Abertay University

Developing a Penetration Testing Toolkit to aid in Global Security through Automation and Smart Reporting

Thomas MacKinnon

School of Design and Informatics
Abertay University BSc(Hons) Ethical Hacking, 2021

supervised by
Ross Heenan

Acknowledgements

Firstly, I would like to thank my supervisor Ross Heenan, for his amazing support, expertise, and encouragement throughout this project. His calm and friendly demeanour was exactly the kind of moral support I needed, and was invaluable in helping me stay on track throughout development.

I would also like to thank my Mother and Father, who have supported more than I ever could have asked for throughout my education. With that I would like to thank my brothers for keeping me grounded and working throughout University.

Finally I would like to thank my friends and any other family who have aided in any way throughout these four long years. This project would not have been successful without the help of these people, and I thank them greatly for it.

Abstract

This project follows the development of Inquisitor, an automated Penetration testing toolkit with a focus on reporting results to a Smart and clean document. The project was created to solve the issue of unpatched vulnerabilities found in small businesses and individual's systems, due to their lack of resources to afford manual Penetration testing. The toolkit is free and designed with novice users in mind, so is relatively easy to use. Inquisitor was developed using existing academic literature to form an idea of what the product could potentially be.

Modular Python functions were used to construct Inquisitor from the ground up, focusing on thoroughly scanning the targets and securely sending results through email. The final program was tested against two very similar toolkits, with the effectiveness of each solution discussed at great length through a series of testing metrics.

The final version of Inquisitor was able to accomplish all aims of the project, conducting a full Penetration test, reporting all the results into a stripped report, and securely sending the results to the tester.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Cybercrime is on the rise and is the greatest threat facing the public, businesses, and even governments in our rapidly evolving digital age. The everyday incorporation of technology has vastly improved our lives, through connecting the world, making existing systems more effective, and creating whole new industries around it. This great boon has had a positive effect on people around the globe, but has also opened the door to a whole new type of crime, aptly named Cybercrime.

Cybercrime comes in many shapes and sizes, and can attack in a variety of ways, meaning that a system has to be protected from a huge array of possible vulnerabilities just to be considered safe. Phishing is a very popular method of cybercrime, which occurs when users are tricked into entering sensitive information (passwords, credit card details) into a fake input form from what they believe is a trusted source. Ransomware is another well known attack method, gaining popularity after WannaCry caused massive distribution and electronic damage to Britain's National Health Service(NHS). It works by encrypting all of the data on the machine and refusing to return it to the owner until a ransom has been paid. In the NHS the issue arose due to systematic failure to patch vulnerabilities, which is the most prevalent problem in the industry. Thousands of cyberattacks occur each year that could have been prevented if the owner had simply patched their system, with an estimated 30% of organisations suffering a data-breach due to this type of attack (Lapena, 2019). More and more vulnerabilities are being discovered year, as shown by figure 1.1, increasing the likelihood of more of these kinds of attacks.

Cybercrime does not just cause businesses monetary loss, but can also severely impact company reputation if the public becomes aware of the attack. Trust is essential for customers, as they often trustingly submit their address, bank details, and other personal information which they assume the company will store safely. Failure to properly protect customer information can lead to huge losses in sales, as your customer base moves to a competitor with greater apparent security. This can be a death sentence for small businesses as they do not have the assets to remain afloat without customer trust.

The primary solution to this issue is the conduction of Penetration Tests, which are high level security audits. In these, a knowledgeable tester acts as a malicious hacker and attempts simulated cyberattacks against the system to expose weakness and vulnerability. The findings of these tests are collated into an easy to understand report, detailing how attacks were performed and what new countermeasures can be implemented to avoid cybercrime.

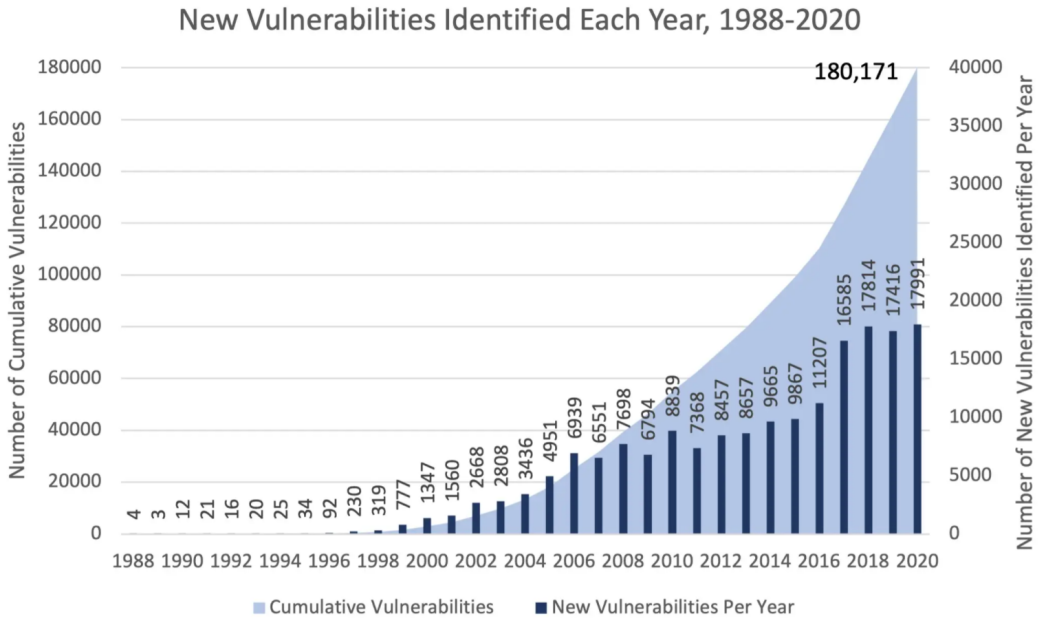


Figure 1.1: Graph displaying the increasing number of Vulnerabilities found per year (Singleton, 2021)

Penetration testing works through a series of stages, aiming to acquire new and unique information for the final report. The first stage is Information Gathering, were basic information about target is discovered, such as port activity, finding hidden hosts, and what operating systems are being run on the network. This information is then used in the second stage, named Vulnerability Scanning, which aims to properly assess the security of the target and detail any possible areas of weakness. Exploitation testing follows, where the tester uses the found vulnerabilities to attempt to exploit the target, aiming to find sensitive information or anything a malicious hacker would want. This process provides a detailed view on the security of a system, and is essential in keeping businesses safe from cybercrime.

However, a proper Penetration Test is very expensive to conduct, and can often cause disruption to everyday business, tests also take significant time to complete. The negatives prevent Penetration Testing from being utilised more in this digital age and often is completely unaffordable for small businesses or individuals wishing to make their system more secure. This leaves a significant number of networks vulnerable, either due to neglect, lack of funds, or simple naivety that network are already sufficiently safe.

This project aims to address this issue, by developing an Automated Penetration Testing Toolkit that provides a full security audit on a target and produces a Smart Report of clean results. The toolkit is named Inquisitor, which instead of hunting out heretics like the Spanish Inquisition it instead finds and reports vulnerabilities to the relevant authority. Inquisitor will run in the background of everyday business, subtly scanning and reporting back to the tester's email address, with no cost attached. This solution would provide businesses, small or large, the ability to properly test their security and greatly reduce the threat of Cybercrime.

1.2 Aim

This project will develop a toolkit that will provide users with the resources they need in order to properly evaluate and improve their security through a standard Penetration testing method using automation and effective reporting.

1.3 Research Question

Three research questions were constructed before major research or development started, aiming to answer each after the project was finished. The question helped keep the research and development focused on meeting the objectives of the project, and aided in properly evaluating the finished product.

1. What can Automation of Penetration tests combined with Smart Reporting do to help improve global security?

This question is focused on throughout Chapter 2: Literature Review, where the subject is thoroughly analysed through a series of Academic papers.

2. How effective, efficient, and fast can a Automated Penetration Test be?

This question is answered in Chapter 4: Results, where several pieces of software similar to Inquisitor are tested against each other to find the best solution, which is further discussed in Chapter 5: Discussion.

3. What Similar projects have been conducted into Automation of Penetration tests, and how effective are these solutions?

This question will be answered in the Chapter 2: Literature Review, where several different Research papers about the topic are discussed and evaluated. Chapter 4: Results also covers some Automated Penetration Testing Toolkits, with the product being compared to Inquisitor in Chapter 5: Discussion.

1.4 Project Scope

This project will involve the development of a toolkit that performs an automated Penetration test on a target or multiple hosts. The toolkit will extensively gather information from the target, and use this to conduct vulnerability scans to evaluate the security of the system. If possible, the target will be exploited using relatively safe techniques as to not disturb or damage the system.

All the findings from the test will be displayed in an aesthetically pleasing report, featuring essential information from the results, with an additional report containing raw results. The reports will be sent securely to the tester and cleaned off the system. The design of the toolkit will be formed from existing academic research, and will be developed with an appropriate design methodology.

1.5 Dissertation Structure

This Paper has been divided into chapters for each major topic covered. Chapter Two involves an in-depth Literature Review of Academic Papers related to Security Automation and background in the subject of Penetration tests, ending in evaluation of some existing toolkits that have been developed. Chapter Three presents the design methodology used to build Inquisitor, and thoroughly goes through the development of the toolkit, explaining why certain features were utilised. Chapter Four details the evaluation method used to test Inquisitor, displaying the raw results from the different toolkits. Chapter Five discusses the meaning of the results from testing, all of the relevant factors and issues that affected development, and how Inquisitor meets the aims set out in the project proposal. Chapter Six concludes the paper, giving an overview and evaluation of the final product, with suggestions for any future work that could be conducted.

Chapter 2

Literature Review

This chapter aims to give a review of existing literature on the topic of Penetration testing and projects related to automation of security auditing, with the aim of highlighting the problems that this project will solve. This is achieved through a detailed background on Penetration testing, followed by the drawbacks of security auditing, which leads into the proposed solution through automation and an overview of similar projects.

2.1 Background on Penetration Testing

Penetration testing may seem like a new concept in our technologically evolving world, but has in fact existed since the late 1960s ([5] Chu, G. & Lisitsa, A., 2018). The US military constructed "Tiger Teams" of highly skilled computer security experts to test how well their systems would handle an attack from a malicious party, which resulted in the majority of systems failing quickly. This is by no means a new concept, as throughout all of human history ancient leaders have attempted to counteract enemy attacks through mock battles and stratagems (Alpine Security, 2021). Penetration testing is a natural evolution as more of our lives and assets go online. Groš and Kovačević (2020) in [7] highlight that these early Penetration Tests were actually incorporated with the military term called "Red Teaming", where potential attacks from an enemy state, in most cases the Soviet Union (hence the name), were simulated in order to find weak spots and properly secure their base. This term has continued on in the corporate world, in scenarios where Red teamers will attempt to compromise a willing company through various methods like social engineering, phishing attacks, and most often Penetration testing.

Penetration tests, as defined by Geer and Harthorne (2002) in their rather romanticised paper on testing [6], is the art of illegitimate acquisition of legitimate authority, contrasting to that of a malicious hacker, who acts without the approval of authority, therefore condemning their actions to be illegal. This legality has led to the birth of a vital and growing industry of ethical hackers, and will be the primary tool of defense for the future. Tests are often segmented into specific sections as detailed in the "Penetration Testing Execution Standard (PTES)", compromising of Information Scanning, Vulnerability Analysis, Exploitation Testing and Reporting ([13] The PTES team, 2017), aiming to find flaws with the system before an outsider does. Tests are performed by seasoned security experts who specialise in Penetration testing, where they systematically go through the consenting network, at the end producing a detailed report on all their findings and prevention methods. Groš and Kovačević (2020) in [7] state that every Penetration tests has four core goals , being: to evaluate the defense system of the target, to see how well the employees of the target are

prepared for an attack, how well the target can monitor malicious activities, and finally how well the target can respond to attacks.

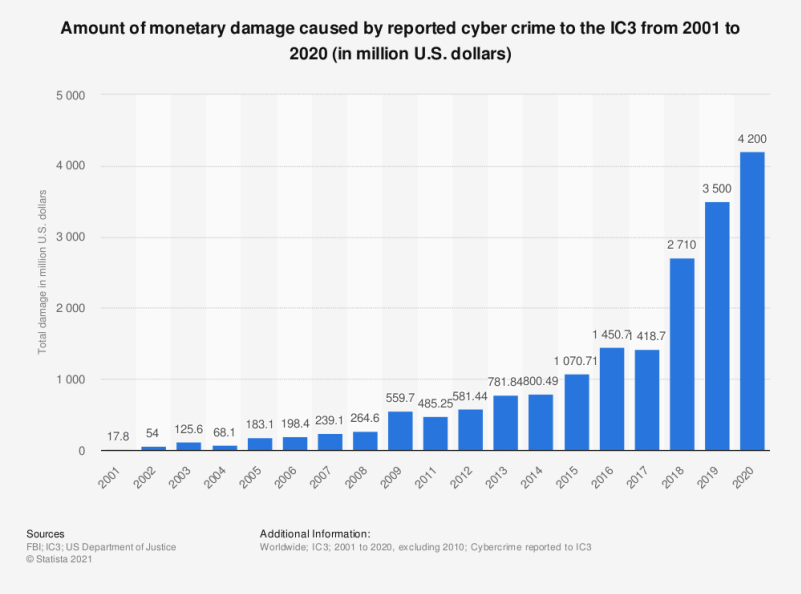


Figure 2.1: Graph displaying the rising monetary damage caused by Cyber Crime (Johnson, 2021)

The need for Penetration testing is ever increasing, new methods of attacks are sprouting up everyday, so it is vital that businesses protect their data and customers from exploitation. The cost of not securing your network is growing every year, figure 2.1 shows the increasing monetary damages caused by Cyber crime. Proper Security auditing is becoming a larger task as technology improves, Chu and Lisitsa (2018) in [5] express how the adoption of IoT (Internet of Things) Devices into businesses provides a great variety of new potential attacks to be exploited on the system. The vast majority of businesses are ill-equipped to deal with this alone, and so the practice of Penetration testing is the best way to mitigate this flood of new threats.

2.2 Drawbacks of Penetration Testing

Penetration testing may seem like a great way to audit your network, but it is by no means a perfect practice. Since the model of Penetration Testing was created by the US military it makes it rather hard for small businesses and individuals to adapt it to their needs. Stefinko et al. (2016) in [11] highlights the issue of cost of Penetration tests, as the price can go anywhere from £1,000 up to £20,000 depending on the complexity and size of the test. The authors explain further that this high cost leads many organisations to only conduct Penetration Tests at important milestones or after significant upgrades. The time involved is also an issue, as tests can easily run for a month or more, which could cause interruptions in business and availability of resources that might be vulnerable. These two factors are the main reason businesses neglect security audits, as the cost is always out weighted by the naive chance their system is completely safe.

Alumbairik and Wills (2016) in their paper on Automated Penetration tests in [1] put a spotlight

on the core flaw of Penetration tests, being that they can never be a hundred percent guarantee of a secure system. Testers check that the target is not vulnerable to a large variety of vulnerabilities, this is done through the MITRE corporations "Common Vulnerabilities and Exposure (CVE)" database, however not all existing threats can be checked as some are not documented, leaving the system with the potential risk from an unknown attack. Testers are also human and therefore make mistakes, a single missed vulnerability could comprise the entire business. The lack of a perfect test to make your system invulnerable to all threats leaves business owners disillusioned at the idea of conducting a Penetration test, as even after shelling out a lot of money their network could still be weak.

This reality often leads to organisations neglecting security in favour of other ventures, however, hiring a cheap tester can often be more dangerous than hiring none at all. Chopra et al. (2020) in [4] reveals that an immature tester increases the risk of an attack, as they have no professional standards that they have to abide by, leading to potential blackmail or breaches. Allowing the immature tester to have access to source code and back-end databases is not worth the risk of their cheap fee, making the costly option the only safe option for manual testing.

The drawbacks of proper testing are likely to be permanent, the boom of new technology being incorporated into our daily lives inevitably leads to more weak spots that an attacker can exploit. IoT devices have rapidly become essential to our homes, healthcare and businesses with little thought of the security risks they present ([3] Chandan & Khairnar, 2018). To properly test a network the Penetration tester must be knowledgeable in emerging technologies, which in turn increases other costs, like time spent testing and price of the test. Although Penetration testing is the most effective way to protect your network and customers, the negative side is often too much for small businesses and individuals to afford, leaving hundreds of thousands of organisations around the globe vulnerable.

2.3 Benefits of Automation

Automation of Penetration tests could be the key to solve this critical issue in the world of online security. The proposed model would be a downloadable piece of software that can run with minimal input, which would follow the steps of the PTES procedure and produce a report readable by a user of any technical skill. Figure 2.2 shows a table constructed by Stefinko et al. (2016) in [11] comparing the differences between Automated and Manual testing through different stages of the Penetration Testing methodology. One of the key advantages presented in figure 2.2 is the automated reporting and cleanup of the test, which are key features of the Inquisitor Penetration Test.

The main issue businesses face with Penetration testing is cost, Jia et al. (2014) in [8] explains how effective automation aids reducing costs of tests. They state that automatic tests do not need a dedicated tester with associated high price, and that in-house staff can take the responsibility of running the program. Samant (2011) in [9] agrees with this statement, adding that the skill of the tester has been implemented into the program itself, providing a far cheaper method of testing. This also saves significantly on time and company resources, as the automated tests run in the background of a normal workday without interfering with staff and software systems they may be using.

This does not completely negate the need for a professional tester, as automation will greatly boost their productivity and time efficiency since tedious tasks are done for them. Alumbairik & Wills (2016) in [1] and Groš & Kovačević (2020) in [7] both agree that an automated Penetration test allows tester's (and/or Red teamers) to perform more challenging and time consuming tasks like Social Engineering. This effectively doubles the testers efficiency and provides businesses with more testing for their time, aiding in making the high cost seem more worthwhile.

	Manual	Automated
Testing Process	Manual, non-standard process; Labor and capital intensive; High cost of customization;	Fast, standard process; Easily repeatable tests;
Vulnerability/ Attack Database Management	Maintenance of database is manual; Need to rely on public database; Need re-write attack code for functioning across different platforms;	Attack database is maintained and updated; Attack codes are written for a variety of platforms;
Reporting	Requires collecting the data manually;	Reports are automated and customized;
Cleanup	The tester has to manually undo the changes to the system every time vulnerabilities found;	Automated testing products offer clean-up solutions;
Training	Testers need to learn non-standard ways of testing; Training can be customized and is time consuming.	Training for automated tools is easier than manual testing.

Figure 2.2: Comparison of Manual VS Automated Penetration testing

The issue of a cheap immature tester is also resolved with an automated test, as only trusted in-house staff would be exposed to sensitive information. Automatic Penetration Tests are clearly the future of global security, through fast, safe, hands-free tests a business of any size can be confident that their network and customers are protected.

2.4 Projects related to Penetration Testing Toolkit

Several other authors of research papers have attempted to create an automated Penetration testing toolkit, with variations of specific targets like IoT devices or websites. Some of the notable ones that have inspired element of this project have been detailed below.

Net-Nirikshak 1.0 by Shah and Mehtre (2014) in [10] is an excellent paper documenting the creation of an automated Penetration testing tool used on Indian Banking systems. The tool aims to find vulnerabilities in both the software side of the network and the website, using MITRE's CVE database. The tool also cleans up after itself, which is a security concern that most similar tools fail to tackle. After the tool finished the test, a report is collated and emailed off to the tester, all report files and text dumps of information are then deleted off the system. This effectively mitigates the risk of a malicious insider using the report contents to exploit the very sensitive system. The program is also written in Python, allowing for a modular approach with the PTES stages of Penetration Testing. Python also has many libraries such as BeasutifulSoup 4.0 which can be used greatly in the tests, including specific modules related to information security. Overall it is a great outline of

what a Penetration testing tool should be, it is only shame that the tool is not available for public usage.

Almubairik and Wills (2016) in [1] detailed their creation of an algorithm that generates a plan for Penetration tests guided by a threat model. Their algorithm provides a good method for testing through flowcharts following similar stages to the PTES procedure, but acts more as a road map for a professional tester to follow rather than an actual test. This paper is good for basics on an automated test, but lacks the core development of a program.

Jia et al. (2014) in [8] provide a detailed mathematical overview of Penetration testing, again a similar staged methodology has been presented to the one planned for the toolkit. The authors have constructed a useful algorithm for Automated Penetration Tests, including detailed flowcharts of Pseudo code. Yet again this paper is purely an algorithm, lacking the substance of a proper automated program, and the amount of mathematics involved seems like an unnecessary add on.

Benedictis et al. (2018) in [2] presents a detailed methodology for Automated Penetration testing, which contains similar staged testing to this project. The tool is essentially a vulnerability scanner for cloud applications, through reporting of threats, vulnerabilities and exploits affecting the system. Furthermore the authors present an effective solution to Exploitation testing, which seems to be the hardest stage of the toolkit to implement. They do this using OpenVas, a competitor to Nessus (the default choice for this project) and using the results to conduct metasploit exploits.

Samant (2011) in his Masters Thesis [9] developed a similar project to the toolkit, and has detailed documentation on how he constructed the automated Penetration test. This paper contains a great deal of useful information on different methods for attack, focusing more on web based applications. Samant's application took the form of a web-based Penetration test, allowing the user to conduct a series of automated Denial of Service (DOS) attack, with the option to change the protocol used (HTTP, TCP/IP, or SIP). The user would simply enter the IP address and port number before starting the attack of maliciously crafted packets, with the option to stop the test at any time. This project does have a large downside, being the inherent danger and damage it could cause in the wrong hands, or even just left alone whilst running. Exploitation is a tricky grey area in Penetration testing, as a client does not want the tester causing damage to network. This tool if left unsupervised could easily cause serious network outages or damage, making it unsuitable to be used in a fully automated program.

