

# Smallwood Rugby Security Software White Paper

Evan, Lukas, Theo and Tomasz

**EH16**

CMP311: Professional Project Development & Delivery

**BSc Ethical Hacking**

2021/22

Student	Section
Evan	Results
Lukas	Discussion
Theo	Introduction
Tomasz	Method

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

# Table of Contents

---

1	Introduction .....	3
1.1	Background .....	3
1.2	Aim .....	4
2	Method .....	5
2.1	Method .....	5
3	Results.....	11
3.1	Results.....	11
4	Discussion.....	13
4.1	General Discussion.....	13
4.2	Conclusions .....	13
4.3	Future Work .....	14
4.4	Call to action .....	14
	References .....	16
	Appendices.....	17
	Appendix A - Deliverables & requirements .....	17
	Appendix B – Minutes.....	18

# 1 INTRODUCTION

## 1.1 BACKGROUND

---

Smallwood-Rugby is a small architect company with 50 employees. The two IT workers are very busy and as such would struggle to keep an eye on the whole system's security manually and as such have requested that they be provided with automated methods of performing some of the security monitoring. Smallwood-Rugby require the means to automatically assess the security of the entire network on a weekly basis. The software provided should monitor changes happening to the network as well as assess user password security, and the results should be output into a format that is user friendly and not too technical as the IT staff do not have training in regards to penetration testing.

Smallwood-Rugby are right to be concerned about network monitoring. It is not a task that should be taken lightly as the consequences of negligence can be severe. As stated in a report by McAfee, "Internal actors were responsible for 43% of data loss, half of which is intentional, half accidental" (McAfee, 2015). While this may not be a majority of data loss, it gets close to making up half of it. The accidental side of it would likely be easier to discover as the users responsible won't be attempting to cover their tracks, meaning that up to 21.5% of the data loss suffered by a business could be quite easily prevented with appropriate network monitoring, especially if complemented with appropriate staff training. An area where employees could easily accidentally compromise company security would be their password strength and indeed, according to a study performed by Lastpass in 2018 across 43000 separate businesses, companies in the 26-100 employee range, which is the range Smallwood-Rugby lies in, have a security score rated at 48 out of 100, which is classified as "fair" but clearly shows a lot of room for improvement. The security score was calculated by taking a number of factors into account, namely the number of duplicate passwords, the number of sites in use that were considered vulnerable due to prior data breaches, the number of weak passwords in use, the average password strength, the strength of shared passwords, and the multifactor authentication score (Lastpass, 2018).



The 21.5% of all data loss caused by malicious internal actors could also be mitigated through appropriate security monitoring as taking note of changes to the network could easily alert IT staff to malicious activity. Taking note of newly installed software is important as some of this software could be malware, potentially installed by malicious employees who have alterior

motives that would negatively impact the network. If this were to go unnoticed, it could either cause damage to the network, spread further in the case of a worm, or even gather sensitive information and cause serious data loss. Similarly, if users had poor password strength it would mean that a malicious party would have a far easier time gaining access to the network, either exposing sensitive information that was intended to be private or by damaging the network. An unexpected machine appearing on the network would be cause for concern as it could be a malicious device being used in an attempt to gain deeper access to the network. Removing these devices as soon as possible is a necessity to keep the network safe.

Ensuring appropriate security in a business is a cumbersome necessity, part of which is these routine checkups. However the duties of Smallwood-Rugby's IT staff include tasks outside of the large scope of security to also cover other tasks, such as setting up new parts of the system, assisting with technical issues, and maintaining equipment. Having only two employees tasked with managing and performing all of this work would be very cumbersome. Any time that could be saved would be very valuable. A program that performs some of the routine tasks the IT staff need to do would not only help improve security directly by identifying issues occurring on the system, but also by freeing up the IT staff to improve the system's overall security in other ways.

