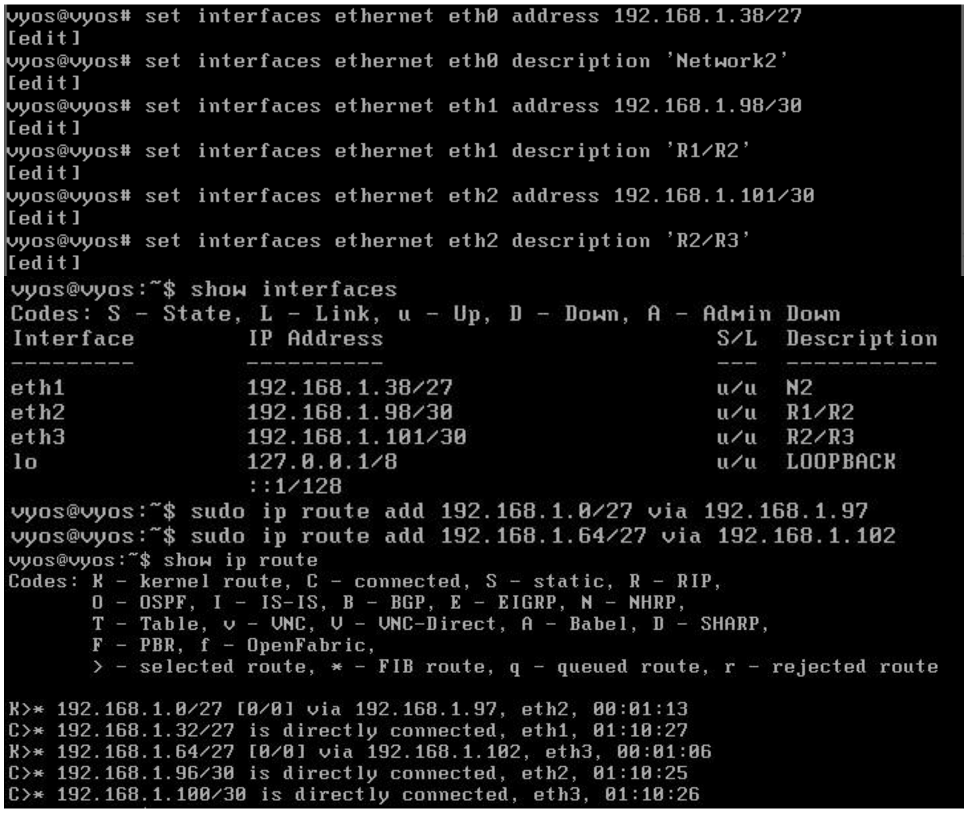
Table 4: Tables 4.1b-d: Tables containing the IP Address and account details for the virtual machines.

Appendix J: Configuration of Test Network Routers



```
vyos@vyos# set interfaces ethernet eth0 address 192.168.1.24/27
[edit]
vyos@vyos# set interfaces ethernet eth0 description 'Network1'
[edit]
vyos@vyos# set interfaces ethernet eth1 address 192.168.1.97/30
[edit]
vyos@vyos# set interfaces ethernet eth1 description 'R1/R2'
[edit]
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth1             192.168.1.24/27                   u/u  Network 1
eth2             192.168.1.97/30                   u/u  R1/R2
lo               127.0.0.1/8                       u/u  LOOPBACK
                 ::1/128
vyos@vyos:~$ sudo ip route add 192.168.1.32/27 via 192.168.1.98
vyos@vyos:~$ sudo ip route add 192.168.1.64/27 via 192.168.1.98
vyos@vyos:~$ sudo ip route add 192.168.1.100/30 via 192.168.1.98
vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

C>* 192.168.1.0/27 is directly connected, eth1, 01:01:11
K>* 192.168.1.32/27 [0/0] via 192.168.1.98, eth2, 00:00:30
K>* 192.168.1.64/27 [0/0] via 192.168.1.98, eth2, 00:00:22
C>* 192.168.1.96/30 is directly connected, eth2, 01:01:09
K>* 192.168.1.100/30 [0/0] via 192.168.1.98, eth2, 00:00:17
```

Figure 19: Configuring interfaces, interfaces and adding IP routes to router 1



```
vyos@vyos# set interfaces ethernet eth0 address 192.168.1.38/27
[edit]
vyos@vyos# set interfaces ethernet eth0 description 'Network2'
[edit]
vyos@vyos# set interfaces ethernet eth1 address 192.168.1.98/30
[edit]
vyos@vyos# set interfaces ethernet eth1 description 'R1/R2'
[edit]
vyos@vyos# set interfaces ethernet eth2 address 192.168.1.101/30
[edit]
vyos@vyos# set interfaces ethernet eth2 description 'R2/R3'
[edit]
vyos@vyos:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                        S/L  Description
---------        ----------                        ---  -----------
eth1             192.168.1.38/27                   u/u  N2
eth2             192.168.1.98/30                   u/u  R1/R2
eth3             192.168.1.101/30                  u/u  R2/R3
lo               127.0.0.1/8                       u/u  LOOPBACK
                 ::1/128
vyos@vyos:~$ sudo ip route add 192.168.1.0/27 via 192.168.1.97
vyos@vyos:~$ sudo ip route add 192.168.1.64/27 via 192.168.1.102
vyos@vyos:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, E - EIGRP, N - NHRP,
       T - Table, v - VNC, V - VNC-Direct, A - Babel, D - SHARP,
       F - PBR, f - OpenFabric,
       > - selected route, * - FIB route, q - queued route, r - rejected route

K>* 192.168.1.0/27 [0/0] via 192.168.1.97, eth2, 00:01:13
C>* 192.168.1.32/27 is directly connected, eth1, 01:10:27
K>* 192.168.1.64/27 [0/0] via 192.168.1.102, eth3, 00:01:06
C>* 192.168.1.96/30 is directly connected, eth2, 01:10:25
C>* 192.168.1.100/30 is directly connected, eth3, 01:10:26
```

Figure 20: Configuring interfaces, interfaces and adding IP routes to router 2