Chapter 3

Methodology

This section of the paper will cover the development and testing of Inquisitor throughout the course of this project. The development of Inquisitor can be split into four distinct sections, with an unofficial fifth for miscellaneous features such as the Start Screen. The first section is the Reporting functionality of Inquisitor, which is used in every aspect of the program. The next two sections are Information Gathering and Vulnerability Scanning, being the two stages of the Penetration Testing Model that this project tackles. Finally there is a section on the emailing features of Inquisitor and post test cleanup.

3.1 Project Planning and Design

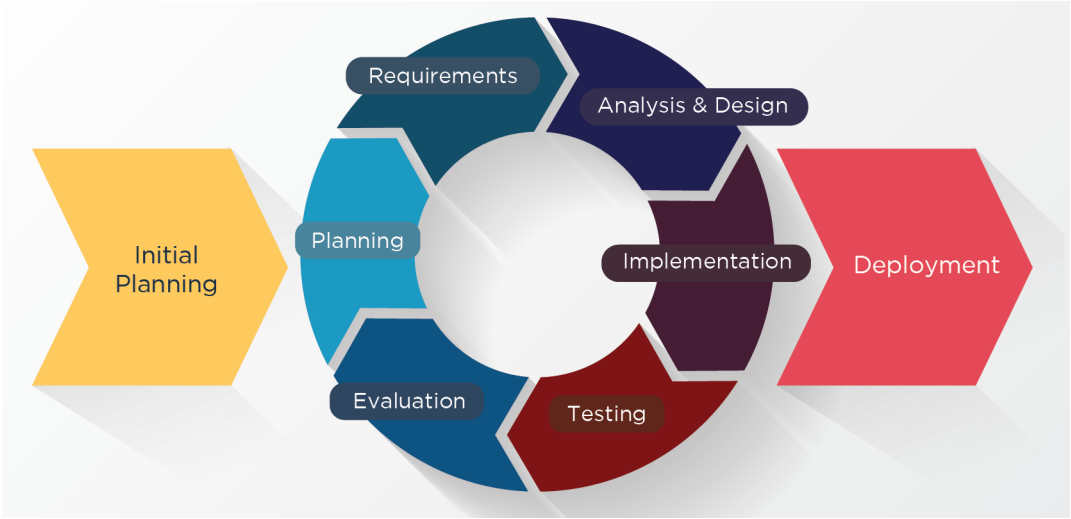


Figure 3.1: Iterative Project Planning (Guthrie-Jensen, 2021)

An Iterative design methodology was implemented in the development of Inquisitor, this was due to modular nature of the tool. Inquisitor was broken down into the aforementioned sections, and broken further into specific tools planned for the project. Time was dedicated to each section through a

Gantt chart, which aided in developing each tool function to full potential, individually adding to the overall program. A risk matrix was also constructed as to mitigate potential risks to development, such as the time estimates for each section being unrealistic, which was mitigated through increased time dedication to each section. A flowchart was also created to give a visual representation of how different sections of Inquisitor interact with each other, which can be seen in figure 3.2. These can all be found in Appendix B.

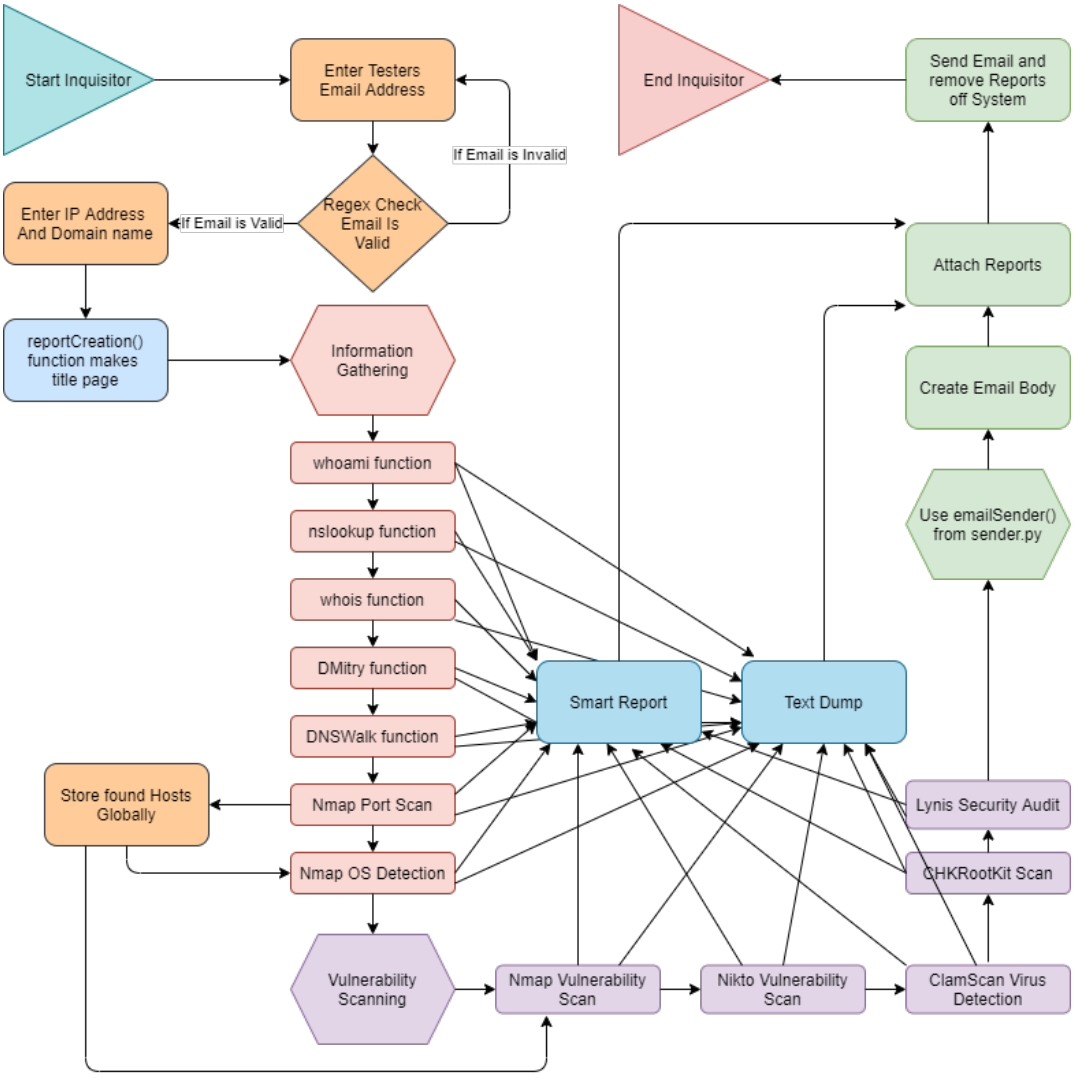


Figure 3.2: Inquisitor design Flowchart

The tool was produced in Python3, which was chosen due to the developers previous experience with the programming language and its wide array of libraries that could be used in the project. The project began development on a Arch Linux dual-booted laptop, but later moved to MX Linux after the old distribution became corrupted. The code was originally written in GNU nano, but

was upgraded to the Python IDLE text editor, which made the code more readable and had better support for indentation. The code was compiled within the Linux terminal with sudo privileges(super user) as to properly test the network. Inquisitor was tested against the developers home network throughout its creation, which proved useful in finding errors as the network was very familiar to the developer.

3.2 Start Screen and User Input

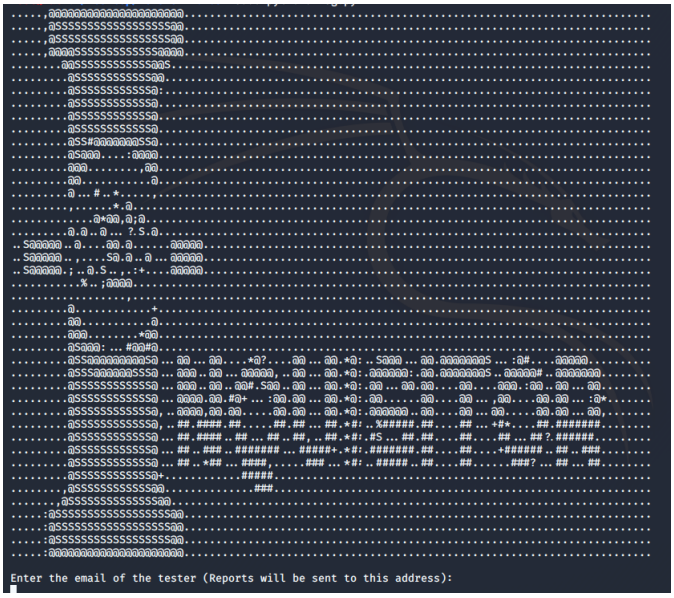


Figure 3.3: Logo Screen for Inquisitor with ASCII art

Every good Linux tool needs some ASCII art to display on launch, so a logo was made using Pixlr's photo editor (Pixlr, 2021) and converted into ASCII using a python program found in Appendix A: Section 3. The code simply makes the logo grey scale and converts the pixels according to colour depth, so white would be represent by a single "." and black would be represented by an "@" symbol. This logo is displayed in the terminal on every launch of Inquisitor, as seen in figure 3.3, which is followed by getting some user input. The main input needed is the tester's email address, to make sure no invalid email is entered Python's RegEx module is used to check the input is in the proper format, example code can be seen in figure 3.4. The user will also enter the IP address (or subnet) and domain name they wish to scan, which will then start the investigation.

```
#Regex to check email is valid
emailChecker = '^(\w|\.|\_|\-)+[@](\w|\_|\-|\.)+[.]\w{2,3}$'
print("Enter the email of the tester (Reports will be sent to this address): ")
validEmail = False
#Checking if the Email address is valid
while validEmail == False:
        reciever = str(input())
        if(re.search(emailChecker, reciever)):
            validEmail = True
        else:
            print("That is not a valid email: ")
```

Figure 3.4: RegEx making sure the email address is valid

3.3 Reporting

Reporting forms the core requirement of the project, as the goal of Inquisitor was always to present its findings in a readable format, but without lacking the technical depth that Security Professionals might find useful. The obvious solution was to split the reports into a Smart report, containing only the useful information, and a text dump of all the results with no missing data. This would allow the tool to be used by staff of any skill level, and still prove useful to expert users.

3.3.1 Smart Report

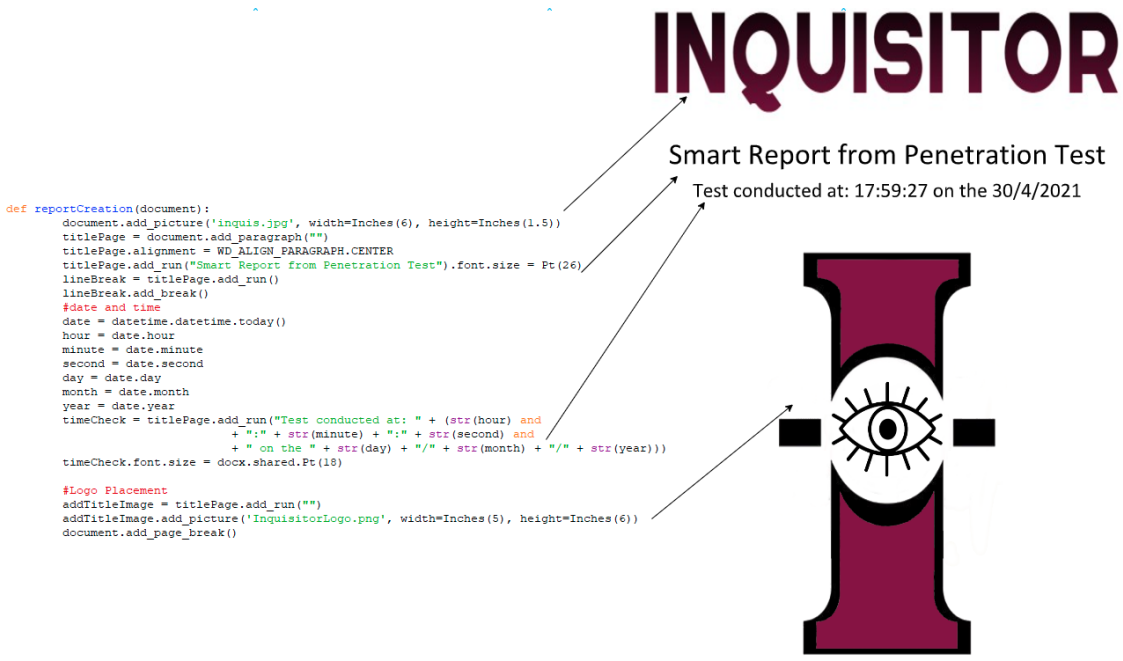


Figure 3.5: Using Python-Docx to create a title page

The Smart report was created using the Python-Docx Library which allows for creation and editing of a Microsoft Word file within a python program. This library allowed Inquisitor to create an

aesthetically pleasing document for the user to read. The first function called in `main()` is the report creation, as every tool will add to the document. This function is an excellent example of what python-docx can achieve, it forms the title page of document, including two images and text. The code also allows for variables to be added, as seen in `timeCheck`, which adds the date and time of test. Figure 3.5 shows the code and result of this function.

The raw results of commands can often be messy with irrelevant information included, so before being added to the report the result needs stripping to only the useful content. An early version of the OS detection function (which will be described in detail in 3.4.7) serves as good example of what content was stripped. Figure 3.6 shows the typical result of an OS detection scan broken down into three categories, being Repeated, Useful, and Irrelevant information. The red rectangle shows all the information already reported by the standard Nmap scan conducted earlier in the test, the Orange rectangle shows the useless contact text found at the end of scan. The green rectangle is all the client will need, so all the information before "Device type" is removed with Python's split operator, and from "OS detection performed" is also removed. Python split is an incredibly versatile yet simple operator, and is used through much of this code to clean up results, it works by splitting data in half at a specific point. The relevant information is saved and added to the smart report, a simple yet highly effective solution.

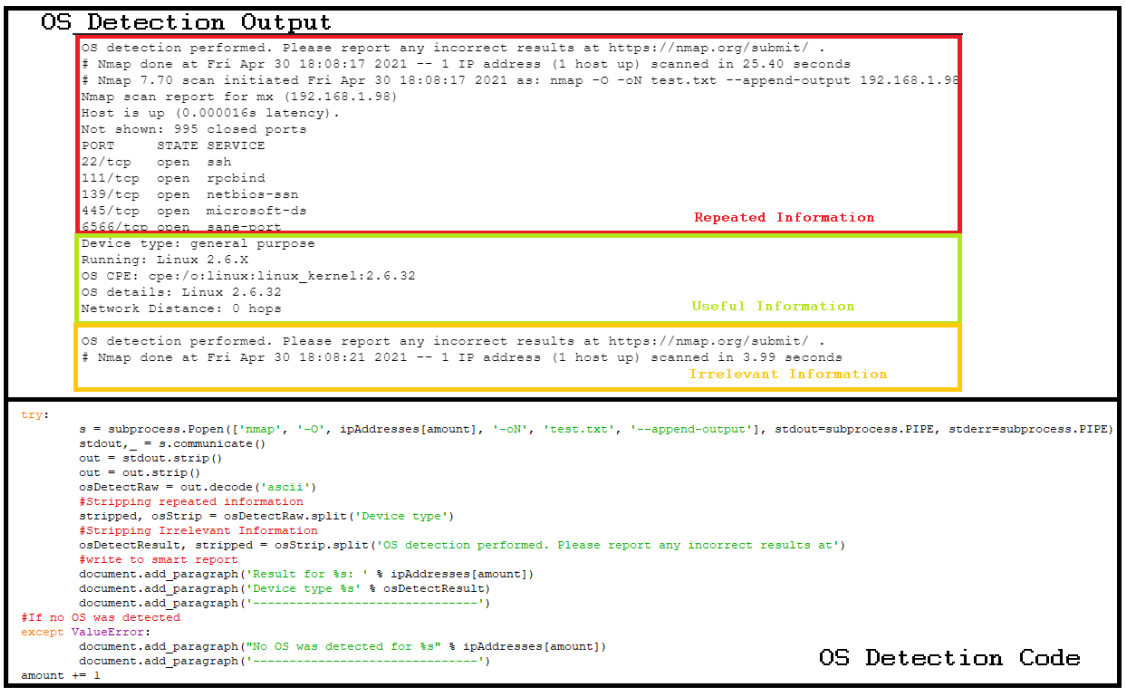


Figure 3.6: Removing the useless information from an OS Detection scan

3.3.2 Text Dump

The text dump file was very easy to implement, as most command line tools allow the results to be saved to a specified file. Even if the tool didn't have that feature the raw result could still be

collected using the saved and decoded value returned by the Subprocess module. The text file would be opened and any result would be pasted in, with a line break to add readability to the text dump.

3.4 Information Gathering

Information gathering is typically the first stage in any Penetration test, involving the collection of data relating to the target system. Development of this stage involved implementation of moderately complex tools, with the goal of mapping the network and finding out simple pieces of information. This sub section will detail each Information Gathering tool added to Inquisitor, full code for each function can be found in Appendix A: Section 1.

3.4.1 whoami

Whoami is a very simple Linux tool used to give the current active username, and was one of the first tools added to Inquisitor. The Subprocess module was used to run the command, using the "Popen" argument, which prevents any code running until the operation is complete. The Subprocess module serves as a more complex and powerful alternative to the OS module, which works for simply jobs but is ill-equipped to deal with some of the more intense scans. The result of the command is saved to a variable using the "communicate()" argument and cleaned up using the "strip()" argument. The result must then be decoded as ASCII before being usable for reporting. Since this command is so simple no splitting is required from the output, so the result is added to both the Smart report and the text dump. Figure 3.7 shows the complete function, with all the code describe above.

```
#whoami function with reporting
def whoamiTool():
        print("Conducting whoami command (1/12)")
        result = subprocess.Popen('whoami', stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        stdout,_ = result.communicate()
        out = stdout.strip()
        out = out.strip()
        whoamiResult = out.decode('ascii')

        #open and add to raw file
        font.size = Pt(12)
        textFile = open("textdump.txt", "a")
        textFile.write("Result for whoami command: " + whoamiResult + "\n")
        textFile.write("------------------------------------------------------ \n")
        #write to smart report
        document.add_heading('Whomai Command', level=1)
        document.add_paragraph('Whoami is a simple linux command that displays the current users name')
        document.add_paragraph('Result for whoami command: %s' % whoamiResult)
```

Figure 3.7: WhoamI function source code

3.4.2 NSLookup

Nslookup is a tool used to query domain names or IP address for information, usually in the form of DNS records. The user inputted domain name from the start of the program is used here, going through similar steps to the whoami function, being communicated and decoded before storing in a variable. From testing Nslookup had only useful information, so the entire variable storing the results was put into each report uncut. Figure 3.8 shows the function, and a example of the result can be found in Appendix C: Section 1.

```
#nslookup function with reporting
def nslookupTool(domainName):
        print("Conducting nslookup command (2/12)")
        s = subprocess.Popen(['nslookup', domainName], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        stdout,_ = s.communicate()
        out = stdout.strip()
        out = out.strip()
        nslookupResult = out.decode('ascii')
        #open and add to raw file
        font.size = Pt(12)
        textFile = open("textdump.txt", "a")
        textFile.write("Result for nslookup command: \n" + nslookupResult + "\n")
        textFile.write("------------------------------------------------------ \n")
        #write to smart report
        document.add_heading('NSLookup Command', level=1)
        document.add_paragraph('NSLookup is tool used to find more information about a domain name or other DNS records')
        document.add_paragraph('Result for nslookup command: \n %s' % nslookupResult)
```

Figure 3.8: Nslookup function source code

3.4.3 WHOIS

```
    def whoisTool(domainName):
            print("Conducting whois command (3/12)")
            strip, strippedName = domainName.split('www.')
            s = subprocess.Popen(['whois', strippedName], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
            stdout,_ = s.communicate()
            out = stdout.strip()
            out = out.strip()
            whoisRaw = out.decode('utf-8')
            try:
                    whoisResult, stripped = whoisRaw.split('>>>', 1)
            except ValueError:
                    whoisResult, stripped = whoisRaw.split('WHOIS lookup made at', 1)

            #open and add to raw file
            font.size = Pt(12)
            textFile = open("textdump.txt", "a")
            textFile.write("Result for whois command: \n" + whoisResult + "\n")
            textFile.write("------------------------------------------------------ \n")
            #write to smart report
            document.add_heading('Whois Command', level=1)
            document.add_paragraph('Whois is a listing system that show records about websites, such as o
            document.add_paragraph('Result for whois command: \n %s' % whoisResult)
```

Figure 3.9: WHOIS function source code

WHOIS is command used to provide further information on a domain name or IP address block. Unlike Nslookup, WHOIS only takes the domain name, without the protocol or sub-domain (i.e. google.com rather than www.google.com) so has to be split accordingly. After the Subprocess has finished the results are saved, but this time decoded as “utf-8”, as the result is much larger than previous tools. The result contains a large section detailing the expiration date and last database update which is irrelevant to the Penetration Test, and so are removed with the split operator. From testing it was found that some websites did not have this irrelevant information and so had to be split at a different location, with a “Try, Except” operator to properly catch the error. The variable is then added to each report, the full function can be seen in figure 3.9, and the chunk of the result placed in the Smart report can be seen in figure 3.10.