There are plenty of network monitoring tools available all over the internet, but the project that has been produced is better suited to the Smallwood-Rugby IT team than a commercial network monitoring tool because the project was custom built to the company's requirements. Commercial products will focus on a set subject area and make their tools work around this territory, meaning Smallwood-Rugby would have to try to fit their requirements into the functions of these products. The requirements laid out by Smallwood-Rugby cover a broad range of subjects, so trying to find a software solution to cover them all could result in having to either choose to neglect certain areas or pay for multiple licences which would incur additional costs. With the project produced by the project team, the driving force is Smallwood-Rugby's own requirements, meaning that the entire broad range of requirements have been covered. There is no need to neglect areas as Smallwood-Rugby are the ones defining the scope instead of an independent, unrelated company. Lastly, the project produced takes the complex results from a variety of tools and compresses them down to a clear, user friendly format whilst most network monitoring tools would be designed to output copious technical information for advanced users who could make use of it, an undesirable feature for Smallwood-Rugby's IT team.

## **1.2 AIM**

---

The aim of this project is for the project team to produce automated methods to assist with and perform the functions requested by the Smallwood-Rugby IT staff, namely:

- Run on an automated weekly schedule
- Scan for changes on the local network regarding general changes to the network, software installed on machines, and user password strength.
- Output the results into a clear and concise human-readable format that can give users an understanding of the network's health at a glance

# 2 METHOD

## 2.1 METHOD

---

The methodology chosen by the team was the Prototype Methodology because the team was able to get feedback from the client and improve the application based on the feedback.

The first part of our method was to identify the features that we can implement and how these features would be implemented.

Because of the modular form of the software, we have built we had to split the creation of separate modules and assign each of the modules to one team member.

Tomasz was responsible for the creation of the GUI interface which included executing other team members modules from the GUI itself. The GUI script was created using PySimpleGUI framework and can be easily modified to extend the software capabilities of our application and it works by executing OS commands which execute other scripts.

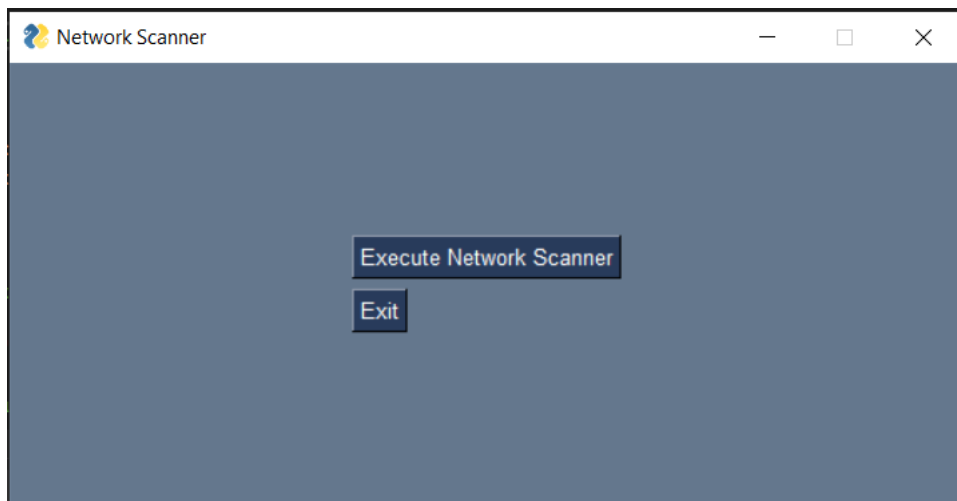


Figure 1. Graphical User Interface.

```
window = mwin.Window("Network Scanner", layout, margins=(200,100))

#loop that waits for user input from the menu

while True:
    event, values = window.read()

    if event == "Execute Network Scanner":
        os.system('python merger.py') #executes merger.py script to execute scripts one after another

    elif event == "Exit" or event == mwin.WIN_CLOSED:
        break

window.close
```

Figure 2. Snippet of gui.py code.

```
import os

#scripts that executes scripts one after another

os.system('python nmapscan.py')
os.system('python programList.py')
os.system('python password.py')
os.system('python output.py')
```

Figure 3. Snippet of merger.py code.

Theo was responsible for implementation of Nmap scanner as a script which would discover all the computer hosts and devices the client's network and save it to a file which can be used by other modules. Nmap scanner works by pinging every device on the subnet and then performing port scan on them. The script outputs the results to xml file which can be parsed by the Output module that compares the result of new scan and older scan and shows the results in the HTML file.

```
Scan = nmap.PortScanner() #Declaring an object for the nmap scan called Scan

if os.path.exists("oldOutput.xml"): #Changing the name of the last output to oldOutput.xml so outputs can be compared
    os.remove("oldOutput.xml")

if os.path.exists("newOutput.xml"):
    os.rename("newOutput.xml", "oldOutput.xml")

hostFile = open('targets.txt') #Opening the file containing the IP addresses to scan
ipFile = open("IP list.txt", "w") #Opening the file to output the IP address of every host found to be up
xmlOutput = open("newOutput.xml", "wb") #Opening the file to store the full output in XML format
targets = hostFile.read()
hostFile.close()
first = True
```

Figure 4. Snippet of nmapscan.py code.

Evan was responsible for creating a script that was responsible for detecting software changes on host computers. It works by using SSH protocol to login to every defined computer host, executes the softwareList.bat batch file which lists all the installed applications on defined host. It also saves the results in a way that compared new results against previous ones and then save them into MasterFile.csv which can be parsed by Output module to generate HTML file.

```
save_path = 'Results' # Directory for all results
absolute_file = str("") # Unique File name bundled with Directory (C:/Results/"PCname"/)
result_file = ""
sub_directory = ""
machine_name = "" # Used for writing to Results file and MasterFile
master_file = "Results/MasterFile.csv"
```

Figure 5. Snippet of programList.py code.

```
def remote_connection(): # Code to connect to other machines, run batch file and write output to file
    global machine_name
    hosts = ['192.168.0.32', '192.168.0.33'] # List of IPs for client PCs

    for host in hosts:
        client = SSHClient()
        user = os.getlogin()
        client.load_host_keys('C:/Users/' + user + '/.ssh/known_hosts')
        client.set_missing_host_key_policy(AutoAddPolicy())
        client.load_system_host_keys()

        # Do not alter program to connect using password, as it will be visible in plaintext in this file
        client.connect(host, username='SecurityAdmin') # No password as SSH keys are used

        host_command = 'hostname' # Command to retrieve name of PC
        commands = ['softwareList.bat'] # Name of batch file on Remote PC

        stdin, stdout, stderr = client.exec_command(host_command) # Command to receive name of PC
        stdin.close()
```

Figure 6. Snippet of programList.py code.

Luke was responsible for creating a script that would retrieve hashes of the passwords on the computer hosts, crack them and compare against dictionary of weak and easy guessable passwords. The module works by executing secretsdump.exe and logging in to the Domain Controller to retrieve all the hashes of users in active directory. Then it uses John the Ripper password cracker against dictionary file and outputs the usernames of the accounts that the passwords were cracked. The passwords dictionary can be easily modified to include as many passwords as possible or even use passwords from different and bigger dictionary to make sure that the users have created their passwords in a complex way.

```
import os

def password_list():
    password=[]
    try:
        os.system("secretsdump.exe -just-dc-ntlm smallwood/Administrator:Password1@192.168.0.100 > hashes.txt")#t
        os.system("john.exe --format=NT --rules -w=dictionary.txt hashes.txt")#these next commands decrypt the pa
        os.system("john.exe --show --format=NT hashes.txt > password.txt")
        f = open('password.txt', 'r')#opens decrypted passwords file
        password = f.readlines()
    except IOError as inst:
        print('ERROR', inst.errno, inst.strerror)
    return password
```

Figure 7. Snippet of password.py code.