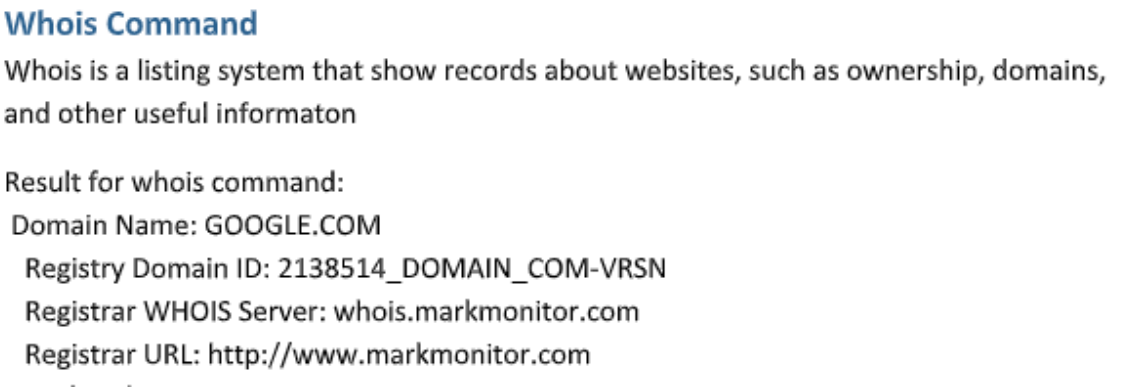


Figure 3.10: WHOIS result place in Smart Report

3.4.4 DMitry scan

DMitry (Deep Magic Information Gathering Tool) is a multi purpose product with the ability to find about a variety of information about a domain. DMitry is used here to find sub-domains, some basic Netcraft information, and possible email addresses attached to the host, this is all done through the "nse" flag in the command. Just like the whois function, the domain name needs to be stripped to work, which is then saved to a variable using a Subprocess and decoded. Unnecessary information is stripped from the result and pasted into both reports, this can all be seen in figure 3.11.

```
def dmitryTool(domainName):
        print("Conducting DMitry scan (4/12)")
        strip, strippedName = domainName.split('www.')
        s = subprocess.Popen(['dmitry', '-nse', strippedName], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        stdout,_ = s.communicate()
        out = stdout.strip()
        out = out.strip()
        dmitryRaw = out.decode('ascii')
        strip1, dmitryRaw2 = dmitryRaw.split('"There be some deep magic going on"')
        dmitryResult, strip2 = dmitryRaw2.split('All scans completed')
        #open and add to raw file
        font.size = Pt(12)
        textFile = open("textdump.txt", "a")
        textFile.write("Result for DMitry command: \n" + dmitryResult + "\n")
        textFile.write("------------------------------------------------------- \n")
        #write to smart report
        document.add_heading('DMitry Command', level=1)
        document.add_paragraph('DMitry (Deepmagic Information Gathering Tool) is a command line application used
        document.add_paragraph('Result for whois command: \n %s' % dmitryResult)
```

Figure 3.11: DMitry code for Inquisitor

3.4.5 DNSWalk scan

DNSWalk is a simple tool used to check whether a domain name is consistent and accurate throughout zone transfers. This tools works in a very similar manner to the whois function, with the result being decoded and added to both reports uncut.

```
def dnswalkTool(domainName):
        print("Conducting DNSWalki command (5/12)")
        strip, strippedName = domainName.split('www.')
        dnsName = strippedName+'.'
        s = subprocess.Popen(['dnswalk', dnsName], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        stdout,_ = s.communicate()
        out = stdout.strip()
        out = out.strip()
        dnswalkRaw = out.decode('ascii')
        #open and add to raw file
        font.size = Pt(12)
        textFile = open("textdump.txt", "a")
        textFile.write("Result for DNSWalk command: \n" + dnswalkRaw + "\n")
        textFile.write("------------------------------------------------------ \n")
        #write to smart report
        document.add_heading('DNSWalk Command', level=1)
        document.add_paragraph('DNSWalk is a Domain Name System (DNS) debugger, checking a domain for
        document.add_paragraph('Result for DNSWalk command: \n %s' % dnswalkRaw)
```

Figure 3.12: DNSWalk function

3.4.6 Nmap Information Scan

Nmap is an incredibly popular network scanner that can discover hosts and services on an IP address or subnet, this achieved through sending packets and analyzing the responses. Nmap is often seen as the backbone of any penetration test, providing a great amount of useful information through a series of features. Fortunately Python has a dedicated library for Nmap, named Python-Nmap, which is superior to the Subprocess method used in other tools, as it allows for easily manipulation of results.

The scan is setup using the "PortScanner()" argument of Nmap, which is then set to appended the raw output to the text dump file. After the scan has finished all of the results get stored in an array, which as seen in figure 3.13 can be iterated through for each host discovered by the scan. All the relevant information from the scan is added to the report, and all the discovered IP addresses are added to a global array to be used in further tests.

```
def nmapScanTool(ipaddr):
        print("Conducting Nmap scan (6/12)")
        font.size = Pt(12)
        document.add_heading('Nmap port scan', level=1)
        document.add_paragraph("Nmap is network scanning tool used in the majority of penetration tests, it can discover hosts
        #Start the Scan
        nmScan = nmap.PortScanner()
        nmScan.scan(ipaddr + ' -oN textdump.txt --append-output')
        #Output to the Smart Report
        for host in nmScan.all_hosts():
                document.add_paragraph('Host : %s (%s)' % (host, nmScan[host].hostname()))
                ipAddresses.append(host)
                document.add_paragraph('State : %s' % nmScan[host].state())
                for proto in nmScan[host].all_protocols():
                        document.add_paragraph('----------')
                        document.add_paragraph('Protocol : %s' % proto)
                        lport = nmScan[host][proto].keys()
                        for port in lport:
                                document.add_paragraph('port : %s\tstate : %s' % (port, nmScan[host][proto][port]['state']))
                document.add_paragraph('-------------------------------')
```

Figure 3.13: Nmap port scanning code with Reporting

3.4.7 Nmap OS Detection

Nmap offers far more than just port scanning, one of its best features is the Operating System Detection flag, which can accurately guess the OS of the target system. This however is not supported in the Python-Nmap library, so a Subprocess command is used instead. Nmap is also used for Vulnerability scanning, to avoid running a third scan the required flags have been added to this command, being "-script vuln".

The entire scanning aspect is found in a "Try, Except" error handling block to avoid Inquisitor breaking at an error, as for some devices(like Routers or games consoles) do not have a detectable OS. If no Operating system is detected then that information is simply added to the Smart Report instead. The Subprocess here actually uses the discovered hosts from the previous array, scanning each IP address one by one and adding the stripped result to the Smart report. The "Append-Output" flag is also set in the command, placing the raw results within the text dump file.

For vulnerability scanning the code was also placed in an error handling block, as not all hosts are vulnerable. Each vulnerable host's results are stripped appropriately as to provide a variable containing the vulnerability results and the OS result. The vulnerability result is added to a global array to accessed later, as well as the vulnerable IP address. The function can be seen in figure 3.14 and the full code can be found in Appendix A: Section 1.

```
def osDetectTool():
        print("Conducting OS detection (7/12)")
        amount = 0
        document.add_heading('OS Detection', level=1)
        document.add_paragraph('Nmap also provides an excellent OS Detection scan, using the result collected previously the entire networks
        #Conducting an OS detection on each IP address found
        while amount < len(ipAddresses):
                try:
                        s = subprocess.Popen(['nmap', '-script', 'vuln','-O', ipAddresses[amount], '-oN', 'textdump.txt', '--append-output']
                        stdout,_ = s.communicate()
                        out = stdout.strip()
                        out = out.strip()
                        osDetectRaw = out.decode('ascii')
                        #Stripping repeated information
                        stripped, osStrip = osDetectRaw.split('Device type')
                        try:
                                osResult, vulns = osStrip.split('Host script results:')
                                vulnResult, stripped = osStrip.split('OS detection performed. Please report any incorrect results at')
                                vulnerability.append(vulnResult)
                                vulnerableIP.append(amount)

                        except ValueError:
                                #Stripping Irrelevant Information
                                osResult, stripped = osStrip.split('OS detection performed. Please report any incorrect results at')

                        #write to smart report
                        document.add_paragraph('Result for %s: ' % ipAddresses[amount])
                        document.add_paragraph('Device type %s' % osResult)
                        document.add_paragraph('--------------------------------')
                #If no OS was detected
                except ValueError:
                        document.add_paragraph("No OS was detected for %s" % ipAddresses[amount])
                        document.add_paragraph('--------------------------------')
                amount += 1
```

Figure 3.14: Nmap OS Detection Code

3.5 Vulnerability Scanning

Vulnerability Scanning is the core takeaway from the smart report, as it displays all the potential weak points a malicious hacker could exploit. A series of tools, such as Nmap, Nikto, and Lynis have been implemented to provide a wide variety of results, effectively auditing the entire system. This section will detail each Vulnerability scanner added to Inquisitor, full code can be found in Appendix A: Section 1.

3.5.1 Nmap Vulnerability Scan

Nmap provides a useful vulnerability scanner that has the added benefit of being able to scan remote hosts. As mentioned in section 3.4.7, the actual scanning and cleaning of results for Nmap's vulnerability scan is conducted within the OS detection function. This leaves the Nmap Vulnerability scanning function only acting as a place to paste the cleaned results into the smart report.

3.5.2 Nikto Vulnerability Scan

Nikto is an excellent vulnerability scanner used to find flaws in webservers and domains, checking for dangerous files, outdated services, and vulnerabilities like cross-site-scripting. The user submitted domain name is added to the command, which begins the scan, the entire results were deemed useful so nothing was stripped from the smart report. A snippet of the code can be seen in figure 3.15.

```
def niktoTool(domainName):
        print("Conducting Nikto Vulnerability Scanning (9/12)")
        s = subprocess.Popen(['nikto', '-h', domainName], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        stdout,_ = s.communicate()
        out = stdout.strip()
        out = out.strip()
        niktoResult = out.decode('ascii')
```

Figure 3.15: Nikto Vulnerability Scanner Subprocess Code

3.5.3 ClamScan Virus Detection

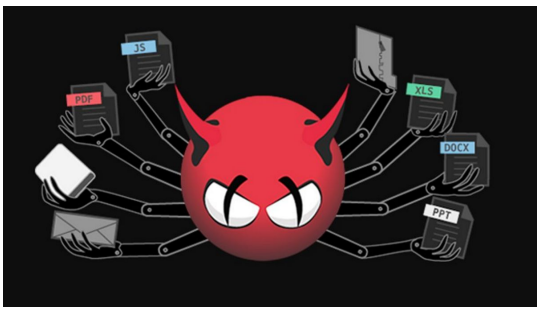


Figure 3.16: ClamScan Logo (ClamAV, 2021)

ClamScan is Linux based antivirus command line tool, that scans the system against a library of known viruses. This tool provides a thorough scan for dangerous files, and is a common countermeasure in any system administrators arsenal. ClamScan was included within Inquisitor as it greatly helps in locating any potential viruses already on the host system, further aiding clients in protecting their network. The tool is run through a Subprocess, once finished the raw result is stripped to just the summary and added to the report. A result from ClamScan can be seen in 3.17.

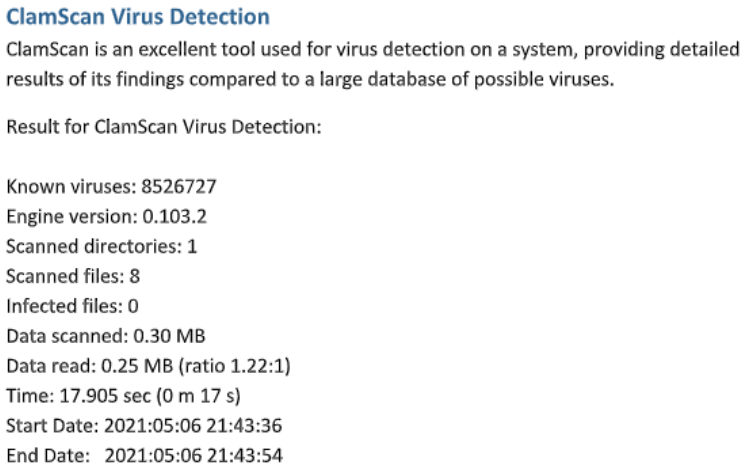


Figure 3.17: ClamScan Virus Detection Result in Smart Report

3.5.4 ChkRootKit Root Kit Detection Scan

RootKits are one of the most dangerous forms of malware facing the digital age, as they give the malicious hacker full control of your system without the owner even realising. ChkRootKit is the perfect tool for uncovering these hidden vulnerabilities, performing a deep scan of the system in order to find any rootkits. ChkRootKit was added to Inquisitor for similar reasons to ClamScan, as it gives a definitive answer on whether the host system is infected with a deadly rootkit. The command returns a long list of checked areas, with a "Warning" label on any found rootkits. This result is hard to work with as it has no clean summary, only a long continuous log of the scan. To solve this a "Try, Except" operator, which can be seen in figure 3.18, was used to check for any warnings by splitting at correct area and pasting the result into the reports. If no rootkits were found then instead of getting an error the code will simply paste a messaging saying no results were found into either reports.

```
try:
    stripped, rootResult = rootRaw.split('Warning:')
    #open and add to raw file
    font.size = Pt(12)
    textFile = open("textdump.txt", "a")
    textFile.write("Result for ChkRootKit Root Kit Detection Scan: \n" + rootResult + "\n")
    textFile.write("------------------------------------------------------ \n")
    document.add_paragraph('Result for ChkRootKit Root Kit Detection: \n %s' % rootResult)
except ValueError:
    #open and add to raw file
    font.size = Pt(12)
    textFile = open("textdump.txt", "a")
    textFile.write('ChkRootKit returned no Warning, meaning no Root Kits were found on your system.' + "\n")
    textFile.write("------------------------------------------------------ \n")
    document.add_paragraph('ChkRootKit returned no Warning, meaning no Root Kits were found on your system.')
```

Figure 3.18: ChkRootKit Root Kit Detection Scan Code

3.5.5 Lynis Security Audit

Lynis is an amazing command line tool that provides a full audit of the system, with detailed instructions on how to fix issue. This is an invaluable tool to any user, and was a must have for Inquisitor, the tool even have a Penetration Testing flag, which was selected for this project. Lynis produces a long output, with only the summary being of interest, however due to the way Lynis formats results makes them inherently incompatible with the smart report. This is due to a lot of special character usage to format indentation in the results, which does not process with Python-Docx. The replace operator of Python was used to remove each special character found in the report, which can be seen in figure 3.19, which then allows it to be added cleanly to the smart report.

```
def lynisTool():
        print("Conducting Lynis security audit (12/12)")
        s = subprocess.Popen(['lynis', '--pentest'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        stdout,_ = s.communicate()
        out = stdout.strip()
        out = out.strip()
        lynisRaw = out.decode('ascii')
        strip1 ,lynisRaw2 = lynisRaw.split('Lynis 2.6.2 Results')
        lynisResult, strip2 = lynisRaw2.split('Follow-up')
        lynisCleanResult = lynisResult.replace('[0m' , '').replace('[1;31m' , '') and
        .replace('[1;37m' , '').replace('[0;36m' , '').replace('[1;33m' , '').replace('[0' , '')
        #open and add to raw file
        textFile = open("textdump.txt", "a")
        textFile.write("Result for Lynis Security Audit: \n" + lynisCleanResult + "\n")
        textFile.write("------------------------------------------------------ \n")
        #write to smart report
        document.add_heading('Lynis Security Audit', level=1)
        document.add_paragraph('Lynis is a security auditing tool that provides a detailed responce on a systems w
        document.add_paragraph('Result for Lynis Security Audit: \n -[ Lynis 2.6.2 Results %s' % lynisCleanResult)
```

Figure 3.19: Lynis Security Audit Code in Inquisitor

3.6 Emailing and Cleanup

One of the key requirements set out in the Proposal was an effective clean up system after the tests had completed. From research an Emailing solution seemed like the best fit for the task, as it allowed just the tester access to the sensitive information and had a built in library for Python. The "smtplib" library was used, along with the email and MIMEBase imports, to fully construct the email and add the reports. The emailing function was separated from the rest of the source code to improve readability, as it was large and irrelevant to the main function of Inquisitor. The File was named "sender.py" with the function being named "emailSender", which requires only the testers email address as an input. The full code can be found in Appendix A, Section 2, whilst figure 3.20 highlights some of the important aspects. A Gmail account was setup named "smartpentestreport@gmail.com", to allow Python to send emails through it the setting "Allow Access Through Insecure Applications" had to be disabled.

```
#HTML Message
html = """\
<html>
 <body>
  <div style="text-align:center;">
   <img src="https://drive.google.com/uc?export=view&id=1jpl4bItstEtP-vcoL-yDQ1dy969bY2je" title="Inquisitor" width="600" height="400" />
   <h3>Greetings</h3><br>
   <p>Please find attached the full report of my findings from the Penetration Test  and a text file containing the raw data from scans.<br>
   Thank you for using Inquisitor.
   </p>
  </div>
 </body>
</html>
"""
htmlpart = MIMEText(html, "html")
message.attach(htmlpart)
        #Attaching Reports
filename = ['report.docx', 'test.txt']
for filename in filename or []:
        with open(filename, "rb") as attachment:
                part =MIMEBase("application", "octet-stream")
                part.set_payload(attachment.read())
                encoders.encode_base64(part)
                part.add_header(
                        "Content-Disposition",
                        'attachment; filename= "%s"' % os.path.basename(filename))
                message.attach(part)

#Adding the Attachement
text = message.as_string()
```

Figure 3.20: HTML message and attachments of reports in sender.py

Standard Python emailing can look a bit bland, but it does support HTML code, allowing for much nicer looking emails (Langen, 2019). The email sent from Inquisitor is rather short and sweet, with some instructional text and a nice logo, that can be seen in figure 3.21, looking both professional and aesthetically pleasing. The image included in the email is hosted through Google Drive, as Gmail does not trust embedded images from any other hosting service. The link had to be edited slightly to make it accessible to Inquisitor, as Google drive images are private by nature (Kamil, 2019). Both the Smart Report and text dump are then encoded in Base64 before being attached to the email using a for loop (making it expandable for more attachments), and sent to the receiver. This is obviously the last function to be called in the program, so after it has finished two OS remove calls(from the OS system Python library) are made to remove the files of the hard drive. This effectively cleans up the system and provides the tester with the reports, and concludes Inquisitor's investigation.

Greetings

Please find attached the full report of my findings from the Penetration Test and a text file containing the raw data from scans.
Thank you for using Inquisitor.

Figure 3.21: Received Email from Inquisitor

Chapter 4

Results

This chapter will cover what tests were performed on the Inquisitor program to properly evaluate its potential as a Vulnerability Assessing Penetration Testing Toolkit. A detailed test plan is set out to find how Effective, Efficient and how fast Inquisitor is compared to similar tools already available. Graphs and diagrams are formed from the raw results to better visualise the differences between the three tools in Chapter 5: Discussion.

4.1 Test Plan

As set out in the Research Questions, Inquisitor was going to be tested to see how effective, efficient, and how fast it was compared to similar tools. To elaborate on that the three testing variables are as follows:

Speed: How long did the toolkit take to run from start to finish, compared through a series of tests with the average as the final result.

Efficiency: How much information was uncovered during the testing time, comparing vulnerabilities and other data against the time taken to perform.

Effectiveness: How effectively does the toolkit present the results found to the user. This testing variable is more subjective than the others, but still an important factor, as Inquisitor had a focus on smart reporting.

To test Inquisitor two different toolkits of similar nature were selected, each being a free GitHub program that performed a form of automated Penetration Testing. Commercial products, such the Intruder Vulnerability scanner, were avoided, as they are much larger in scope compared to Inquisitor leading to unbalanced testing. These commercial toolkits were also found to be either expensive, or requiring a credit card to access the free trial (also requiring Company information to create an account). The free toolkits found were to have similar scopes as Inquisitor, and so were selected for testing.

The IP address “192.168.1.98”, the MX Linux laptop, was selected as a primary test for each solution, as it was known to be vulnerable and it’s operating system was able to be detected by software. Subnets were not scanned as some toolkits did not provide support for that kind of scanning. If the toolkit offered a website scanning feature then “DVWA.co.uk” was entered, since its a deliberately vulnerable site that allows testing attempts (DVWA, 2021). A secondary round of testing was conducted on the solutions, this time using the “Metasploitable 2” virtual machine, which is also intentionally vulnerable and is used to improve Penetration Tester’s skills (Rapid7, 2021).

4.2 Kaboom automated Penetration Test

Kaboom (GitHub, 2020) is an automated Penetration Test, that performs a variety of Information Gathering and Vulnerability Scanning on a target IP address. This solution served as inspiration for Inquisitor during the proposal of this project and has several tools in common, such as Nmap and Nikto. Kaboom is accessed through a terminal, accepting all of the user inputs before the main operations start, as seen in figure 4.1. The option "iv" was selected so that Information Gathering and Vulnerability Scanning are conducted. The test then runs, outputting minimal results to the terminal, whilst saving the raw information to a hierarchy of files. The tool offers no website scanning, so this was marked as "Not Applicable" for the results.

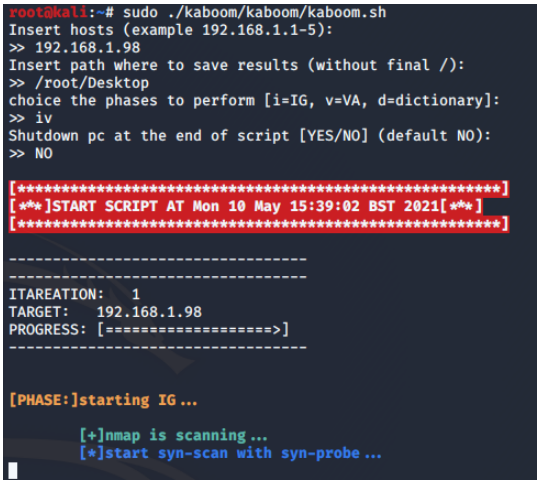


Figure 4.1: Kaboom startup for testing on 192.168.1.98

4.3 Yuki Chan The Auto Pentest

Figure 4.2: Yuki Chan Start Screen and User Input

Yuki Chan (GitHub, 2017) is an automated Penetration test conducted on a domain name, performing a large variety of scans on the target in order to properly assess its security. It shares many similarities to Inquisitor, such as the Nikto Vulnerability scan, but lacks the scanning of IP addresses. Yuki Chan is accessed through terminal, only taking the domain name as an input, before starting the scans. The output is displayed on the terminal itself with no option to save as a file, showing the full raw results to tester. The tool however is designed well and looks very presentable, with ASCII art just like Inquisitor, as seen in figure 4.2.