Lukas was responsible to get the data from the previous modules outputs and present them in visually pleasing and easy to read format for the client. The module works by taking outputs from the Nmapscan, programList and password modules separately. Then it is showing differences in Nmapscan results from previous scan, lists all installed software since initial setup (including banned software) and the percentage number of the passwords cracked along with the usernames list which the passwords were cracked.

```

import pyndiff
import dominate
from dominate.tags import *

def createDiff():
    # create diff file from the two xml files
    diff = pyndiff.generate_diff(
        "oldOutput.xml",
        "newOutput.xml",
        ignore_udp_open_filtered=False,
        output_type="txt"
    )

    file = open('diff.txt', 'w')
    file.write(diff)
    file.close()

```

Figure 8. Snippet of output.py code.

```

def main():
    # create report document
    report = dominate.document(title="Smallwood Rugby Security Report")

    # setup doc
    setupDoc(report)

    # insert first paragraph
    insertParagraph(report, 'This report is a human readable version of the security software results.', "")

    # add line
    report.add(hr())

    # create diff file for nmap output
    createDiff()

    # filter nmap output and add to report
    nmapOutput(report)

    # filter programFile output and add to report
    programOutput(report)

    # filter passwords output and add to report
    passwordOutput(report)

    # upload document
    uploadReport = open('index.html', 'w')
    uploadReport.write(report.render())
    uploadReport.close()

```

Figure 9. Snippet of output.py code.

The team has decided to use Python language as the main language for our software application due to its ease of creating scripts and modify them later if the client wished us to. Other than that Python language had advantage of the fact that the scripts did not have to be compiled every time a change was made.