4.4 Primary Testing Results

These are the results for testing the toolkits versus the testing IP address (192.168.1.98, MX Linux Machine) and the domain "DVWA.co.uk". These results measure the time and vulnerabilities found on the target, being the **Speed** and **Efficiency** from the test plan, **Effectiveness** will be discussed in Chapter 5: Discussion as it is more subjective.

TEST ON 192.168.1.98	TIME (minutes)	VULNS found host	VULNS found site
Kaboom 1	3.21	0	N/A
Kaboom 2	3.25	0	N/A
Kaboom 3	3.16	0	N/A
Kaboom 4	3.22	0	N/A
Kaboom 5	3.01	0	N/A
Inquisitor 1	17.88	3	19
Inquisitor 2	10.43	3	19
Inquisitor 3	15.37	3	19
Inquisitor 4	13.48	3	19
Inquisitor 5	16.02	3	19
Yuki 1	21.53	N/A	39
Yuki 2	21.45	N/A	39
Yuki 3	22.08	N/A	39
Yuki 4	22.01	N/A	39
Yuki 5	21.55	N/A	39

Figure 4.3: Results for testing IP address

4.5 Secondary Testing Results

These are the results for testing the toolkits versus the Metasploitable IP address (192.168.175.130) and the domain "DVWA.co.uk". Yuki Chan is ineligible for this test as Metasploitable does not have a domain to test against. Yet again these results measure the time and vulnerabilities found on the target, being the **Speed** and **Efficiency** from the test plan, with **Effectiveness** being discussed in Chapter 5: Discussion.

TEST ON Metasploitable	TIME (minutes)	VULNS found host	VULNS found site
Kaboom 1	15.23	0	N/A
Kaboom 2	15.01	0	N/A
Kaboom 3	14.71	0	N/A
Kaboom 4	15.14	0	N/A
Kaboom 5	15.01	0	N/A
Inquisitor 1	14.36	9	19
Inquisitor 2	15.22	9	19
Inquisitor 3	15.24	9	19
Inquisitor 4	14.36	9	19
Inquisitor 5	14.56	9	19

Figure 4.4: Results for Metasploitable 2

Chapter 5

Discussion

This chapter will thoroughly discuss Inquisitor, detailing how the design methodology affected the program, how well development ran and any issues faced, how well the program perform against its peers, and a final evaluation on how well the project met the aim and research questions.

5.1 Design Discussion

Iterative Project Planning was the design methodology selected for Inquisitor, and was found to be the perfect fit for a project of this nature. A toolkit is essentially just a series of modules contributing to solve the problem at hand, in Inquisitor's case each module was a function containing a tool. Throughout development tools were added individually after testing and evaluation had concluded for each module, and were relatively independent from the main code. This meant that they didn't need to be edited later after more modules were added. Iterative Project Planning allowed for smooth development and easy deployment of modules, helping Inquisitor reach its full potential.

Frequency/ Consequence	1-Very Unlikely	2-Remote	3-Occasional	4-Probable	5-Frequent
4-Catastrophic		R4			
3-Critical		R6	R3		
2-Major		R2		R1	
1-Significant			R5		
1-Minor					

Figure 5.1: Risk Matrix

The construction of the risk matrix also aided in keeping the project on track. Figure 5.1 shows the matrix and a full explanation can be found in Appendix B: Section 2. The countermeasures setup mitigated many of the risks that could have severely delayed project development, a notable example being **R6: Hardware/Software Failure**. This predicted risk occurred in late March

when the Arch Linux distribution became corrupted and unusable. Inquisitor began its life on Arch Linux and was the place were the majority of development occurred, however, frequent backups of source code were made and a Kali Linux Virtual machine was used whilst a new distribution was sorted out, preventing any interruptions to development.

This was not the case for all risks, **R1: Unrealistic Time Estimate** was mitigated through giving each section of Inquisitor a lengthy development time with sufficient buffer (as seen in the Gantt chart in Appendix B: Section 1). What was not planned for were external assignments running parallel to Inquisitor's development, this was particularly noticeable with the Exploitation Testing section where developer attention was constantly switching between tasks. This issue will be discussed further in section 5.2.3 of this chapter.

It is import to note that the entire course of this project was planned, developed and finished during the Covid-19 pandemic. This caused several limitations to Inquisitor as no external body could aid in debugging the code or be used for testing metrics, and all meetings relating to the project had to be conducted virtually. However, this issue was slightly mitigated in the Risk matrix (**R3: Act of God**), a stable working environment was maintained throughout the entirety of development. Covid-19, although a terrible virus, did aid somewhat in developing Inquisitor, as without a formal workplace more flexible or unusual working hours could be implemented in order to fit with the developer's preferred schedule.

5.2 Development Discussion

Development of Inquisitor took place over four months, which included large portions of research followed by segmented development of modules. The original plan for development can be seen in Appendix B: Section 1, which shows the planned Gantt chart and the later updated chart to better fit the project timescale.

5.2.1 Information Gathering

Information Gathering was the first section of Inquisitor to be worked on, acting as example code to prove the feasibility of this project. It was also understood to be the simplest section to work on, as most tools provide a clean and small result. The Information Gathering section of Inquisitor encompasses seven different tools, each adding to the report in their own way.

The only issue that arose from this section was after the implementation of the Nmap Python Library, which conducted a port scan on the target. After testing and evaluation of the module it was added to the main source code, which immediately caused issue. No errors were reported to terminal, but the program would run endlessly without sending the report. After some time debugging it was found that Python's Nmap Library automatically creates an array named "port" after scanning, however, a variable by the same name already existed in the emailing function, acting as the port to send the report from. What must have resulted was the emailing module attempting to send the report through each port number in the array, causing Inquisitor to take hours to process. This issue took a significant time to be found and fixed, leading to slight development delays, mostly due to the lack of an error message.

5.2.2 Vulnerability Scanning

Vulnerability scanning was planned to utilise four different tools, but five were added in the end with several changes to what Inquisitor would consist of. The majority of Vulnerability scanning

ran smoothly, adding excellent tools like Nikto and a Nmap Vulnerability scan built in to previous scans to save on running time of Inquisitor.

OpenVAS is a web-based vulnerability scanner and manager, providing a deep scan of a host and outputting the results into a PDF file. These results are laid out very well, with aesthetically pleasing graphs to better display information. OpenVAS also offers a command line service that can be interacted with Python in order to automate scans. A great deal of effort went into trying to get this software implemented into Inquisitor but no workable solution was ever found. The Python module seemed disjointed, and required input into the OpenVAS website just to get the tests working. Furthermore every scan required the users password to be entered, which meant additional setup of a OpenVAS account was needed before Inquisitor could run. This with the fact that the installation was complex lead to OpenVAS being deemed as an unfeasible addition. Inquisitor is aimed at small businesses who don't have a dedicated security specialist, a novice tech user would have great difficulty at installing this software correctly, so Inquisitor is better off without it.

Nessus was another web-based security scanner considered for this project, offering the same features as OpenVAS. Unfortunately the Python Library proved just as difficult to use, and rarely worked properly when run, often breaking the running code. The results generated were also stored in a PDF file like OpenVAS, which would be extremely arduous to strip and add to the Smart report that all the other tools were using. The same problems before were present in Nessus, and so was dropped from Inquisitor in favor of more reliable tools that provide more varied results.

5.2.3 Exploitation Testing

Since the proposal of this project Exploitation testing has always been seen as an addition to Inquisitor rather than a set in stone requirement. The act of Exploitation requires dynamic responses to vulnerabilities and often needs a lot of user input. This meant that automation would be very difficult to implement. It was also slightly outside the scope of this project, as a small business running Inquisitor would not want their system actually being exploited, which could lead to leakage of sensitive information in the report. Still, Exploitation testing can prove useful at displaying the severity of vulnerabilities, so attempts were made to implement some safer automated exploits. None of these attempts were successful, and as mentioned earlier external projects interrupted the development phase, leading to no Exploitation Testing being added to Inquisitor.

OWASP zap, an open-source web application security scanner, was attempted to be implemented into Inquisitor as a spidering tool, finding all the hidden URLs and databases. OWASP zap also had a Python module created for it, which would make designing the code a lot more readable, with results that could easily be manipulated for the smart report. However, this tool never functioned properly, displaying errors with no available fixes throughout development and without ever running properly. It was a shame that this tool could not be added as it could properly display the vulnerabilities in action without compromising the website, but as it never worked correctly it had to be dropped.

Nessus was also considered for Exploitation testing, as it offers a similar method of attack to OWASP zap but with more focus on the exploitation of the vulnerabilities found in the actual Nessus results. There was even a Python Library, named "nessrest", that could be used to automate the service. As mentioned earlier, Nessus was never able to run correctly through Inquisitor, and the same applied to the Python Library, giving many errors with few methods to fix them. Due to this and all the issues that arose from the Vulnerability Scanning Nessus attempt, it was deemed unfeasible for the final product.

The original plan for Exploitation Testing was to perform an SQL injection attack against any login forms, with the potential of cloning the database and cracking the passwords or sensitive information. This idea came from several papers, which can be found in Chapter 2: Literature Review, which mentioned it as a possible attack method for automated tests. The Python module used for this attack, named "Mechanize", was found to be incompatible with Python 3, which the entirety of Inquisitor was developed in. It was not worth it to rewrite Inquisitor in an older version Python, which would also cause several tools to become incompatible, just to perform an exploit, so it was dropped from the program.

A last ditch form of attack was considered to be added to Inquisitor, being a Distributed Denial of Service (DDoS) to the target website. Samant (2011) created an automated Penetration Test using this method, performing several DDoS attacks against a target. This seemed far too dangerous to automate into Inquisitor, as it could cause serious outages to systems and potentially cause serious damage. Automating Exploits is also generally discouraged, as in the wrong hands a hacker could use it for bad intentions, which goes against everything Inquisitor stands for. This exploit would not have even given any usable results for either reports, making it a waste of time. The target demographic does not need their website or services being overwhelmed with packets just for a Penetration test, so it was not developed.

5.2.4 Reporting and Emailing

Reporting was the main aim of Inquisitor, and luckily was developed without any issues. The Python-Docx library proved to be incredibly versatile and relatively simple to use. This was aided by the extensive online documentation for the library, and also external literature such as Sweigart's book "Automate the Boring Stuff with Python"(2015) which had a dedicated chapter to this topic.

Emailing was similar, as the inbuilt Python module was very well equipped to send attractive looking emails without making it too complex. The only issues encountered were the port number issue, described earlier and Gmails trusted image problem that was resolved quickly.

5.3 Results Discussion

To properly evaluate Inquisitor's capabilities it was tested against two toolkits of a very similar nature, measuring the time taken to test, the vulnerabilities found, and how well the toolkit reports back to the user. These raw results were taken and made into graphs to better visualise the differences between solutions.

5.3.1 Vulnerabilities found

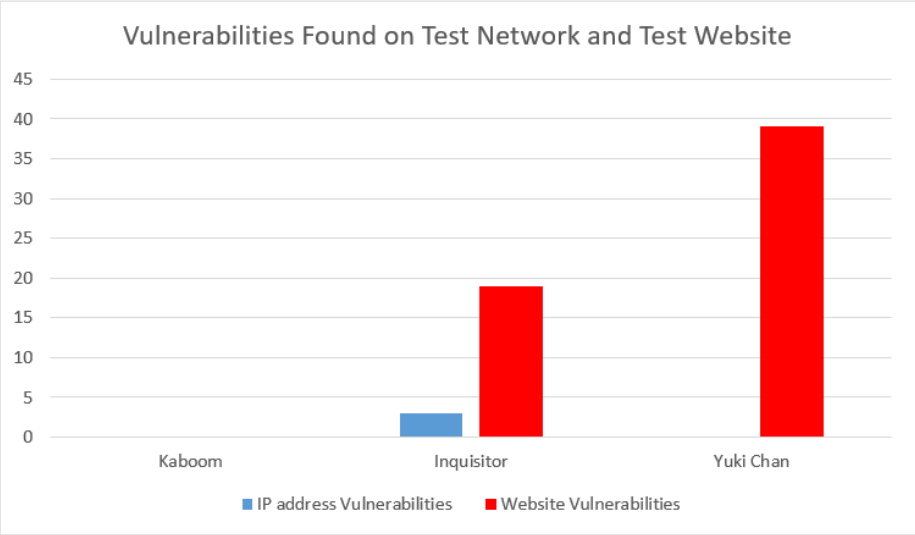


Figure 5.2: Graph comparing Vulnerabilities found between toolkits against testing network

Vulnerabilities should be the key takeaway from any Penetration Testing Toolkit, as they display to the client all the areas of weakness that a malicious Hacker could exploit. Vulnerabilities found came under the "**Efficiency**" testing metric. Each toolkit was tested five times against the testing network and site. There was zero difference between vulnerabilities found in each test, showing that each toolkit was consistently efficient. Figure 5.2 shows a bar graph comparing the vulnerabilities found against each target. Kaboom massively under performed, failing to find a single vulnerability. Inquisitor proved to be very efficient in finding issues in both targets, but failed to meet Yuki Chan's staggering result for website vulnerabilities. From analysis it is clear that Yuki Chan has a greater number of scanners to use against sites, finding issues that Inquisitor was blind to, mainly being SQL logins that were vulnerable to Injection.

Tests were also conducted against the deliberately vulnerable Metasploitable 2 virtual machine, Yuki Chan was not tested as it did not have host scanning features. The results were similar to the previous results, with Inquisitor finding many vulnerabilities and Kaboom finding nothing, which can be seen in figure 5.3. From both tests it's clear that Yuki Chan was able to find the most vulnerabilities, however Inquisitor found a considerable number of vulnerabilities over both targets, which is impossible on the Yuki toolkit. Host based vulnerabilities are often more dangerous than domain based, as they have the potential to give the Hacker root access to the entire system, including any website data.

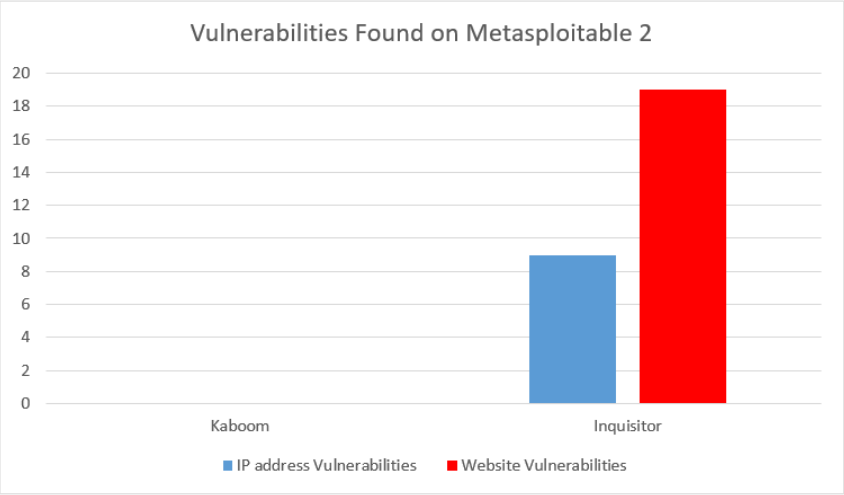


Figure 5.3: Graph comparing Vulnerabilities found between toolkits Metasploitable 2

5.3.2 Speed of the Test

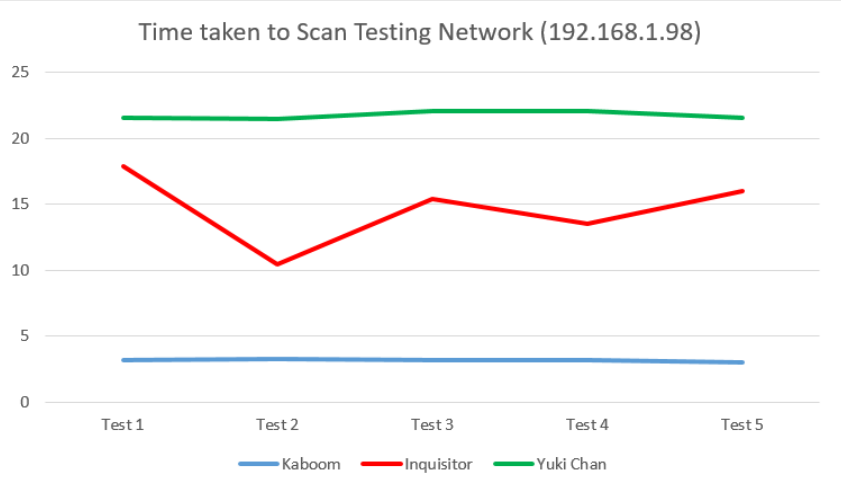


Figure 5.4: Graph comparing time between toolkits against testing network

The "**Speed**" metric was measured throughout each test, and plotted on two different line graphs, as seen in figure 5.4 for the testing network and figure 5.5 for the Metasploitable 2 virtual machine. Kaboom and Inquisitor both had inbuilt timing features which aided in gathering this result, whilst an external timer was used for Yuki Chan. The two external toolkits both had incredibly consistent Speed results, with Kaboom having a Standard Deviation of 0.09 and Yuki Chan having 0.27. Inquisitor's results were much more varied, with a Standard Deviation of 2.5 between times, which can be seen as a negative when it comes to usage, however the deviation is not large enough to cause

any problems.

The Metasploitable results proved much more consistent for Inquisitor, having a Deviation of only 0.4 compared to Kaboom's 0.18. Interestingly, Kaboom took around five times longer to scan the vulnerable virtual machine, but still produced no results.

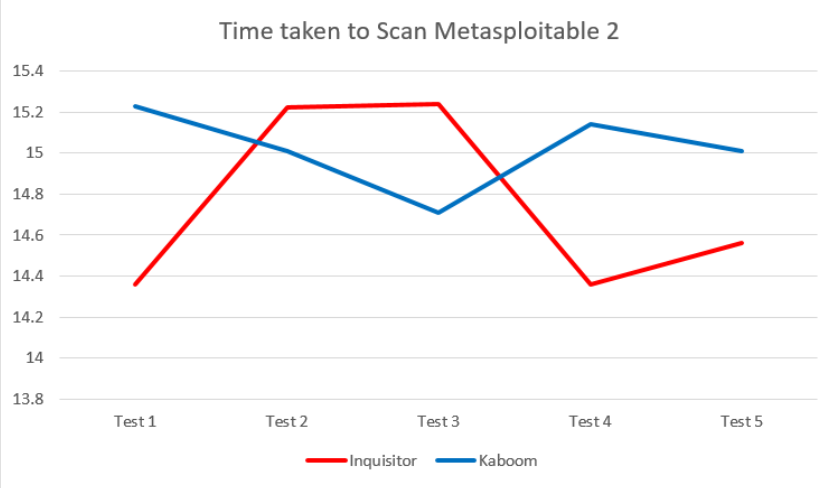


Figure 5.5: Graph comparing time between toolkits against Metasploitable 2

Comparing speeds is not the best testing metric, as Kaboom would have easily been the best solution whilst returning nothing of value to the client. To properly evaluate the toolkits the Vulnerabilities found per minute were calculated to show the most efficient solution. Figure 5.6 shows the final results, with Inquisitor having an impressive 1.37 vulnerabilities found per minute, compared to Kaboom's disappointing result of zero. Yuki Chan does have a slightly higher value of 1.81, but considering that the scope is limited to just websites makes the result less impressive. The result for Inquisitor is still good considering only the Nmap and Nikto vulnerabilities were used for testing without the use of the Lynis tool or root kit checker.

TEST ON 192.168.1.98	Total Vulns found	Median time	Vulns per Minute
Kaboom	0	3.22	0
Inquisitor	21	15.37	1.37
Yuki Chan	39	21.55	1.81
TEST ON Metasploitable	Total Vulns found	Median time	Vulns per Minute
Kaboom	0	15.01	0
Inquisitor	28	14.56	1.92

Figure 5.6: Finding Vulnerabilities found per Minute for each toolkit

5.3.3 Effectiveness of the Report

As mentioned before, Reporting was the main focus of Inquisitor, so the "**Effectiveness**" testing metric is by far the most important in evaluating the success of the project. A report should effectively display the issues needing to addressed, a messy raw report will be of no benefit to a client. Each toolkit reported in a different way, Kaboom provided a hierarchy of files containing raw results, Yuki Chan displayed everything on the terminal itself, and Inquisitor provided two reports through email. Simply comparing these methods of communication reveals a lot about the security of the toolkits. Kaboom's file system does separate each tool into an individual text file, but this does run the risk of a malicious insider using the results to cause damage. Inquisitor's emailing feature completely prevents this, by securely sending the reports to only the tester before deleting them off the system entirely. Yuki Chan's method of sending result to terminal (often raw without editing) left a lot to be desired, as the tester had to scroll through much data just to find useful information. Some terminal application have length limits for a single command, meaning that Yuki Chan cuts off the majority of the first scans, that and the fact that it had no way to output the results leave this method as being unsatisfactory .

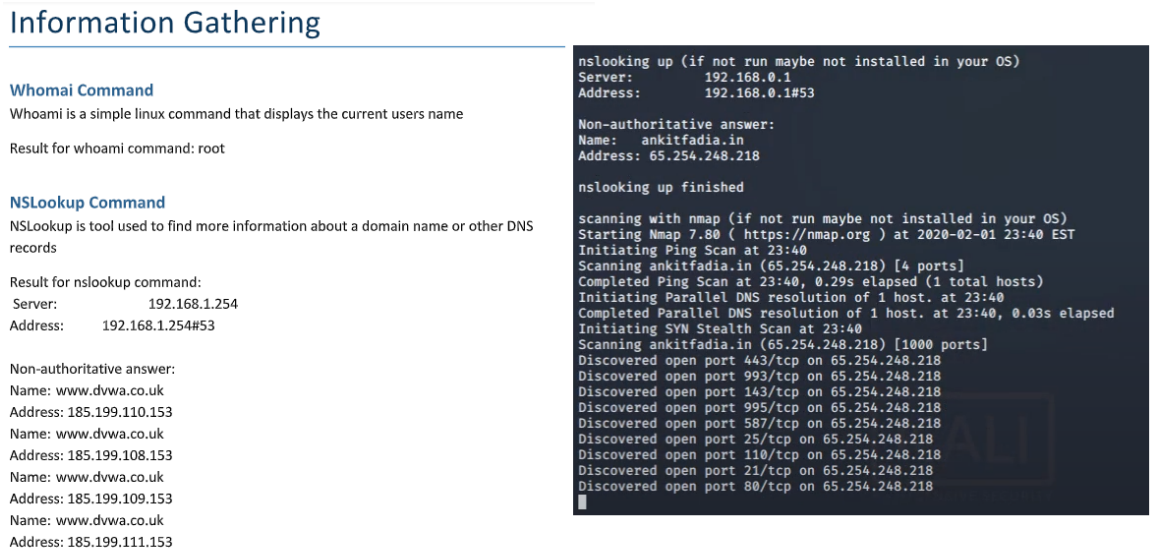


Figure 5.7: Example of Inquisitor Report Versus Yuki Chan displaying results

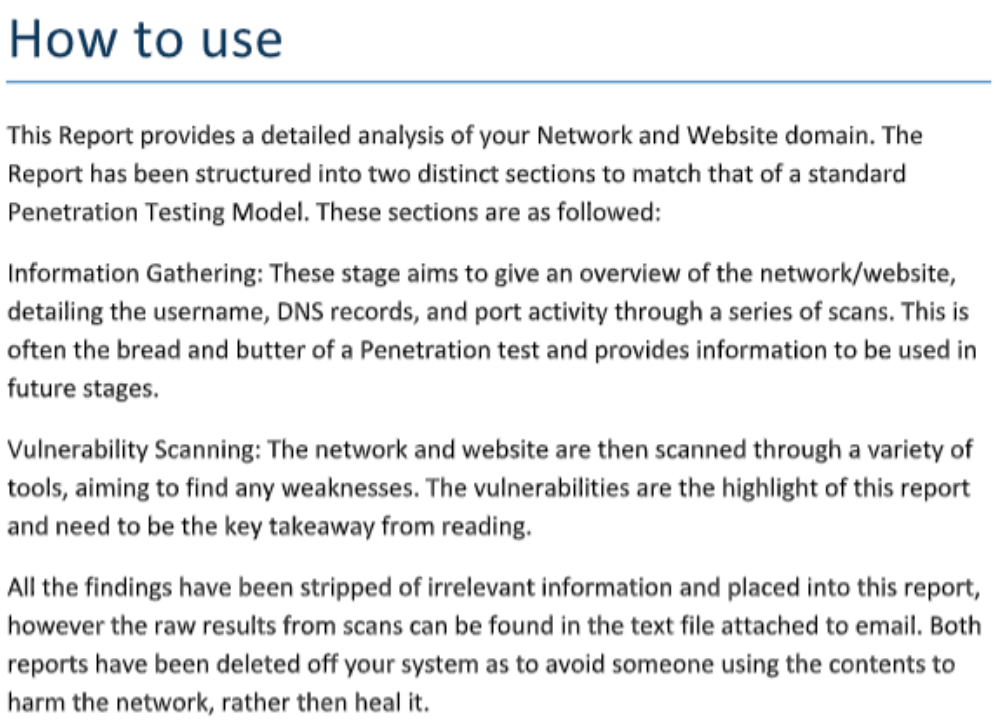
Comparing the designs of the reporting method is also important, an ugly report will be paid less attention than an aesthetically pleasing document. Figure 5.7 shows the difference between Inquisitors Smart Report and Yuki Chan's display of results, both showing the NSLookup result on screen. Inquisitor is clearly much better, with headings to display which area of the Penetration Testing Model the tools are under, clear spacing to separate entries, clean results and even a section of text describing the tool. To any user Inquisitor's results are much easier to understand, better laid out, and effectively communicate the results of the test, making the most effective solution.

5.4 Evaluation of success

The aim of this project was to develop a toolkit that will provide users with the resources they need in order to properly evaluate and improve their security through a standard Penetration testing model using automation and effective reporting.

Every objective of this aim was met. Inquisitor provides an easy to use automated Penetration test, conducting a variety of scans on the target IP address and domain. The user only has to enter minimal information to begin the test, being their email and the targets. The Smart report created is aesthetically pleasing with clean results and proper design through headings and spacing. The report is aided further by helpful text describing each tool and what purpose they serve in improving the client's security. To make Inquisitor friendly to novice users an in-depth "How to use" page has been provided at the start of every report, as seen in figure 5.8.

Inquisitor performed very well compared to other toolkits, having a similar vulnerabilities found per minute rate, and outclassing all of them in reporting on the test. Inquisitor matched many of the similar projects detailed in Chapter 2: Literature Review in features and capabilities, only lacking in exploitation which was always considered slightly out of scope.



How to use

This Report provides a detailed analysis of your Network and Website domain. The Report has been structured into two distinct sections to match that of a standard Penetration Testing Model. These sections are as followed:

Information Gathering: These stage aims to give an overview of the network/website, detailing the username, DNS records, and port activity through a series of scans. This is often the bread and butter of a Penetration test and provides information to be used in future stages.

Vulnerability Scanning: The network and website are then scanned through a variety of tools, aiming to find any weaknesses. The vulnerabilities are the highlight of this report and need to be the key takeaway from reading.

All the findings have been stripped of irrelevant information and placed into this report, however the raw results from scans can be found in the text file attached to email. Both reports have been deleted off your system as to avoid someone using the contents to harm the network, rather then heal it.

Figure 5.8: Example of Inquisitor's How to Use Page

Chapter 6

Conclusion

In Conclusion, the proposed project of a Penetration Testing Toolkit to solve the issue of small businesses or individuals wishing to properly audit and improve their security was successfully completed. Manual Penetration tests were found to be too expensive, time consuming, and resource intensive to conduct for the majority of the population, leaving many people and businesses vulnerable to attacks. Inquisitor provides an excellent solution to this through automated Penetration testing, thoroughly scanning the target system through a variety of tools, producing an easy to read Smart report. The toolkit provides an easy and free way for those who could not conduct manual Penetration tests to properly secure their system, keeping them and their customer's information safe.

Inquisitor also performed well in all tests conducted, effectively finding vulnerabilities on various systems with no issues. The Smart reporting system was also found to be far greater than any competing toolkits, even outperforming similar projects from academic sources. The focus on security and client safety allows Inquisitor to be accessible by all, without fear that using the software will aid in compromising their network. The emailing feature prevents any unwanted eyes from seeing the Report which is full of sensitive information, whilst also making it portable to the tester.

Inquisitor is not only useful to those lacking in security skills, but can also be incredibly beneficial to professional Penetration Testers. The toolkit takes around half an hour to assess the targets, which can allow professional tester to perform other more demanding actions, such as social engineering. Technical detail is also provided in the raw text dump report produced by Inquisitor, which could prove useful to these professional testers, not wanting to miss any detailed information.

Overall, Inquisitor provides an excellent method of security auditing, through its use of the Penetration Testing Standard Model and its automation of tools. Each report provides the tester with exactly what they need, no matter the skill level in security. This toolkit has the potential to be a massive aid to those who could not perform security auditing themselves, and is capable of preventing businesses or individuals from become a victim of a Malicious Hacker. Inquisitor met all the aims of this project, and fulfilled each research question created before development. Inquisitor in deployment will have a positive effect on Global security, hunting out the Vulnerabilities before they can destroy a client's system.

6.1 Future Work

Inquisitor was finished successfully, meeting all the original aims set out in the proposal. However, that does not mean it is a perfect product, several aspect of the software would benefit from improvements or new features that could be conducted for future work on the project.

As Inquisitor is a modular program, consisting of many functions of tools to conduct the Penetration test, more can always be added. The toolkit concluded with twelve tools, seven Information Gathering and five conducting various Vulnerability Scans. If more time was allotted to this project more and more tools would have been added to this section, to help improve the overall test. These potential tools could provide a variety of features and functions, as long as they improve the usefulness of the Smart report, and not just bloat it with useless information.

Several of the Vulnerability Scanning tools discussed in Chapter 5: Discussion were not able to be added to Inquisitor. These tools, like OpenVAS, would have provided the client with much more information to use, and would have made Inquisitor more efficient in the results gathered in testing. If more time was given to this project these more complex tools could have been implemented into the code, adding much more content to the Smart report. These tools simply needed more time in development, which was not possible at the time, but without the restrictions of deadline could definitely be added to Inquisitor.

This also applies to Exploitation Testing, as no successful exploits were added to Inquisitor. This was for a variety of reasons, but mainly due to technical errors and the inherent danger of automating exploits. If added to Inquisitor, the focus has to be on keeping the client's network safe, performing attacks like Denial of Service or leaking sensitive information from a database aids no-one in securing their network. Perhaps this could be done through an educational method, showing the potential attack spot on the client's network in the report, and then describing an exploit conducted on a hypothetical target. If given more time these ideas would have been explored more thoroughly, resulting in a safe and effective Exploitation Test.

Finally, the Smart report produced could have been improved slightly through a variety of methods. Visualisation is incredibly effective at describing information and would aid clients in understanding the Smart report, this was one feature that Inquisitor lacked. If given more time, a proper visualisation of vulnerabilities would have been implemented, through use of charts and tables to properly show the data in an eye-catching way.

Chapter 7

References

Journals
1. Almubairik, N.A. & Wills , G. 2016. Automated penetration testing based on a threat model. *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*. pp. 413-414.

2. Benedictis, A.D., Casola, V., Rak, M. & Villano, U. 2018. Towards Automated Penetration Testing for Cloud Applications. *2018 IEEE 27th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. pp. 24-29.
3. Chandan, A.R. & Khairnar, V.D. 2018. Security Testing Methodology of IoT. *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*. pp. 1431-1435

4. Chopra, U.K., Jyothindar, V., Pandey, R. 2020. Vulnerability Assessment and Penetration Testing: A portable solution Implementation. *2020 12th International Conference on Computational Intelligence and Communication Networks (CICN)*. pp. 398-402

5. Chu, G. & Lisitsa, A. 2018. Penetration Testing for Internet of Things and Its Automation. *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. pp. 1479-1484

6. Geer, D. & Harthorne, J. 2002. Penetration testing: a duet. *18th Annual Computer Security Applications Conference, 2002. Proceedings.* pp. 185-195.

7. Groš, S. & Kovačević, I. 2020. Red Teams - Pentesters, APTs, or Neither. *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*. pp. 1242-1249

8. Jia, Q., Qiu, X., Wang, S., Xia, C. & Xia, Q. 2014. An automated method of penetration testing. *2014 IEEE Computers, Communications and IT Applications Conference.* pp. 211-216

9. Samant, N. 2011. AUTOMATED PENETRATION TESTING. Master's Projects

10. Shah, S. & Mehtre, B.M. 2014. An automated approach to Vulnerability Assessment and Penetration Testing using Net-Nirikshak 1.0. *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*. pp. 707-712.

11. Stefinko, Y., Piskozub, A. & Banakh, R. 2016. Manual and automated penetration testing. Benefits and drawbacks. Modern tendency. *2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET)*. pp. 488-491.

12. Sweigart, A. 2020. *Automate the Boring Stuff with Python*. Second Edition. William Pollock. San Francisco.

13. The PTES Team. 2017. The Penetration Testing Execution Standard. "The Penetration Testing Execution Standard Documentation Release 1.1". pp. 3

Other References

Alpine Security. 2021. The History of Penetration Testing.[online] Alpine Security. Available at: https://alpinesecurity.com/blog/history-of-penetration-testing/ [Accessed 29 March 2021]

ClamAV. 2021. ClamAV Documentation. [online] ClamAV. Available at: https://www.clamav.net/documents/scanning [Accessed 02 May 2021]

GitHub. 2020. Kaboom. [online] GitHub. Available at: https://github.com/Leviathan36/kaboom [Accessed 5 May 2021]

GitHub. 2017. Yuki Chan. [online] GitHub. Available at: https://github.com/Yukinoshita47/Yuki-Chan-The-Auto-Pentest [Accessed 5 May 2021]

Guthrie-Jensen, 2021. Top Project Management Approaches Explained [A Visual Guide]. [online] Guthrie Jensen. Available at: https://guthriejensen.com/blog/top-project-management-approaches-visual-guide/ [Accessed 02 May 2021]

Johnson, J. 2021. IC3: total damage caused by reported cyber crime 2001-2020. [online] Statista. Available at: https://www.statista.com/statistics/267132/total-damage-caused-by-by-cyber-crime-in-the-us/ [Accessed 02 May 2021]

Kamil. 2019. Direct link to a hosted image in email signatures. Available at: https://www.mail-signatures.com/articles/direct-link-to-hosted-image/ [Accessed 02 May 2021]

Langen, J.D. 2019. Sending Emails With Python. [online] RealPython. Available at: https://realpython.com/python-send-email/ [Accessed 02 May 2021]

Lapena, R. 2019. Unpatched Vulnerabilities Caused Breaches in 27% of Orgs, Finds Study. [online] TripWire. Available at:
https://www.tripwire.com/state-of-security/vulnerability-management/
unpatched-vulnerabilities-breaches/ [Accessed 5 May 2021]

Pixlr. 2021. Pixlr Photo editor. [online] Pixlr. Available at:
https://pixlr.com/ [Accessed 5 May 2021]

Singleton, C. 2021. Top 10 Cybersecurity Vulnerabilities of 2020. [online] Security Intelligence. Available at:
https://securityintelligence.com/posts/top-10-cybersecurity-vulnerabilities-2020/ [Accessed 5 May 2021]

The MITRE Corporation. 2021. Common Vulnerabilities and Exposure Database. [online] CVE. Available at:
https://cve.mitre.org/ [Accessed 29 March 2021]