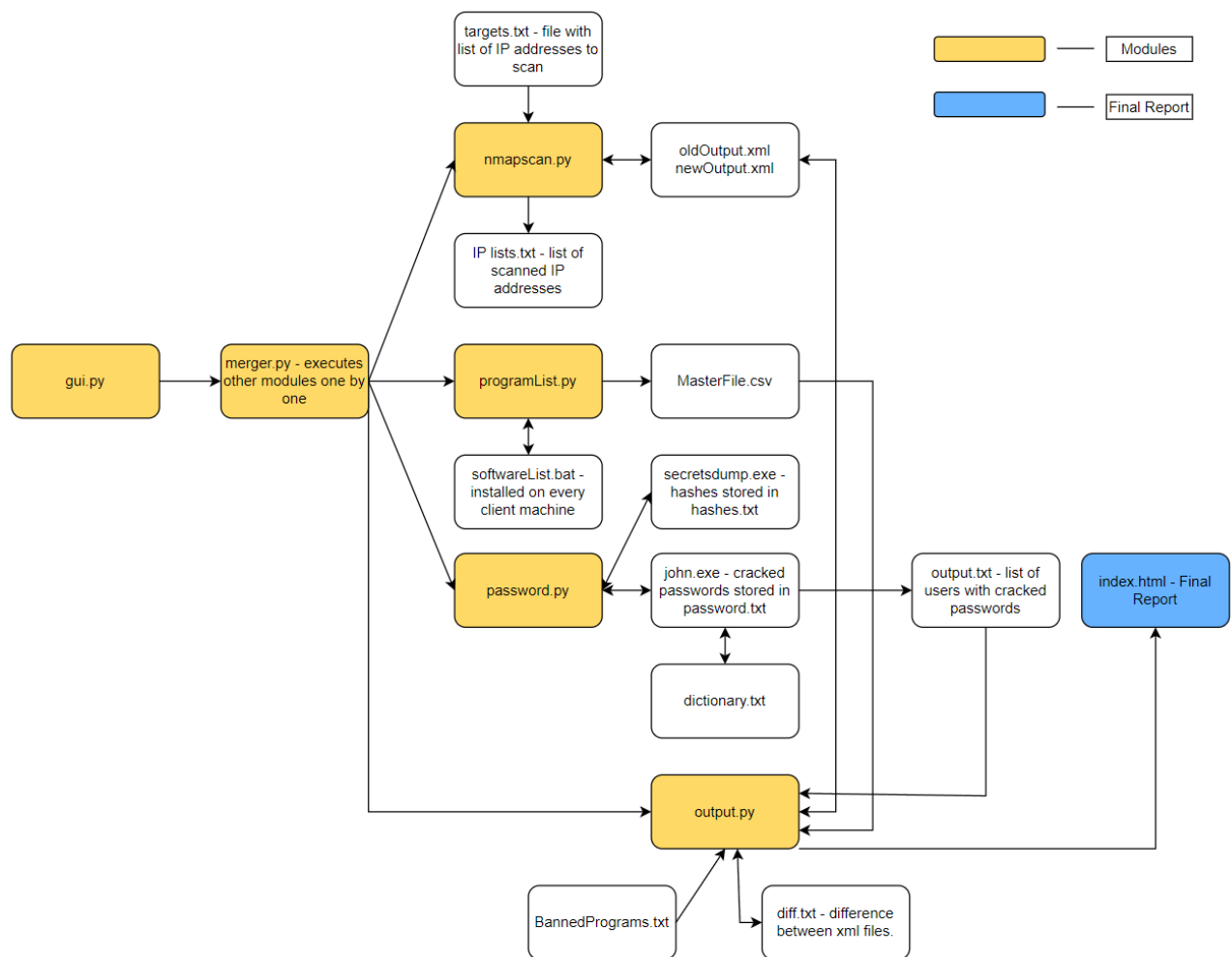


Figure 10. Diagram showing program workflow from start to finish.

To test the prototypes of our software a virtual environment had to be created. VMware Workstation was utilised to create virtual machines – one for Windows Server 2019 with GUI experience and additional three for Windows 10 clients. The virtual environment was created in a way that the virtual machines were on their own separate subnet network to simulate real life client's network environment. Apart from this another Windows 10 virtual machine was created to simulate the client's spare machine. As it was used to execute the scripts and gather the results it did not have to be connected to the domain but still had to be in the same subnet. Windows operating system had to be chosen for the machine that executes our software application because one part of the module utilised Windows batch file which functionality could not be replicated on Linux operating systems.

When Windows Server 2019 was installed, it was promoted to Domain Controller and all the Windows 10 client machines were added to the domain. To do that the server had to have assigned static IP address and additional server roles had to be installed - "Active Directory and Services" and "DNS server." Using Active Directory user accounts were added to the domain users for testing purposes.

The team has decided to have weekly online meeting discussing our progress, working out solutions to problems that have arisen during development of our software application and produce fresh solutions on how to improve the application. The online meetings were not necessary to be done each week as the team was very productive and it did not take much time to create a finished product.

The meetings were hold on a Discord server where we could implement separate channels for meetings, found resources that could be implemented in our software solution, power point presentation, and white paper discussions. That approach gave us opportunity to limit the number of online meetings since most information was shared and visible to every group member.

The GitHub repository was created to ensure that all team members work in synchronised fashion. GitHub is a great Team Software Collaboration tool due to its features such as Adding Team Members, Pull Requests, Bug Tracking, Analytics, Project Management, Continuous Integration, Code Review and Documenting. GitHub proved to be essential for our team because each team member was able to create their own branch of the module they were working on and update the main branch with every incrementation (newer version / fixed version) of their module. Thanks to that all the users could see and track the effects of other team member's work.

Apart from that the team had meeting with the client to show results of our work and get feedback needed to improve our software solution. The client was pleased with the progress the team was making.

## 3 RESULTS

### 3.1 RESULTS

---

This section of the report will discuss the Results produced by the programming, discussing the readability of the report, total run time of the program, as well as the accuracy of the different aspects of the program.

During the development of our Project, one of the big goals was to make sure the output of the program was non-technical and easy to understand. The layout of the report makes it easy to read through and understand. One of the features that helps make our reports readable at a glance is the traffic light system. By simply using the colours Green, yellow, and red, we can convey to the reader a very low-level sense of the state of security. This lets the IT staff quickly assess the situation and move to fix any issues present in the shortest amount of time, which can be critical if there is a serious security vulnerability present.