Chapter 8

Appendix

8.1 Appendix A - Source Code

8.1.1 Section 1: Inquisitor.py

```
import os
import nmap
import datetime
import sender
import email, smtplib, ssl
from email import encoders
from email.mime.base import MIMEBase
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from docx import Document
from docx.enum.text import WD_ALIGN_PARAGRAPH
import docx
from docx.shared import Inches
from docx.shared import Pt
import subprocess
import re

#Print Logo
logoRead = open('TextLogo.txt', 'r')
logoOutput = logoRead.read()
print(logoOutput)
logoRead.close()

#Start Document and set font
document = Document()
style = document.styles['Normal']
font = style.font
font.name = 'Calibri'
font.size = Pt(12)
```

```
#IP address array, discovered by Nmap
ipAddresses = []
vulnerability = []
vulnerableIP = []

#Regex to check email is valid
emailChecker = '^(\w|\.|\_|\-)+[@](\w|\_|\-|\.)+[.]\w{2,3}$'

#Create Report title page and how to use page
def reportCreation(document):
        document.add_picture('inquis.jpg', width=Inches(6), height=Inches(1.5))
        titlePage = document.add_paragraph("")
        titlePage.alignment = WD_ALIGN_PARAGRAPH.CENTER
        titlePage.add_run("Smart Report from Penetration Test").font.size = Pt(26)
        lineBreak = titlePage.add_run()
        lineBreak.add_break()
        #date and time
        date = datetime.datetime.today()
        hour = date.hour
        minute = date.minute
        second = date.second
        day = date.day
        month = date.month
        year = date.year
        timeCheck = titlePage.add_run("Test conducted at: "
        + (str(hour) + ":" + str(minute) + ":" + str(second) +
        " on the " + str(day) + "/" + str(month) + "/" + str(year)))
        timeCheck.font.size = docx.shared.Pt(18)

        #Logo Placement
        addTitleImage = titlePage.add_run("")
        addTitleImage.add_picture('InquisitorLogo.png',
        width=Inches(5), height=Inches(6))
        document.add_page_break()

        #How to use Page
        document.add_heading('How to use', 0)
        document.add_paragraph('This Report provides a detailed
        analysis of your Network and Website domain. The Report
        has been structured into two distinct sections to match
        that of a standard Penetration Testing Model. These sections
        are as followed:')
        document.add_paragraph('Information Gathering: These stage
        aims to give an overview of the network/website, detailing the
        username, DNS records, and port activity through a series of
        scans. This is often the bread and butter of a Penetration
        test and provides information to be used in future stages.')
        document.add_paragraph('Vulnerability Scanning: The network
```

```
        and website are then scanned through a variety of tools,
        aiming to find any weaknesses. The vulnerabilities are the
        highlight of this report and need to be the key takeaway
        from reading.')
        document.add_paragraph('All the findings have been stripped
        of irrelevant information and placed into this report, however
        the raw results from scans can be found in the text file
        attached to email. Both reports have been deleted off your
        system as to avoid someone using the contents to harm the
        network, rather then heal it.')
        document.add_page_break()

#whoami function with reporting
def whoamiTool():
        print("Conducting whoami command (1/12)")
        result = subprocess.Popen('whoami', stdout=subprocess.PIPE
        , stderr=subprocess.PIPE)
        stdout,_ = result.communicate()
        out = stdout.strip()
        out = out.strip()
        whoamiResult = out.decode('ascii')

        #open and add to raw file
        font.size = Pt(12)
        textFile = open("textdump.txt", "a")
        textFile.write("Result for whoami command: " +
        whoamiResult + "\n")
        textFile.write("-------------------------------------------------------------------------------- \n")
        #write to smart report
        document.add_heading('Whomai Command', level=1)
        document.add_paragraph('Whoami is a simple linux command
        that displays the current users name')
        document.add_paragraph('Result for whoami command: %s' % whoamiResult)

#nslookup function with reporting
def nslookupTool(domainName):
        print("Conducting nslookup command (2/12)")
        s = subprocess.Popen(['nslookup', domainName], stdout=subprocess.PIPE,
        stderr=subprocess.PIPE)
        stdout,_ = s.communicate()
        out = stdout.strip()
        out = out.strip()
        nslookupResult = out.decode('ascii')
        #open and add to raw file
        font.size = Pt(12)
        textFile = open("textdump.txt", "a")
        textFile.write("Result for nslookup command: \n" + nslookupResult + "\n")
        textFile.write("-------------------------------------------------------------------------------- \n")
```

```
        #write to smart report
        document.add_heading('NSLookup Command', level=1)
        document.add_paragraph('NSLookup is tool used to find more information
        about a domain name or other DNS records')
        document.add_paragraph('Result for nslookup command: \n %s' % nslookupResult)

#whois function with reporting
def whoisTool(domainName):
        print("Conducting whois command (3/12)")
        strip, strippedName = domainName.split('www.')
        s = subprocess.Popen(['whois', strippedName], stdout=subprocess.PIPE
        , stderr=subprocess.PIPE)
        stdout,_ = s.communicate()
        out = stdout.strip()
        out = out.strip()
        whoisRaw = out.decode('utf-8')
        try:
                whoisResult, stripped = whoisRaw.split('>>>', 1)
        except ValueError:
                whoisResult, stripped = whoisRaw.split('WHOIS lookup made at', 1)

        #open and add to raw file
        font.size = Pt(12)
        textFile = open("textdump.txt", "a")
        textFile.write("Result for whois command: \n" + whoisResult + "\n")
        textFile.write("-------------------------------------------------------------- \n")
        #write to smart report
        document.add_heading('Whois Command', level=1)
        document.add_paragraph('Whois is a listing system that
        show records about websites, such as ownership, domains, and other useful infor
        document.add_paragraph('Result for whois command: \n %s' % whoisResult)

def dmitryTool(domainName):
        print("Conducting DMitry scan (4/12)")
        strip, strippedName = domainName.split('www.')
        s = subprocess.Popen(['dmitry', '-nse', strippedName]
        , stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        stdout,_ = s.communicate()
        out = stdout.strip()
        out = out.strip()
        dmitryRaw = out.decode('ascii')
        strip1, dmitryRaw2 = dmitryRaw.split('"There be some
        deep magic going on"')
        dmitryResult, strip2 = dmitryRaw2.split('All scans completed')
        #open and add to raw file
        font.size = Pt(12)
        textFile = open("textdump.txt", "a")
        textFile.write("Result for DMitry command: \n" + dmitryResult + "\n")
```

```
        textFile.write("---------------------------------------------------------- \n")
        #write to smart report
        document.add_heading('DMitry Command', level=1)
        document.add_paragraph('DMitry (Deepmagic Information Gathering Tool) is a
        command line application used to find a wide variety of information about
        a host. DMitry is used here to find subdomains of the host and possible
        email addresses attached to it, whilst displaying basic Netcraft
        information.')
        document.add_paragraph('Result for whois command: \n %s' % dmitryResult)

def dnswalkTool(domainName):
        print("Conducting DNSWalki command (5/12)")
        strip, strippedName = domainName.split('www.')
        dnsName = strippedName+'.'
        s = subprocess.Popen(['dnswalk', dnsName], stdout=subprocess.PIPE
        , stderr=subprocess.PIPE)
        stdout,_ = s.communicate()
        out = stdout.strip()
        out = out.strip()
        dnswalkRaw = out.decode('ascii')
        #open and add to raw file
        font.size = Pt(12)
        textFile = open("textdump.txt", "a")
        textFile.write("Result for DNSWalk command: \n" + dnswalkRaw + "\n")
        textFile.write("---------------------------------------------------------- \n")
        #write to smart report
        document.add_heading('DNSWalk Command', level=1)
        document.add_paragraph('DNSWalk is a Domain Name System (DNS) debugger
        , checking a domain for consistency and accuracy through zone
        transfers.')
        document.add_paragraph('Result for DNSWalk command: \n %s' % dnswalkRaw)

#Nmap scan, discovery of hosts and reporting of results
def nmapScanTool(ipaddr):
        print("Conducting Nmap scan (6/12)")
        font.size = Pt(12)
        document.add_heading('Nmap port scan', level=1)
        document.add_paragraph("Nmap is network scanning tool used in the majority
        of penetration tests, it can discover hosts to be used in further tests and
        services on the network, Nmap can also provide information on which ports
        are open.")
        #Start the Scan
        nmScan = nmap.PortScanner()
        nmScan.scan(ipaddr + ' -oN textdump.txt --append-output')
        #Output to the Smart Report
        for host in nmScan.all_hosts():
                document.add_paragraph('Host : %s (%s)' %
                (host, nmScan[host].hostname()))
```

```
                ipAddresses.append(host)
                document.add_paragraph('State : %s' % nmScan[host].state())
                for proto in nmScan[host].all_protocols():
                        document.add_paragraph('----------')
                        document.add_paragraph('Protocol : %s' % proto)
                        lport = nmScan[host][proto].keys()
                        for port in lport:
                                document.add_paragraph('port :
                                %s\tstate : %s' % (port, nmScan[host]
                                [proto][port]['state']))
                document.add_paragraph('-----------------------------------------')

#OS Detection with discovered hosts and reporting
def osDetectTool():
        print("Conducting OS detection (7/12)")
        amount = 0
        document.add_heading('OS Detection', level=1)
        document.add_paragraph('Nmap also provides an excellent OS Detection scan
        , using the result collected previously the entire networks Operating
        Systems can be found. This information can be useful for finding out more
        about unknown hosts or simply to map your network.')
        #Conducting an OS detection on each IP address found
        while amount < len(ipAddresses):
                try:
                        s = subprocess.Popen(['nmap', '-script', 'vuln','-O',
                        ipAddresses[amount], '-oN', 'textdump.txt',
                        '--append-output'], stdout=subprocess.PIPE,
                        stderr=subprocess.PIPE)
                        stdout,_ = s.communicate()
                        out = stdout.strip()
                        out = out.strip()
                        osDetectRaw = out.decode('ascii')
                        #Stripping repeated information
                        stripped, osStrip = osDetectRaw.split('Device type')
                        try:
                                osResult, vulns = osStrip.split
                                ('Host script results:')
                                vulnResult, stripped = osStrip.split
                                ('OS detection performed
                                . Please report any incorrect
                                results at')
                                vulnerability.append(vulnResult)
                                vulnerableIP.append(amount)

                        except ValueError:
                                #Stripping Irrelevant Information
                                osResult, stripped = osStrip.split
                                ('OS detection performed.
```

```
                                Please report any incorrect
                                results at')

                        #write to smart report
                        document.add_paragraph('Result for
                        %s: ' % ipAddresses[amount])
                        document.add_paragraph('Device type
                        %s' % osResult)
                        document.add_paragraph('-----------------------------------------')
                #If no OS was detected
                except ValueError:
                        document.add_paragraph("No OS was detected for %s" %
                        ipAddresses[amount])
                        document.add_paragraph('-----------------------------------------')
                amount += 1

def nmapvulnTool():
        print("Conducting Nmap Vulnerability Scanning (8/12)")
        document.add_heading('Nmap Vulnerability Scanning', level=1)
        document.add_paragraph('Nmap provides an excellent method
        of vulnerability scanning, that has the power to scan remote
        hosts. Each discovered Ip address is scanned for Vulnerabilities
        and report on.')
        amount = 0
        while amount < len(vulnerability):
                #write to smart report
                        place = vulnerableIP[amount]
                        document.add_paragraph('%s was found to be vulnerable:
                        ' % ipAddresses[place])
                        document.add_paragraph('Host script results: %s' %
                        vulnerability[amount])
                        document.add_paragraph('-----------------------------------------')
                        amount += 1

def niktoTool(domainName):
        print("Conducting Nikto Vulnerability Scanning (9/12)")
        s = subprocess.Popen(['nikto', '-h', domainName],
        stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        stdout,_ = s.communicate()
        out = stdout.strip()
        out = out.strip()
        niktoResult = out.decode('ascii')
        #open and add to raw file
        font.size = Pt(12)
        textFile = open("textdump.txt", "a")
        textFile.write("Result for Nikto Vulnerability Scan: \n"
        + niktoResult + "\n")
        textFile.write("-------------------------------------------------------- \n")
```

```
        #write to smart report
        document.add_heading('Nikto Vulnerability Scan', level=1)
        document.add_paragraph('Nikto is a webserver vulnerability
        scanner that provides information on dangerous files, possible
        attacks (such as cross-site-scripting), and general problems.
        It can also find cookies on the webserver.')
        document.add_paragraph('Result for Nikto Vulnerability Scan:
        \n %s' % niktoResult)

def clamscanTool():
        print("Conducting ClamScan Virus Detection (10/12)")
        s = subprocess.Popen('clamscan', stdout=subprocess.PIPE
        , stderr=subprocess.PIPE)
        stdout,_ = s.communicate()
        out = stdout.strip()
        out = out.strip()
        clamscanRaw = out.decode('ascii')
        stripped, clamscanResult = clamscanRaw.split
        ('----------- SCAN SUMMARY -----------')
        #open and add to raw file
        font.size = Pt(12)
        textFile = open("textdump.txt", "a")
        textFile.write("Result for ClamScan Virus Detection:
        \n" + clamscanResult + "\n")
        textFile.write("------------------------------------------------------------ \n")
        #write to smart report
        document.add_heading('ClamScan Virus Detection', level=1)
        document.add_paragraph('ClamScan is an excellent tool
        used for virus detection on a system, providing detailed results
        of its findings compared to a large database of possible viruses.')
        document.add_paragraph('Result for ClamScan Virus Detection:
        \n %s' % clamscanResult)

def chkrootkitTool():
        print("Conducting Chkrootkit scan (11/12)")
        #write to smart report
        document.add_heading('ChkRootKit Root Kit Detection Scan', level=1)
        document.add_paragraph('ChkRootKit is a commonly used application
        that scans a system for possible root kits.')
        s = subprocess.Popen('chkrootkit', stdout=subprocess.PIPE
        , stderr=subprocess.PIPE)
        stdout,_ = s.communicate()
        out = stdout.strip()
        out = out.strip()
        rootRaw = out.decode('ascii')
        try:
            stripped, rootResult = rootRaw.split('Warning:')
            #open and add to raw file
```

```
            font.size = Pt(12)
            textFile = open("textdump.txt", "a")
            textFile.write("Result for ChkRootKit Root Kit Detection Scan:
            \n" + rootResult + "\n")
            textFile.write("-------------------------------------------------------------
            \n")
            document.add_paragraph('Result for ChkRootKit Root Kit Detection:
            \n %s' % rootResult)
        except ValueError:
            #open and add to raw file
            font.size = Pt(12)
            textFile = open("textdump.txt", "a")
            textFile.write('ChkRootKit returned no Warning, meaning no
            Root Kits were found on your system.' + "\n")
            textFile.write("-------------------------------------------------------------
            \n")
            document.add_paragraph('ChkRootKit returned no Warning, meaning no
            Root Kits were found on your system.')

def lynisTool():
        print("Conducting Lynis security audit (12/12)")
        s = subprocess.Popen(['lynis', '--pentest'],
        stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        stdout,_ = s.communicate()
        out = stdout.strip()
        out = out.strip()
        lynisRaw = out.decode('ascii')
        strip1 ,lynisRaw2 = lynisRaw.split('Lynis 2.6.2 Results')
        lynisResult, strip2 = lynisRaw2.split('Follow-up')
        lynisCleanResult = lynisResult.replace('[0m', '').
        replace('[1;31m', '').replace('[1;37m', '')
        .replace('[0;36m', '').replace('[1;33m', '')
        .replace('[0', '').replace(';37m', '')
        #open and add to raw file
        textFile = open("textdump.txt", "a")
        textFile.write("Result for Lynis Security Audit: \n"
        + lynisCleanResult + "\n")
        textFile.write("-------------------------------------------------------------- \n")
        #write to smart report
        document.add_heading('Lynis Security Audit', level=1)
        document.add_paragraph('Lynis is a security auditing tool that
        provides a detailed responce on a systems weaknesses and issues
        with the goal of hardening the system.')
        document.add_paragraph('Result for Lynis Security Audit:
        \n -[ Lynis 2.6.2 Results %s' % lynisCleanResult)

#Main with user inputs and cleanup
def main():
```

```
print("Enter the email of the tester
(Reports will be sent to this address): ")
validEmail = False
#Checking if the Email address is valid
while validEmail == False:
        reciever = str(input())
        if(re.search(emailChecker, reciever)):
            validEmail = True
        else:
            print("That is not a valid email: ")
#Taking in the intial IP
print("Enter a IP address to begin the Investigation: ")
ipaddr = str(input())
#Taking in the domain name
print("Enter Domain name (i.e. www.name.com)")
domainName = str(input())

#Main operation of Inquisitor
reportCreation(document)
document.add_heading('Information Gathering', 0)
whoamiTool()
nslookupTool(domainName)
whoisTool(domainName)
dmitryTool(domainName)
dnswalkTool(domainName)
nmapScanTool(ipaddr)
osDetectTool()
#Vulns
document.add_page_break()
document.add_heading('Vulnerability Scanning', 0)
nmapvulnTool()
niktoTool(domainName)
clamscanTool()
chkrootkitTool()
lynisTool()




#Emailing and Cleanup
document.save('report.docx')
sender.emailSender(reciever)
os.remove("report.docx")
os.remove("textdump.txt")
print("Report emailed and removed off device")
```

main ( )

8.1.2 Section 2: Sender.py

```
import os
import nmap
import datetime
import email, smtplib, ssl
from email import encoders
from email.mime.base import MIMEBase
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from docx import Document



def emailSender(reciever):
        emailport = 587
        password = "password"
        sender = "smartpentestreport@gmail.com"
        subject = "Smart Report and Text dump from Inquisitor"
        #Mime stuff
        message = MIMEMultipart()
        message["From"] = sender
        message["To"] = reciever
        message["Subject"] = subject
        html = """\
        <html>
         <body>
          <div style="text-align:center;">
           <img src="https://drive.google.com/uc?export=view&id=1jpl4bItstEtP-
           vcoL-yDQ1dy969bY2je" title="Inquisitor" width="600" height="400" />
           <h3>Greetings</h3><br>
           <p>Please find attached the full report of my findings from
           the Penetration Test  and a text file containing the raw
           data from scans.<br>
           Thank you for using Inquisitor.
           </p>
          </div>
         </body>
        </html>
        """
        htmlpart = MIMEText(html, "html")
        message.attach(htmlpart)

        filename = ['report.docx', 'textdump.txt']
        for filename in filename or []:
                with open(filename, "rb") as attachment:
                        part =MIMEBase("application", "octet-stream")
                        part.set_payload(attachment.read())
```

```
                encoders.encode_base64(part)
                part.add_header(
                        "Content-Disposition",
                        'attachment; filename= "%s"'
                        % os.path.basename(filename))
                message.attach(part)

#Adding the Attachement
text = message.as_string()

#Sending the email
context = ssl.create_default_context()
with smtplib.SMTP('smtp.gmail.com', emailport) as server:
    server.ehlo()
    server.starttls()
    server.login(sender, password)

    server.sendmail(sender, reciever, text)
```

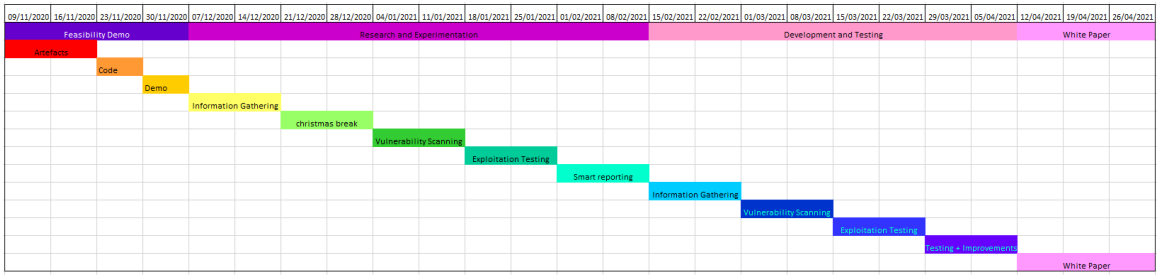
8.1.3 Section 3: ImageToAscii

```
import PIL.Image

ASCII_CHARS = ["@", "#", "S", "%", "?", "*", "+", ";", ":", ",", "."]

def resizeLogo(image, newWidth=100):
    width, height = image.size
    ratio = height / width
    newHeight = int(new_width * ratio)
    resized_logo = image.resize((newWidth, newHeight))
    return(resized_logo)

def grayify(image):
    greyscaleLogo = image.convert("L")
    return(greyscaleLogo)

def pixelsToAscii(image):
    pixels = image.getdata()
    characters = "".join([ASCII_CHARS[pixel//25] for pixel in pixels])
    return(characters)

def main(newWidth=100):
    path = input("Enter an Image name to turn to ASCII:\n")
    newImage = pixelsToAscii(grayify(resizeLogo(image)))

    pixelCount = len(new_image_data)
    asciiLogo = "\n".join(newImage[i:(i+newWidth)] for i in
    range(0, pixelCount, newWidth))
    print(asciiLogo)

    with open("asciiImage.txt", "w") as f:
        f.write(asciiLogo)

main()
```

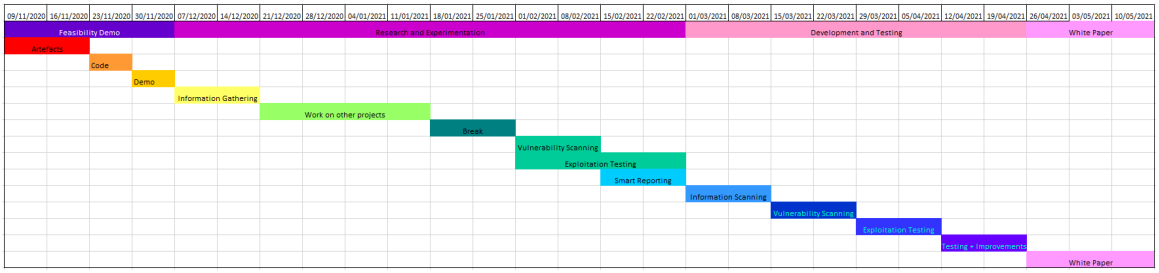
8.2 Appendix B - Design Documents

8.2.1 Section 1: Gantt Charts

Old Gantt Chart



Updated Gantt Chart



8.2.2 Section 2: Risk Matrix

Frequency/ Consequence	1-Very Unlikely	2-Remote	3-Occasional	4-Probable	5-Frequent
4-Catastrophic		R4			
3-Critical		R6	R3		
2-Major		R2		R1	
1-Significant			R5		
1-Minor					

- R1: Unrealistic Time Estimate.
  - There is the risk of incorrectly estimating the time for each task allocated in the Gantt chart, this could lead to the delivery date shifting and features lacking development.

  - This is being mitigated by giving each period of development a good length of time so all tasks can be completed.

- R2: Feature bloating.
  - The toolkit has potential to encompass many aspects of Penetration testing, this could inadvertently lead to some features having too much content, whilst others lack substance.
  - Set limits have been placed on each section of development as to fulfill the projects requirements without missing out features. Once each section is finished their is a time slot given for adding improvements and new features without the risk of feature bloating.

- R3: Development Issues.
  - There is the risk that the developer of the toolkit does not have the necessary skill and/or experience to finish this project.
  - To mitigate this significant time has been dedicated to each section of development as to give the developer a chance to work through issues and gain skill whilst producing the end product. Research is also being conducted before hand in order to hit the ground running when development starts.

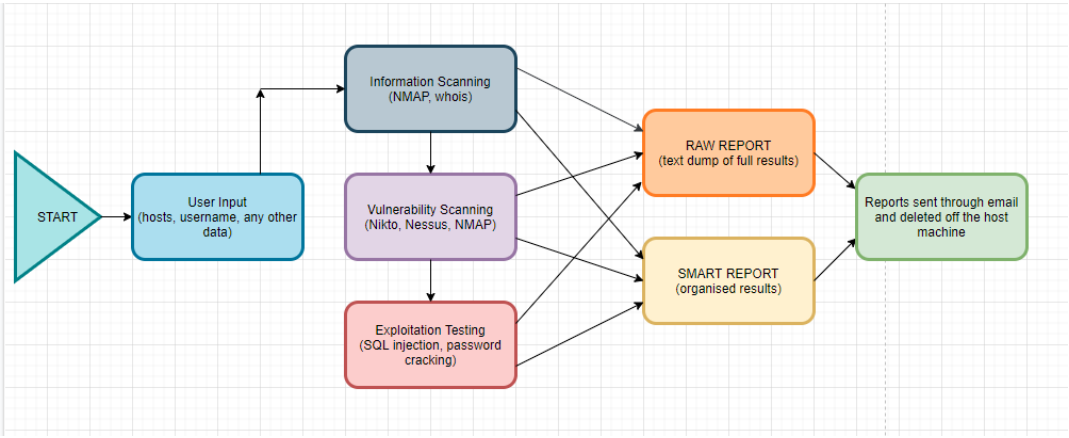
- R4: Act of God.
  - There is always the risk that an event completely outside the control of the developer occurs which could cause the project to slow down or even become impossible in the remaining time. Events could include ecological disasters/problems, pandemics, government issues, etc.
  - There is little way to mitigate this risk other than to have a stable working environment that falls out of reach from the problem at hand, this risk is also far less likely than others on the list.

- R5: Toolkit Performance Issues.
  - There is the risk that the toolkit only works on certain machines, and is ineffective or useless on others.
  - Research and testing is being conducted throughout the entire development in order to mitigate this risk, aiming to make the toolkit as accessible as possible.

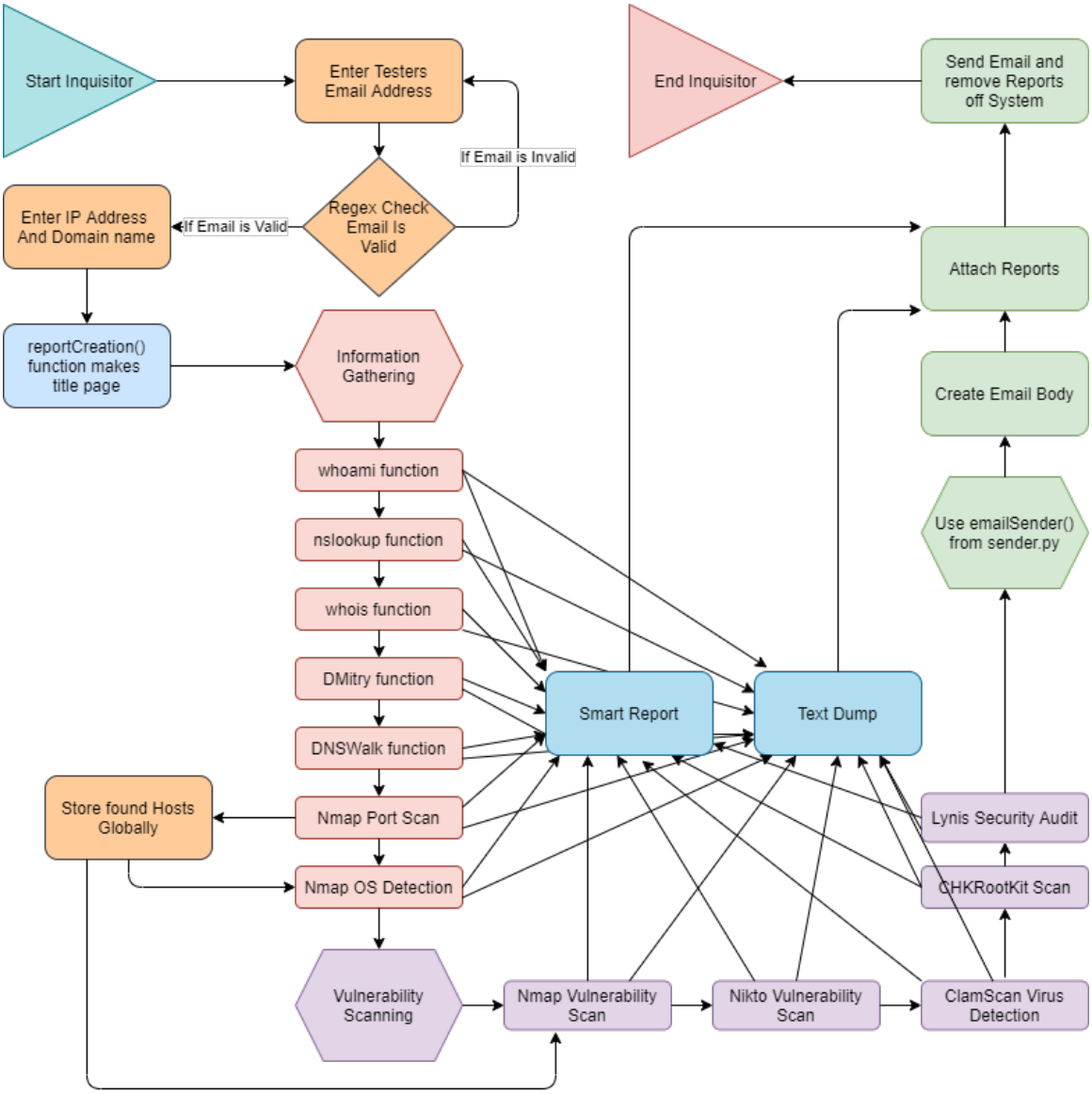
- R6: Hardware/Software Failure.
  - If the computer used to develop the toolkit has a significant failure causing loss of progress or total loss of the project. Similar problems could occur with software used, if the main resource used in development abruptly stops working then the whole project will have to shift, causing loss of time and progress.
  - To mitigate these risks backups of the project will be created and stored in various locations, such as google drive and physical storage. Software is harder to mitigate, but as the toolkit it being developed in Python it can easily be switched between text editors. If a virtual machine used to develop the toolkit fails several alternatives have been lined up to take its place, such as Arch linux, Kali Linux and Ubuntu.

8.2.3 Section 2: Flowchart

Old Flow Chart



Updated Flow Chart



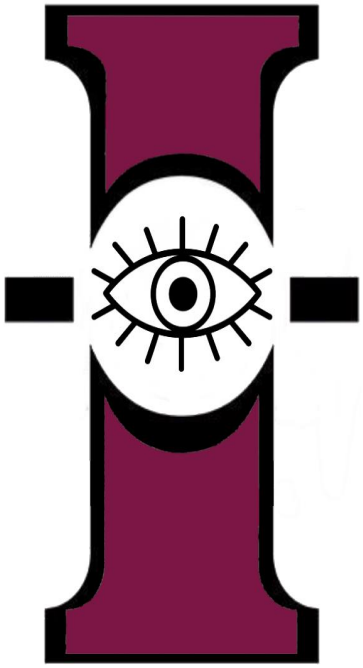
8.3 Apendix C - Example Reports

8.3.1 Section 1: Smart Report

INQUISITOR

Smart Report from Penetration Test

Test conducted at: 11:17:30 on the 13/5/2021



How to use

This Report provides a detailed analysis of your Network and Website domain. The Report has been structured into two distinct sections to match that of a standard Penetration Testing Model. These sections are as followed:

Information Gathering: These stage aims to give an overview of the network/website, detailing the username, DNS records, and port activity through a series of scans. This is often the bread and butter of a Penetration test and provides information to be used in future stages.

Vulnerability Scanning: The network and website are then scanned through a variety of tools, aiming to find any weaknesses. The vulnerabilities are the highlight of this report and need to be the key takeaway from reading.

All the findings have been stripped of irrelevant information and placed into this report, however the raw results from scans can be found in the text file attached to email. Both reports have been deleted off your system as to avoid someone using the contents to harm the network, rather then heal it.

Information Gathering

Whomai Command

Whoami is a simple linux command that displays the current users name

Result for whoami command: root

NSLookup Command

NSLookup is tool used to find more information about a domain name or other DNS records

Result for nslookup command:
Server: 192.168.1.254
Address: 192.168.1.254#53

Non-authoritative answer:
Name: www.dvwa.co.uk
Address: 185.199.110.153
Name: www.dvwa.co.uk
Address: 185.199.108.153
Name: www.dvwa.co.uk
Address: 185.199.109.153
Name: www.dvwa.co.uk
Address: 185.199.111.153

Whois Command

Whois is a listing system that show records about websites, such as ownership, domains, and other useful informaton

Result for whois command:
Domain name:
dvwa.co.uk

Data validation:
Nominet was able to match the registrant's name and address against a 3rd party data source on 24-Apr-2018

Registrar:
1&1 Ionos SE [Tag = 1AND1]
URL: https://www.ionos.co.uk

Relevant dates:
Registered on: 30-Aug-2009
Expiry date: 30-Aug-2021
Last updated: 21-Jan-2021

Registration status:
Registered until expiry date.

Name servers:
ns1.digitalocean.com
ns2.digitalocean.com
ns3.digitalocean.com

DMitry Command

DMitry (Deepmagic Information Gathering Tool) is a command line application used to find a wide variety of information about a host. DMitry is used here to find subdomains of the host and possible email addresses attached to it, whilst displaying basic Netcraft information.

Result for whois command:

HostIP:185.199.109.153
HostName:dvwa.co.uk

Gathered Netcraft information for dvwa.co.uk

Retrieving Netcraft.com information for dvwa.co.uk
Netcraft.com Information gathered

Gathered Subdomain information for dvwa.co.uk

Searching Google.com:80...
HostName:www.dvwa.co.uk
HostIP:185.199.111.153
Searching Altavista.com:80...
Found 1 possible subdomain(s) for host dvwa.co.uk, Searched 0 pages containing 0 results

Gathered E-Mail information for dvwa.co.uk

Searching Google.com:80...
Searching Altavista.com:80...
Found 0 E-Mail(s) for host dvwa.co.uk, Searched 0 pages containing 0 results

DNSWalk Command

DNSWalk is a Domain Name System (DNS) debugger, checking a domain for consistency and accuracy through zone transfers.

Result for DNSWalk command:
Checking dvwa.co.uk.
FAIL: Zone transfer of dvwa.co.uk. from ns1.digitalocean.com failed: NOTIMP
FAIL: Zone transfer of dvwa.co.uk. from ns2.digitalocean.com failed: NOTIMP
FAIL: Zone transfer of dvwa.co.uk. from ns3.digitalocean.com failed: NOTIMP
BAD: All zone transfer attempts of dvwa.co.uk. failed!

Nmap port scan

Nmap is network scanning tool used in the majority of penetration tests, it can discover hosts to be used in further tests and services on the network, Nmap can also provide information on which ports are open.

Host : 192.168.1.254 (bthub)

State : up

Protocol : tcp

port : 53 state : open

port : 80 state : open

port : 139 state : open

port : 443 state : open

port : 445 state : open

port : 7911 state : open

port : 8888 state : open

port : 49152 state : open

Host : 192.168.1.70 (Irelia)

State : up

Protocol : tcp

port : 443 state : open

port : 903 state : open

port : 5357 state : open

Host : 192.168.1.98 (mx)

State : up

Protocol : tcp

port : 22 state : open

port : 111 state : open

port : 139 state : open

port : 445 state : open

port : 6566 state : open

OS Detection

Nmap also provides an excellent OS Detection scan, using the result collected previously the entire networks Operating Systems can be found. This information can be useful for finding out more about unknown hosts or simply to map your network.

No OS was detected for 192.168.1.254

Result for 192.168.1.70:

Device type : general purpose
Running (JUST GUESSING): Microsoft Windows 2008 (90%)
OS CPE: cpe:/o:microsoft:windows_server_2008::sp1
cpe:/o:microsoft:windows_server_2008:r2
Aggressive OS guesses: Microsoft Windows Server 2008 SP1 or Windows Server 2008 R2 (90%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

Result for 192.168.1.98:

Device type : general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.32
OS details: Linux 2.6.32
Network Distance: 0 hops

Vulnerability Scanning

Nmap Vulnerability Scanning

Nmap provides an excellent method of vulnerability scanning, that has the power to scan remote hosts. Each discovered Ip address is scanned for Vulnerabilities and report on.

192.168.1.98 was found to be vulnerable:

Host script results: : general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.32
OS details: Linux 2.6.32
Network Distance: 0 hops

Host script results:
|_smb-vuln-ms10-054: false
|_smb-vuln-ms10-061: false
| smb-vuln-regsvc-dos:
| VULNERABLE:
| Service regsvc in Microsoft Windows systems vulnerable to denial of service
| State: VULNERABLE
| The service regsvc in Microsoft Windows 2000 systems is vulnerable to denial of service caused by a null deference
| pointer. This script will crash the service if it is vulnerable. This vulnerability was discovered by Ron Bowes
| while working on smb-enum-sessions.
|_

Nikto Vulnerability Scan

Nikto is a webserver vulnerability scanner that provides information on dangerous files, possible attacks (such as cross-site-scripting), and general problems. It can also find cookies on the webserver.

Result for Nikto Vulnerability Scan:
 - Nikto v2.1.5

+ Target IP: 185.199.110.153
+ Target Hostname: www.dvwa.co.uk
+ Target Port: 80
+ Start Time: 2021-05-13 11:31:29 (GMT1)

+ Server: GitHub.com
+ Retrieved via header: 1.1 varnish
+ Retrieved x-served-by header: cache-lcy19233-LCY
+ The anti-clickjacking X-Frame-Options header is not present.
+ Uncommon header 'x-fastly-request-id' found, with contents:
8ae85bc8ec0928bd430dbb3fc6643d7019b0e07f
+ Uncommon header 'x-cache-hits' found, with contents: 2
+ Uncommon header 'x-cache' found, with contents: HIT
+ Uncommon header 'x-served-by' found, with contents: cache-lcy19233-LCY
+ Uncommon header 'x-github-request-id' found, with contents:
6248:187E:1599821:16A5F42:609D0001
+ Uncommon header 'x-timer' found, with contents: S1620901889.452424,VS0,VE0
+ Root page / redirects to: https://dvwa.co.uk/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server leaks inodes via ETags, header found with file /, fields: 0x5f7baea5 0x239b
+ Uncommon header 'content-security-policy' found, with contents: default-src 'none';
style-src 'unsafe-inline'; img-src data:; connect-src 'self'
+ Uncommon header 'x-xss-protection' found, with contents: 0
+ Uncommon header 'x-content-type-options' found, with contents: nosniff
+ Uncommon header 'strict-transport-security' found, with contents: max-
age=31536000
+ Uncommon header 'x-frame-options' found, with contents: deny
+ Server banner has changed from 'GitHub.com' to 'Varnish' which may suggest a WAF,
load balancer or proxy is in place
+ Uncommon header 'x-origin-cache' found, with contents: HIT
+ 6544 items checked: 0 error(s) and 16 item(s) reported on remote host
+ End Time: 2021-05-13 11:43:53 (GMT1) (744 seconds)

+ 1 host(s) tested

ClamScan Virus Detection

ClamScan is an excellent tool used for virus detection on a system, providing detailed results of its findings compared to a large database of possible viruses.

Result for ClamScan Virus Detection:

Known viruses: 8528798
Engine version: 0.103.2
Scanned directories: 1
Scanned files: 8
Infected files: 0
Data scanned: 0.29 MB
Data read: 0.25 MB (ratio 1.17:1)
Time: 17.771 sec (0 m 17 s)
Start Date: 2021:05:13 11:43:54
End Date: 2021:05:13 11:44:11

ChkRootKit Root Kit Detection Scan

ChkRootKit is a commonly used application that scans a system for possible root kits.

ChkRootKit returned no Warning, meaning no Root Kits were found on your system.

Lynis Security Audit

Lynis is a security auditing tool that provides a detailed responce on a systems weaknesses and issues with the goal of hardening the system.

Result for Lynis Security Audit:
```
 -[ Lynis 2.6.2 Results  ]-

  Warnings (3):
  ----------------------------
  ! Version of Lynis is very old and should be updated [LYNIS]
    https://cisofy.com/controls/LYNIS/

  ! Couldn't find 2 responsive nameservers [NETW-2705]
    https://cisofy.com/controls/NETW-2705/

  ! Redis configuration file /etc/redis/redis-openvas.conf is world readable and might
```

leak sensitive details [DBS-1882]
   - Details : /etc/redis/redis-openvas.conf
   - Solution : Use chmod 640 to change file permissions
     https://cisofy.com/controls/DBS-1882/

  Suggestions (56):
  ----
  * Install libpam-tmpdir to set $TMP and $TMPDIR for PAM sessions [CUST-0280]
     https://your-domain.example.org/controls/CUST-0280/

  * Install libpam-usb to enable multi-factor authentication for PAM sessions [CUST-0285]
     https://your-domain.example.org/controls/CUST-0285/

  * Install apt-listbugs to display a list of critical bugs prior to each APT installation. [CUST-0810]
     https://your-domain.example.org/controls/CUST-0810/

  * Install apt-listchanges to display any significant changes prior to any upgrade via APT. [CUST-0811]
     https://your-domain.example.org/controls/CUST-0811/

  * Install debian-goodies so that you can run checkrestart after upgrades to determine which services are using old versions of libraries and need restarting. [CUST-0830]
     https://your-domain.example.org/controls/CUST-0830/

  * Install needrestart, alternatively to debian-goodies, so that you can run needrestart after upgrades to determine which daemons are using old versions of libraries and need restarting. [CUST-0831]
     https://your-domain.example.org/controls/CUST-0831/

  * Install debsecan to generate lists of vulnerabilities which affect this installation. [CUST-0870]
     https://your-domain.example.org/controls/CUST-0870/

  * Install debsums for the verification of installed package files against MD5 checksums. [CUST-0875]
     https://your-domain.example.org/controls/CUST-0875/

* Install fail2ban to automatically ban hosts that commit multiple authentication errors. [DEB-0880]
  https://cisofy.com/controls/DEB-0880/

* Set a password on GRUB bootloader to prevent altering boot configuration (e.g. boot in single user mode without password) [BOOT-5122]
  https://cisofy.com/controls/BOOT-5122/

* Protect rescue.service by using sulogin [BOOT-5260]
  https://cisofy.com/controls/BOOT-5260/

* Determine why /vmlinuz is missing on this Debian/Ubuntu system. [KRNL-5788]
  - Details : /vmlinuz
  https://cisofy.com/controls/KRNL-5788/

* Check the output of apt-cache policy manually to determine why output is empty [KRNL-5788]
  https://cisofy.com/controls/KRNL-5788/

* Install a PAM module for password strength testing like pam_cracklib or pam_passwdqc [AUTH-9262]
  https://cisofy.com/controls/AUTH-9262/

* Configure minimum password age in /etc/login.defs [AUTH-9286]
  https://cisofy.com/controls/AUTH-9286/

* Configure maximum password age in /etc/login.defs [AUTH-9286]
  https://cisofy.com/controls/AUTH-9286/

* Default umask in /etc/login.defs could be more strict like 027 [AUTH-9328]
  https://cisofy.com/controls/AUTH-9328/

* Default umask in /etc/init.d/rc could be more strict like 027 [AUTH-9328]
  https://cisofy.com/controls/AUTH-9328/

* To decrease the impact of a full /home file system, place /home on a separated partition [FILE-6310]

https://cisofy.com/controls/FILE-6310/

* To decrease the impact of a full /tmp file system, place /tmp on a separated partition [FILE-6310]
https://cisofy.com/controls/FILE-6310/

* To decrease the impact of a full /var file system, place /var on a separated partition [FILE-6310]
https://cisofy.com/controls/FILE-6310/

* Disable drivers like USB storage when not used, to prevent unauthorized storage or data theft [STRG-1840]
https://cisofy.com/controls/STRG-1840/

* Disable drivers like firewire storage when not used, to prevent unauthorized storage or data theft [STRG-1846]
https://cisofy.com/controls/STRG-1846/

* Check DNS configuration for the dns domain name [NAME-4028]
https://cisofy.com/controls/NAME-4028/

* Remove duplicate lines in /etc/hosts [NAME-4402]
https://cisofy.com/controls/NAME-4402/

* Check RPM database as RPM binary available but does not reveal any packages [PKGS-7308]
https://cisofy.com/controls/PKGS-7308/

* Install debsums utility for the verification of packages with known good database. [PKGS-7370]
https://cisofy.com/controls/PKGS-7370/

* Check your resolv.conf file and fill in a backup nameserver if possible [NETW-2705]
https://cisofy.com/controls/NETW-2705/

* Consider running ARP monitoring software (arpwatch,arpon) [NETW-3032]
https://cisofy.com/controls/NETW-3032/

* Access to CUPS configuration could be more strict. [PRNT-2307]
  https://cisofy.com/controls/PRNT-2307/

* Configure a firewall/packet filter to filter incoming and outgoing traffic [FIRE-4590]
  https://cisofy.com/controls/FIRE-4590/

* Consider hardening SSH configuration [SSH-7408]
 - Details : AllowTcpForwarding (YES --> NO)
  https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
 - Details : ClientAliveCountMax (3 --> 2)
  https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
 - Details : Compression (YES --> (DELAYED|NO))
  https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
 - Details : LogLevel (INFO --> VERBOSE)
  https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
 - Details : MaxAuthTries (6 --> 2)
  https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
 - Details : MaxSessions (10 --> 2)
  https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
 - Details : PermitRootLogin (WITHOUT-PASSWORD --> NO)
  https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
 - Details : Port (22 -->)
  https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
 - Details : TCPKeepAlive (YES --> NO)
   https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
 - Details : X11Forwarding (YES --> NO)
   https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
 - Details : AllowAgentForwarding (YES --> NO)
   https://cisofy.com/controls/SSH-7408/

* Configure the 'requirepass' setting for Redis [DBS-1884]
 - Details : /etc/redis/redis-openvas.conf
 - Solution : configure 'requirepass' setting in /etc/redis/redis-openvas.conf
   https://cisofy.com/controls/DBS-1884/

* Configure the 'requirepass' setting for Redis [DBS-1884]
 - Details : /etc/redis/redis.conf
 - Solution : configure 'requirepass' setting in /etc/redis/redis.conf
   https://cisofy.com/controls/DBS-1884/

* Use the 'rename-command CONFIG' setting for Redis [DBS-1886]
 - Details : /etc/redis/redis-openvas.conf
 - Solution : configure 'rename-command CONFIG' in /etc/redis/redis-openvas.conf
   https://cisofy.com/controls/DBS-1886/

* Use the 'rename-command CONFIG' setting for Redis [DBS-1886]
 - Details : /etc/redis/redis.conf
 - Solution : configure 'rename-command CONFIG' in /etc/redis/redis.conf
   https://cisofy.com/controls/DBS-1886/

* Check what deleted files are still in use and why. [LOGG-2190]
   https://cisofy.com/controls/LOGG-2190/

* Add a legal banner to /etc/issue, to warn unauthorized users [BANN-7126]
   https://cisofy.com/controls/BANN-7126/

* Enable process accounting [ACCT-9622]
  https://cisofy.com/controls/ACCT-9622/

* Enable sysstat to collect accounting (no results) [ACCT-9626]
  https://cisofy.com/controls/ACCT-9626/

* Enable auditd to collect audit information [ACCT-9628]
  https://cisofy.com/controls/ACCT-9628/

* Check ntpq peers output for unreliable ntp peers and correct/replace them [TIME-3120]
  https://cisofy.com/controls/TIME-3120/

* Install a file integrity tool to monitor changes to critical and sensitive files [FINT-4350]
  https://cisofy.com/controls/FINT-4350/

* Determine if automation tools are present for system management [TOOL-5002]
  https://cisofy.com/controls/TOOL-5002/

* One or more sysctl values differ from the scan profile and could be tweaked [KRNL-6000]
  - Solution : Change sysctl value or disable test (skip-test=KRNL-6000:<sysctl-key>)
  https://cisofy.com/controls/KRNL-6000/

* Harden compilers like restricting access to root user only [HRDN-7222]
  https://cisofy.com/controls/HRDN-7222/