On top of this, the report is created using HTML. This is important, as it means the report can be accessed, or sent to and read, on a wide range of devices. This ties into the readability, making sure our reports are able to be read on as many different systems as possible. In comparison, a Microsoft Word document, whilst easy to use, wouldn't offer as much compatibility across all potential devices.

The report file is laid out in a manner that makes it easy to interpret and understand. This is achieved through the use of italics to differentiate between text, good use of white space to separate lines blocks of text from one and another, as well as samples of code having less space so that the overall document is not bloated. To further add to the report's readability, HTML5, when opened with Google Chrome, uses the Times New Roman font. This font is very presentable and easy on the eyes, ensuring the user has no difficulty reading the report.

In addition to this, the program is able to complete all processes in a short amount of time. Although this isn't a requirement explicitly laid out by the client, it was of great import that our program was able to run in a timely manner. Data was gathered on the total run time of the individual modules and will be presented in tables below, along with their specific test cases, allowing the data gathered to be scaled up with at least some accuracy.

The program finder was tested against two example Client PCs, the Network Scanner conducted a local scan only (254 Possible IP Addresses), and the Password Cracker completed the retrieval of 50 user password hashes.

Run Time	Program Finder	Network Scanner	Password Cracker
Test 1	2.8 seconds	55 seconds	3.1 seconds
Test 2	2.1 seconds	55 seconds	2.9 seconds
Test 3	2.5 seconds	50 seconds	3.5 seconds
Average Time	2.5 seconds	53.33 seconds	3.2 seconds

In total, the programs core modules in our testing took just under a minute to perform all necessary procedures. This is only a reference time and will most likely be longer when implemented at Smallwood Rugby due to greater complexity of the network(s) and the increase in devices/users. The Program Finder module was by the far the quickest, although it only had two devices to connect to and retrieve a list of programs from. The Project team is aware that Smallwood Rugby currently employs at least 50 employees. This allows us to scale the data up, letting us say with a high degree of certainty that connecting to fifty devices would take just over two minutes, an impressively short amount of time for such an operation.

As mentioned, the Network scanner data comes from conducting a local scan only. The layout of Small Wood Rugby's network(s) is not known to our team, however from our data it is possible to infer potential run times on smaller or larger networks. For example, a network roughly half the size would take approximately half the time to scan.

Overall, our program should always gather its results and produce its report in a handful of minutes.

The Password Cracker was able to successfully crack five passwords against a small list of hashed common passwords. This gives it a very high success rate. Making sure all passwords are secure is of the upmost importance, as a weak employee password can rend all other security measures ineffective.

During our internal testing, the Network Scanner was able to successfully scan and identify all devices on the target network, returning a list of IP addresses, MAC addresses, and open ports on each device. The gathering of MAC addresses is significant, as it allows the Network Scanner to effectively recognises new devices, regardless of its current IP address.

Whilst testing the Program Finder, it successfully managed to find programs that should not be installed on the Client PCs. The application was Mozilla Firefox and was successfully recorded in the report as present on both Client PC 1 and Client PC 2.

Once this program was uninstalled on both these devices, the Program Finder was run again. This time it successfully reported back that there were no unauthorised programs present on either device.

Overall, the EH16 Project Program can successfully produce accurate, readable, and understandable results presented in a non-technical manner. This is achieved through a mix of features, design choices, and Python coding. The use of the traffic light system, HTML, and general report layout all contribute to an excellent report output. The Python coding working in the background successfully manages to complete all tasks in a timely manner, returning accurate results to be displayed in the report.

# 4 DISCUSSION

## 4.1 GENERAL DISCUSSION

---

The project team requirements outlined in **Appendix A - Deliverables & requirements** were successfully met by the project team. The GUI is simple for the client to use, and the outputted report is straightforward to understand immediately and includes all relevant information. On top of this, the entire program runtime is significantly shorter than initially thought, meaning the whole program could be run in a couple of minutes. Having an efficient program allows the IT team to run the program early in the morning and quickly understand the network's status. Additionally, this ensures that if any irregularities are found, the IT team can promptly deal with them before they significantly impact the company.

The benefit of using 'ndiff' when comparing the old and new nmap XML outputs is that any changes between the files are noted, including ports and MAC addresses. This ensures that even if a new device connects with the same IP as an old device, the new device would still appear in the report.

One of the most beneficial implementations in the software is the password cracking feature, as protecting the rest of the network becomes redundant if the employee passwords are insecure and easy to crack. First Contact reported that 51% of people use the same password for their work accounts as their personal accounts (DataProt, 2022). Not using unique passwords leaves employee accounts more vulnerable to attacks which use password lists of previously compromised accounts. Having this feature in the software helps identify accounts with simple passwords, meaning the IT staff can request they reset their password and change it to something more secure. Additionally, the word list used to crack the passwords can be updated by replacing the text file 'dictionary.txt', making it easy to maintain this part of the project and keep it relevant.

As noted from the original project brief, Smallwood Rugby's main concerns regarding the software was the output being non-technical to ensure the IT staff who do not have penetration testing experience can understand the issues that the software flagged. The project team was able to meet the client's requirement by formatting the report in a 'human-readable' way, which was done by parsing the data and returning it in paragraphs rather than in the original XML and CSV format. The traffic light system used within the report also makes it very easy to quickly identify the main areas of concern to help the IT staff prioritise their workload.

## 4.2 CONCLUSIONS

---

Overall, the automated security software produced for Smallwood Rugby was successful as the produced software meets all of the clients' requirements. Accordingly, the program would give Smallwood Rugby a helpful insight into the security of their current systems and assist them in prioritising any concerns that may arise from the software's output.

The requirements were met by creating several python scripts linked together using a simple, one-button GUI and then feeding the information from those scripts to a final python script that outputs the data in a non-technical, human-readable format. The final report also uses a traffic light system to alert the staff to any concerns immediately, with green indicating no issues, yellow indicating a minor/non-urgent issue and red indicating an issue of significant importance.

As well as the GUI and the HTML report, the software would benefit the company with the following features:

- A log of any differences surrounding devices connected to the network, including open ports and MAC addresses

- Finding out if any additional software has been installed on employee devices, noting any banned programs
- Verifies the security of employee passwords by attempting to crack them using a hashed password list

Additionally, due to the program being split into several different scripts and well commented, it should be easy to maintain, and update should the company go through any significant changes that would affect its effectiveness. Finally, should the company need to add additional scripts to the program, this can be easily added to the GUI program, which uses OS calls to initialise the other scripts.

As Smallwood Rugby currently has no indication of how secure its systems are, the project team's solution would be an excellent way to discover any threats/concerns within the network. Therefore, the software would enable the company to address these issues and ensure that its network is secure from cybersecurity threats.

### 4.3 FUTURE WORK

---

Although the project team met the clients' requirements, there are several points worth discussing regarding possible future work and adjustments if the project team was allocated extra time and resources.

Currently, the nmap scan is not customisable from the GUI, but instead reads the targets from a 'targets.txt' file which is slightly more inconvenient. A slightly more convenient solution would be to introduce a text box within the GUI, allowing for quick and easy adjustments should the company make network changes. As well as this, the Nmap program currently only runs port scanning. On top of this, Nmap scripts could be introduced to detect vulnerabilities within the network. Adding Nmap scripts would be simple to implement within the 'nmapscan.py' program however would require significant changes to the 'output.py' program to ensure the results are displayed correctly.

In its current state, the program finder script only runs on Windows machines, as it uses a '.bat' file that runs a PowerShell command. Instead, the script could be adjusted to determine the local hosts' OS and then run the appropriate command for that OS, or a solution that works on all OS could be further researched.