8.3.2 Section 2: Text Dump

```
Result for whoami command: root
```

```
Result for nslookup command:
Server:         192.168.1.254
Address:        192.168.1.254#53

Non-authoritative answer:
Name:   www.dvwa.co.uk
Address: 185.199.110.153
Name:   www.dvwa.co.uk
Address: 185.199.108.153
Name:   www.dvwa.co.uk
Address: 185.199.109.153
Name:   www.dvwa.co.uk
Address: 185.199.111.153
```

```
Result for whois command:
Domain name:
        dvwa.co.uk

    Data validation:
        Nominet was able to match the registrant's name and address
        against a 3rd party data source on 24-Apr-2018

    Registrar:
        1&amp;amp;1 Ionos SE [Tag = 1AND1]
        URL: https://www.ionos.co.uk

    Relevant dates:
        Registered on: 30-Aug-2009
        Expiry date:  30-Aug-2021
        Last updated:  21-Jan-2021

    Registration status:
        Registered until expiry date.

    Name servers:
        ns1.digitalocean.com
        ns2.digitalocean.com
        ns3.digitalocean.com
```

```
Result for DMitry command:
```

```
HostIP:185.199.109.153
HostName:dvwa.co.uk

Gathered Netcraft information for dvwa.co.uk
------------------------------------

Retrieving Netcraft.com information for dvwa.co.uk
Netcraft.com Information gathered

Gathered Subdomain information for dvwa.co.uk
------------------------------------
Searching Google.com:80...
HostName:www.dvwa.co.uk
HostIP:185.199.111.153
Searching Altavista.com:80...
Found 1 possible subdomain(s) for host dvwa.co.uk, Searched 0 pages
containing 0 results

Gathered E-Mail information for dvwa.co.uk
------------------------------------
Searching Google.com:80...
Searching Altavista.com:80...
Found 0 E-Mail(s) for host dvwa.co.uk, Searched 0 pages containing 0 results


------------------------------------------------------------
Result for DNSWalk command:
Checking dvwa.co.uk.
FAIL: Zone transfer of dvwa.co.uk. from ns1.digitalocean.com failed: NOTIMP
FAIL: Zone transfer of dvwa.co.uk. from ns2.digitalocean.com failed: NOTIMP
FAIL: Zone transfer of dvwa.co.uk. from ns3.digitalocean.com failed: NOTIMP
BAD: All zone transfer attempts of dvwa.co.uk. failed!
------------------------------------------------------------
# Nmap 7.70 scan initiated Thu May 13 11:17:39 2021 as: nmap -oX -
-oN textdump.txt --append-output -sV 192.168.1.70/24
Nmap scan report for Irelia (192.168.1.70)
Host is up (-0.011s latency).
Not shown: 997 filtered ports
PORT     STATE SERVICE          VERSION
443/tcp  open  ssl/https        VMware Workstation SOAP API 14.1.1
903/tcp  open  ssl/vmware-auth  VMware Authentication Daemon 1.10 (Uses VNC, SOAP)
5357/tcp open  http             Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
MAC Address: FC:AA:14:6C:80:46 (Giga-byte Technology)
Service Info: OS: Windows; CPE: cpe:/o:vmware:Workstation:14.1.1,
cpe:/o:microsoft:windows

Nmap scan report for bthub (192.168.1.254)
Host is up (0.0017s latency).
```

```
Not shown: 844 closed ports, 148 filtered ports
PORT      STATE SERVICE     VERSION
53/tcp    open  domain      (generic dns response: REFUSED)
80/tcp    open  http        lighttpd
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: HOME)
443/tcp   open  ssl/http    lighttpd
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: HOME)
7911/tcp  open  omapi       ISC (BIND|DHCPD) OMAPI
8888/tcp  open  http-proxy  tinyproxy 1.8.4
49152/tcp open  upnp        Portable SDK for UPnP devices 1.6.18
(Linux 3.4.11-rt19; UPnP 1.0)
1 service unrecognized despite returning data. If you know the
service/version, please submit the following fingerprint at
https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port53-TCP:V=7.70%I=7%D=5/13%Time=609CFD72%P=x86_64-pc-linux-gnu%r(DNSV
SF:ersionBindReqTCP,2D,"\0\+\0\x06\x85\x80\0\x01\0\x01\0\0\0\0\x07version\
SF:x04bind\0\0\x10\0\x03\xc0\x0c\0\x10\0\x03\0\0\0\0\0\x01\0")%r(DNSStatus
SF:RequestTCP,E,"\0\x0c\0\0\x90\x85\0\0\0\0\0\0\0\0");
MAC Address: 80:20:DA:FF:BB:FB (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel:3.4.11-rt19

Nmap scan report for mx (192.168.1.98)
Host is up (0.000017s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE     VERSION
22/tcp    open  ssh         OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
111/tcp   open  rpcbind     2-4 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
6566/tcp open  tcpwrapped
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
# Nmap done at Thu May 13 11:23:15 2021 -- 256 IP addresses
(3 hosts up) scanned in 336.22 seconds
# Nmap 7.70 scan initiated Thu May 13 11:23:16 2021 as:
nmap -script vuln -O -oN textdump.txt --append-output 192.168.1.254
Pre-scan script results:
| broadcast-avahi-dos:
|   Discovered hosts:
|     224.0.0.251
|   After NULL UDP avahi packet DoS (CVE-2011-1002).
|_  Hosts are all up (not vulnerable).
Nmap scan report for bthub (192.168.1.254)
Host is up (0.0017s latency).
Not shown: 844 closed ports, 148 filtered ports
PORT      STATE SERVICE
```

```
53/tcp    open  domain
80/tcp    open  http
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-majordomo2-dir-traversal: ERROR: Script execution failed (use -d to debug)
|_http-passwd: ERROR: Script execution failed (use -d to debug)
| http-slowloris-check:
|   VULNERABLE:
|   Slowloris DOS attack
|     State: LIKELY VULNERABLE
|     IDs:  CVE:CVE-2007-6750
|       Slowloris tries to keep many connections to the target web
server open and hold
|       them open as long as possible.  It accomplishes this by
opening connections to
|       the target web server and sending a partial request.
By doing so, it starves
|       the http server's resources causing Denial Of Service.
|
|     Disclosure date: 2009-09-17
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6750
|_      http://ha.ckers.org/slowloris/
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
| http-vuln-cve2010-0738:
|_  /jmx-console/: Authentication was not required
|_http-vuln-cve2017-1001000: ERROR: Script execution failed (use -d to debug)
139/tcp   open  netbios-ssn
443/tcp   open  https
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-majordomo2-dir-traversal: ERROR: Script execution failed (use -d to debug)
|_http-passwd: ERROR: Script execution failed (use -d to debug)
| http-slowloris-check:
|   VULNERABLE:
|   Slowloris DOS attack
|     State: LIKELY VULNERABLE
|     IDs:  CVE:CVE-2007-6750
|       Slowloris tries to keep many connections to the target web
server open and hold
|       them open as long as possible.  It accomplishes this by
opening connections to
|       the target web server and sending a partial request. By
doing so, it starves
|       the http server's resources causing Denial Of Service.
|
|     Disclosure date: 2009-09-17
|     References:
```

```
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6750
|_      http://ha.ckers.org/slowloris/
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
| http-vuln-cve2010-0738:
|_  /jmx-console/: Authentication was not required
|_http-vuln-cve2017-1001000: ERROR: Script execution failed (use -d to debug)
| ssl-poodle:
|   VULNERABLE:
|   SSL POODLE information leak
|     State: LIKELY VULNERABLE
|     IDs:  OSVDB:113251  CVE:CVE-2014-3566
|           The SSL protocol 3.0, as used in OpenSSL through 1.0.1i and other
|           products, uses nondeterministic CBC padding, which makes it easier
|           for man-in-the-middle attackers to obtain cleartext data via a
|           padding-oracle attack, aka the "POODLE" issue.
|     Disclosure date: 2014-10-14
|     Check results:
|       TLS_RSA_WITH_AES_128_CBC_SHA
|       TLS_FALLBACK_SCSV properly implemented
|     References:
|       https://www.imperialviolet.org/2014/10/14/poodle.html
|       https://www.openssl.org/~bodo/ssl-poodle.pdf
|       http://osvdb.org/113251
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-3566
|_sslv2-drown:
445/tcp   open  microsoft-ds
7911/tcp  open  unknown
8888/tcp  open  sun-answerbook
49152/tcp open  unknown
MAC Address: 80:20:DA:FF:BB:FB (Unknown)
No exact OS matches for host (If you know what OS is running
on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.70%E=4%D=5/13%OT=53%CT=135%CU=31840%PV=Y%DS=1%DC=D%G=Y%M=8020DA
OS:%TM=609CFF1B%P=x86_64-pc-linux-gnu)SEQ(SP=101%GCD=1%ISR=10F%TI=Z%TS=U)OP
OS:S(O1=M5B4NNSNW5%O2=M5B4NNSNW5%O3=M5B4NW5%O4=M5B4NNSNW5%O5=M5B4NNSNW5%O6=
OS:M5B4NNS)WIN(W1=3908%W2=3908%W3=3908%W4=3908%W5=3908%W6=3908)ECN(R=Y%DF=Y
OS:%T=40%W=3908%O=M5B4NNSNW5%CC=N%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=
OS:)T2(R=N)T3(R=N)T4(R=N)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R
OS:=N)T7(R=N)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=
OS:G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 1 hop

Host script results:
|_samba-vuln-cve-2012-1182: Bind() returned an unexpected packet type
(not BIND_ACK)
|_smb-vuln-ms10-054: false
```

```
|_smb-vuln-ms10-061: Bind() returned an unexpected packet type (not BIND_ACK)

OS detection performed. Please report any incorrect results at https:
//nmap.org/submit/ .
# Nmap done at Thu May 13 11:27:39 2021 -- 1 IP address (1 host up)
scanned in 263.83 seconds
# Nmap 7.70 scan initiated Thu May 13 11:27:40 2021 as:
nmap -script vuln -O -oN textdump.txt --append-output 192.168.1.70
Pre-scan script results:
| broadcast-avahi-dos:
|   Discovered hosts:
|     224.0.0.251
|   After NULL UDP avahi packet DoS (CVE-2011-1002).
|_  Hosts are all up (not vulnerable).
Nmap scan report for Irelia (192.168.1.70)
Host is up (0.0026s latency).
Not shown: 997 filtered ports
PORT     STATE SERVICE
443/tcp  open  https
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-slowloris-check:
|   VULNERABLE:
|   Slowloris DOS attack
|     State: LIKELY VULNERABLE
|     IDs:  CVE:CVE-2007-6750
|       Slowloris tries to keep many connections to the target web
server open and hold
|       them open as long as possible.  It accomplishes this by
opening connections to
|       the target web server and sending a partial request. By doing
so, it starves
|       the http server's resources causing Denial Of Service.
|
|     Disclosure date: 2009-09-17
|     References:
|       http://ha.ckers.org/slowloris/
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6750
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
|_ssl-ccs-injection: No reply from server (TIMEOUT)
|_sslv2-drown:
903/tcp  open  iss-console-mgr
5357/tcp open  wsdapi
MAC Address: FC:AA:14:6C:80:46 (Giga-byte Technology)
Warning: OSScan results may be unreliable because we could not find
at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows 2008 (90%)
```

```
OS CPE: cpe:/o:microsoft:windows_server_2008::sp1 cpe:/o:
microsoft:windows_server_2008:r2
Aggressive OS guesses: Microsoft Windows Server 2008 SP1 or
Windows Server 2008 R2 (90%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
# Nmap done at Thu May 13 11:30:44 2021 -- 1 IP address
(1 host up) scanned in 185.12 seconds
# Nmap 7.70 scan initiated Thu May 13 11:30:45 2021 as:
nmap -script vuln -O -oN textdump.txt --append-output 192.168.1.98
Pre-scan script results:
| broadcast-avahi-dos:
|   Discovered hosts:
|     224.0.0.251
|   After NULL UDP avahi packet DoS (CVE-2011-1002).
|_  Hosts are all up (not vulnerable).
Nmap scan report for mx (192.168.1.98)
Host is up (0.000016s latency).
Not shown: 995 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
6566/tcp open  sane-port
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.32
OS details: Linux 2.6.32
Network Distance: 0 hops

Host script results:
|_smb-vuln-ms10-054: false
|_smb-vuln-ms10-061: false
| smb-vuln-regsvc-dos:
|   VULNERABLE:
|   Service regsvc in Microsoft Windows systems vulnerable
to denial of service
|     State: VULNERABLE
|       The service regsvc in Microsoft Windows 2000 systems
is vulnerable to denial of service caused by a null deference
|       pointer. This script will crash the service if it is
vulnerable. This vulnerability was discovered by Ron Bowes
|       while working on smb-enum-sessions.
|_
```

```
OS detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
# Nmap done at Thu May 13 11:31:28 2021 -- 1 IP address
(1 host up) scanned in 43.68 seconds
Result for Nikto Vulnerability Scan:
- Nikto v2.1.5
---------------------------------------------------------------------------
+ Target IP:          185.199.110.153
+ Target Hostname:    www.dvwa.co.uk
+ Target Port:        80
+ Start Time:         2021-05-13 11:31:29 (GMT1)
---------------------------------------------------------------------------
+ Server: GitHub.com
+ Retrieved via header: 1.1 varnish
+ Retrieved x-served-by header: cache-lcy19233-LCY
+ The anti-clickjacking X-Frame-Options header is not present.
+ Uncommon header 'x-fastly-request-id' found, with contents:
8ae85bc8ec0928bd430dbb3fc6643d7019b0e07f
+ Uncommon header 'x-cache-hits' found, with contents: 2
+ Uncommon header 'x-cache' found, with contents: HIT
+ Uncommon header 'x-served-by' found, with contents: cache-
lcy19233-LCY
+ Uncommon header 'x-github-request-id' found, with contents:
6248:187E:1599821:16A5F42:609D0001
+ Uncommon header 'x-timer' found, with contents: S1620901889
.452424,VS0,VE0
+ Root page / redirects to: https://dvwa.co.uk/
+ No CGI Directories found (use '-C all' to force check all
possible dirs)
+ Server leaks inodes via ETags, header found with file /,
fields: 0x5f7baea5 0x239b
+ Uncommon header 'content-security-policy' found, with contents:
default-src 'none'; style-src 'unsafe-inline'; img-src data:;
connect-src 'self'
+ Uncommon header 'x-xss-protection' found, with contents: 0
+ Uncommon header 'x-content-type-options' found, with
contents: nosniff
+ Uncommon header 'strict-transport-security' found, with
contents: max-age=31536000
+ Uncommon header 'x-frame-options' found, with contents: deny
+ Server banner has changed from 'GitHub.com' to 'Varnish'
which may suggest a WAF, load balancer or proxy is in place
+ Uncommon header 'x-origin-cache' found, with contents: HIT
+ 6544 items checked: 0 error(s) and 16 item(s) reported on remote host
+ End Time:           2021-05-13 11:43:53 (GMT1) (744 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

```
-------------------------------------------------------------------
Result for ClamScan Virus Detection:

Known viruses: 8528798
Engine version: 0.103.2
Scanned directories: 1
Scanned files: 8
Infected files: 0
Data scanned: 0.29 MB
Data read: 0.25 MB (ratio 1.17:1)
Time: 17.771 sec (0 m 17 s)
Start Date: 2021:05:13 11:43:54
End Date:   2021:05:13 11:44:11
-------------------------------------------------------------------
ChkRootKit returned no Warning, meaning no Root Kits were found on your system.
-------------------------------------------------------------------
Result for Lynis Security Audit:
 ]-

  Warnings (3):
  ----------------------------

  ! Version of Lynis is very old and should be updated [LYNIS]
      https://cisofy.com/controls/LYNIS/

  ! Couldn't find 2 responsive nameservers [NETW-2705]
      https://cisofy.com/controls/NETW-2705/

  ! Redis configuration file /etc/redis/redis-openvas.conf is
  world readable and might leak sensitive details [DBS-1882]
    - Details  : /etc/redis/redis-openvas.conf
    - Solution : Use chmod 640 to change file permissions
      https://cisofy.com/controls/DBS-1882/

  Suggestions (56):
  ----------------------------

  * Install libpam-tmpdir to set $TMP and $TMPDIR for PAM sessions
  [CUST-0280]
      https://your-domain.example.org/controls/CUST-0280/

  * Install libpam-usb to enable multi-factor authentication for
  PAM sessions [CUST-0285]
      https://your-domain.example.org/controls/CUST-0285/

  * Install apt-listbugs to display a list of critical bugs prior
  to each APT installation. [CUST-0810]
      https://your-domain.example.org/controls/CUST-0810/

  * Install apt-listchanges to display any significant changes
```

```
prior to any upgrade via APT. [CUST-0811]
    https://your-domain.example.org/controls/CUST-0811/

* Install debian-goodies so that you can run checkrestart after
upgrades to determine which services are using old versions of
libraries and need restarting. [CUST-0830]
    https://your-domain.example.org/controls/CUST-0830/

* Install needrestart, alternatively to debian-goodies, so
that you can run needrestart after upgrades to determine which
daemons are using old versions of libraries and need restarting.
[CUST-0831]
    https://your-domain.example.org/controls/CUST-0831/

* Install debsecan to generate lists of vulnerabilities which
affect this installation. [CUST-0870]
    https://your-domain.example.org/controls/CUST-0870/

* Install debsums for the verification of installed package
files against MD5 checksums. [CUST-0875]
    https://your-domain.example.org/controls/CUST-0875/

* Install fail2ban to automatically ban hosts that commit
multiple authentication errors. [DEB-0880]
    https://cisofy.com/controls/DEB-0880/

* Set a password on GRUB bootloader to prevent altering boot
configuration (e.g. boot in single user mode without password)
[BOOT-5122]
    https://cisofy.com/controls/BOOT-5122/

* Protect rescue.service by using sulogin [BOOT-5260]
    https://cisofy.com/controls/BOOT-5260/

* Determine why /vmlinuz is missing on this Debian/Ubuntu
system. [KRNL-5788]
  - Details  : /vmlinuz
    https://cisofy.com/controls/KRNL-5788/

* Check the output of apt-cache policy manually to determine
why output is empty [KRNL-5788]
    https://cisofy.com/controls/KRNL-5788/

* Install a PAM module for password strength testing like
pam_cracklib or pam_passwdqc [AUTH-9262]
    https://cisofy.com/controls/AUTH-9262/

* Configure minimum password age in /etc/login.defs [AUTH-9286]
```

```
    https://cisofy.com/controls/AUTH-9286/

* Configure maximum password age in /etc/login.defs [AUTH-9286]
    https://cisofy.com/controls/AUTH-9286/

* Default umask in /etc/login.defs could be more strict
like 027 [AUTH-9328]
    https://cisofy.com/controls/AUTH-9328/

* Default umask in /etc/init.d/rc could be more strict
like 027 [AUTH-9328]
    https://cisofy.com/controls/AUTH-9328/

* To decrease the impact of a full /home file system,
place /home on a separated partition [FILE-6310]
    https://cisofy.com/controls/FILE-6310/

* To decrease the impact of a full /tmp file system,
place /tmp on a separated partition [FILE-6310]
    https://cisofy.com/controls/FILE-6310/

* To decrease the impact of a full /var file system,
place /var on a separated partition [FILE-6310]
    https://cisofy.com/controls/FILE-6310/

* Disable drivers like USB storage when not used, to
prevent unauthorized storage or data theft [STRG-1840]
    https://cisofy.com/controls/STRG-1840/

* Disable drivers like firewire storage when not used,
to prevent unauthorized storage or data theft [STRG-1846]
    https://cisofy.com/controls/STRG-1846/

* Check DNS configuration for the dns domain name
[NAME-4028]
    https://cisofy.com/controls/NAME-4028/

* Remove duplicate lines in /etc/hosts [NAME-4402]
    https://cisofy.com/controls/NAME-4402/

* Check RPM database as RPM binary available but does
not reveal any packages [PKGS-7308]
    https://cisofy.com/controls/PKGS-7308/

* Install debsums utility for the verification of packages
with known good database. [PKGS-7370]
    https://cisofy.com/controls/PKGS-7370/
```

```
* Check your resolv.conf file and fill in a backup
nameserver if possible [NETW-2705]
    https://cisofy.com/controls/NETW-2705/

* Consider running ARP monitoring software (arpwatch,arpon)
[NETW-3032]
    https://cisofy.com/controls/NETW-3032/

* Access to CUPS configuration could be more strict.
[PRNT-2307]
    https://cisofy.com/controls/PRNT-2307/

* Configure a firewall/packet filter to filter incoming
and outgoing traffic [FIRE-4590]
    https://cisofy.com/controls/FIRE-4590/

* Consider hardening SSH configuration [SSH-7408]
  - Details  : AllowTcpForwarding (YES --> NO)
    https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
  - Details  : ClientAliveCountMax (3 --> 2)
    https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
  - Details  : Compression (YES --> (DELAYED|NO))
    https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
  - Details  : LogLevel (INFO --> VERBOSE)
    https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
  - Details  : MaxAuthTries (6 --> 2)
    https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
  - Details  : MaxSessions (10 --> 2)
    https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
  - Details  : PermitRootLogin (WITHOUT-PASSWORD --> NO)
    https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
  - Details  : Port (22 --> )
    https://cisofy.com/controls/SSH-7408/
```

```
* Consider hardening SSH configuration [SSH-7408]
  - Details  : TCPKeepAlive (YES --> NO)
    https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
  - Details  : X11Forwarding (YES --> NO)
    https://cisofy.com/controls/SSH-7408/

* Consider hardening SSH configuration [SSH-7408]
  - Details  : AllowAgentForwarding (YES --> NO)
    https://cisofy.com/controls/SSH-7408/

* Configure the 'requirepass' setting for Redis
[DBS-1884]
  - Details  : /etc/redis/redis-openvas.conf
  - Solution : configure 'requirepass' setting in
  /etc/redis/redis-openvas.conf
    https://cisofy.com/controls/DBS-1884/

* Configure the 'requirepass' setting for Redis [DBS-1884]
  - Details  : /etc/redis/redis.conf
  - Solution : configure 'requirepass' setting in
  /etc/redis/redis.conf
    https://cisofy.com/controls/DBS-1884/

* Use the 'rename-command CONFIG' setting for Redis
[DBS-1886]
  - Details  : /etc/redis/redis-openvas.conf
  - Solution : configure 'rename-command CONFIG' in
  /etc/redis/redis-openvas.conf
    https://cisofy.com/controls/DBS-1886/

* Use the 'rename-command CONFIG' setting for Redis [DBS-1886]
  - Details  : /etc/redis/redis.conf
  - Solution : configure 'rename-command CONFIG' in
  /etc/redis/redis.conf
    https://cisofy.com/controls/DBS-1886/

* Check what deleted files are still in use and why.
[LOGG-2190]
    https://cisofy.com/controls/LOGG-2190/

* Add a legal banner to /etc/issue, to warn unauthorized
users [BANN-7126]
    https://cisofy.com/controls/BANN-7126/

* Enable process accounting [ACCT-9622]
    https://cisofy.com/controls/ACCT-9622/
```

```
* Enable sysstat to collect accounting (no results)
[ACCT-9626]
    https://cisofy.com/controls/ACCT-9626/

* Enable auditd to collect audit information [ACCT-9628]
    https://cisofy.com/controls/ACCT-9628/

* Check ntpq peers output for unreliable ntp peers and correct/
replace them [TIME-3120]
    https://cisofy.com/controls/TIME-3120/

* Install a file integrity tool to monitor changes to critical
and sensitive files [FINT-4350]
    https://cisofy.com/controls/FINT-4350/

* Determine if automation tools are present for system management
[TOOL-5002]
    https://cisofy.com/controls/TOOL-5002/

* One or more sysctl values differ from the scan profile
and could be tweaked [KRNL-6000]
  - Solution : Change sysctl value or disable test
  (skip-test=KRNL-6000:<sysctl-key>)
    https://cisofy.com/controls/KRNL-6000/

* Harden compilers like restricting access to root user only
[HRDN-7222]
    https://cisofy.com/controls/HRDN-7222/
```

8.4 Appendix D - GDPR Research Data Management Form

Abertay University

GDPR Research Data Management
Data Sign Off Form

For undergraduate or postgraduate student projects supervised by an Abertay staff member.

This form MUST be included in the student's thesis/dissertation. Note that failure to do this will mean that the student's project cannot be assessed/examined.

Part 1: Supervisors to Complete

By signing this form, you are confirming that you have checked and verified your student's data according to the criteria stated below (e.g., raw data, completed questionnaires, superlab/Eprime output, transcriptions etc.)

Student Name:	Thomas MacKinnon		
Student Number:	1704872		
Lead Supervisor Name:	Ross Heenan		
Lead Supervisor Signature	R.Heenan		
Project title:	Developing a Penetration Testing Toolkit to aid in Global Security through Automation and Smart Reporting		
Study route:	PhD ☐	MbR ☐	MPhil ☐
	Undergraduate ☑	PhD by Publication ☐	

Part 2: Student to Complete

	Initial here to confirm 'Yes'
I confirm that I have handed over all manual records from my research project (e.g., consent forms, transcripts) to my supervisor for archiving/storage	T.M
I confirm that I have handed over all digital records from my research project (e.g., recordings, data files) to my supervisor for archiving/storage	T.M
I confirm that I no longer hold any digital records from my research project on any device other than the university network and the only data that I may retain is a copy of an anonymised data file(s) from my research	T.M
I understand that, for undergraduate projects, my supervisor may delete manual/digital records of data if there is no foreseeable use for that data (with the exception of consent forms, which should be retained for 10 years)	T.M

Student signature : **Thomas MacKinnon**

Date: **12/05/2021**