Additionally, when the HTML report is generated, it writes to the local directory, meaning it can only be accessed from the device that runs the scan unless the IT staff sets up an alternative method to connect to the device. Instead, the program's report could be uploaded to a website hosted within the network, allowing it to be accessed remotely on multiple devices, including work phones. There would also be room for changes to the HTML and CSS of the site to improve its visual appearance.

Lastly, the GUI could be updated to allow the python modules to be configurable, allowing them to be disabled or reenabled appropriately. Allowing the GUI to configure the modules should also simplify implementing new modules into the program. There is also room for the GUI to be redesigned to be more visually appealing.

### 4.4 CALL TO ACTION

---

The project team would be happy to initially set up the program for Smallwood Rugby by coming into the workplace and setting up all the scripts on the spare machine to ensure that it is working as expected. Furthermore, the project team would also be prepared to have an initial meeting with all relevant staff to address any potential queries that may arise from introducing this software.

Should anything unexpected occur during the initial setup of the software, the project team would ensure that this is appropriately fixed before parting with the client. Furthermore, if there are minor changes that the client feeds back to the team, these can be implemented with no additional cost to the client.

Finally, the team is happy to contact the client regarding future projects and project updates. Should any future inquiries arise, these can be directed to [Team@EH16.com](mailto:Team@EH16.com).

## REFERENCES

DataProt, 2022. *DataProt*. [Online]

Available at: <https://dataprot.net/statistics/password-statistics/>

[Accessed 28 April 2022].

Mcafee, 2015. *Grand Theft Data – Data Exfiltration Study: Actors, Tactics, and Detection*. [online]

Mcafee.com. Available at: <https://www.mcafee.com/enterprise/en-us/assets/reports/rp-data-exfiltration.pdf> [Accessed 6 May 2022].

Lastpass, 2018. The 2018 Global Password Security Report. [online] lastpass.com. Available at: [https://lp-cdn.lastpass.com/lporcamedia/document-library/lastpass/pdf/en/IAM\\_LastPass\\_SOTP\\_ebook.pdf](https://lp-cdn.lastpass.com/lporcamedia/document-library/lastpass/pdf/en/IAM_LastPass_SOTP_ebook.pdf)

[Accessed 7 May 2022].

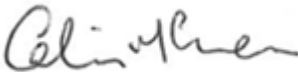


# APPENDICES

## APPENDIX A - DELIVERABLES & REQUIREMENTS

---

### Agreement Form: Project Deliverables

<b>Group Name, Names of Team Members, and Programme</b>	<u>TEAM EH16</u> Lukas Evan Tomasz Luke Theo
<b>Subject specialist's Name (Client)</b>	<b>Colin McLean</b>
<b>The deliverables listed below will be submitted by the team by the due date.</b>	
<b>Deliverables</b>	Python Script to output to Readable Report Python script with scheduling option for timed scans, runs nmap scans on network, checks against previous scans for discrepancies and runs user passwords hashes against a common password list. Network Testbed
<b>Subject specialist's (Client) signature</b>	
<b>Team members' signatures</b>	<b>Lukas</b> <b>Theodore</b> <b>Luke</b> <b>Evan</b> <b>Tomasz</b>

# Agreement Form: Requirements

Group Name: EH16

Team members (print): LUKAS, EVAN, LUKE, THEO, TOMASZ

Project Title: Smallwood Rugby Security Software

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

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

ID	Functional Requirements
1	Detects new devices on the network
2	Detects changes in devices on the network
3	Analyses user passwords
4	Outputs in an easy to access format
5	Ability to schedule scans
6	Simple interactive UI
ID	Non-Functional Requirements
1	Program runtime should take less than 12 hours
2	Software output should be accessible and non technical
3	Program will be kept relevant by allowing for user input to update scan boundaries
4	Software will be tested thoroughly to ensure robustness

Stakeholders	Signatures	Date
Team members	Evan Theodore Luke Lukas Tomasz	11/02/2022
Subject Specialist	<i>A. U. Githin</i>	24/02/2022
Client	Colin McLean	12/02/2022

## APPENDIX B – MINUTES

---

See activity logs