COMPUTING WITH PLYMOUTH UNIVERSITY

School of Computing and Mathematics

PUSL3119

# Computing Individual Project

BSc (Hons) Computer Security

Rathnayaka Rathnayaka

Javelin: Automated penetration testing tool

2022/2023

# Acknowledgement

I wish to express my deepest gratitude to my project supervisor, Mr. Chamindra Attanayake, for their invaluable guidance, patience, and unwavering support throughout this project. Their depth of knowledge and practical insights were instrumental in the successful completion of this work.

I am also thankful to all the faculty members in the faculty of computing at NSBM green university and the University of Plymouth for their help and support. Their teachings and encouragement provided the foundation for this project.

I would like to thank the Open Worldwide Application Security Project (OWASP), the services of which were pivotal in gathering data and conducting experiments that were crucial to this project.

I would also like to acknowledge my peers for their camaraderie, constructive criticism, and intellectual discussions which helped shape this project.

Lastly, my heartfelt appreciation goes to my family and friends, whose constant encouragement and moral support were invaluable. Their belief in my capabilities inspired me to push my boundaries and strive for excellence.

# Javelin:

## Automated penetration testing tool

R.M.R.M.L. Rathnayaka[1], Chamindra Attanayake[2]

[1]*University of Plymouth, Drake Circus, Plymouth PL4 8AA, United Kingdom*
*10747919@students.plymouth.ac.uk*

[2]*National School of Business Management, Mahenwaththa, Pitipana, Homagama, 10200, Sri Lanka*
*rutcattanayake@plymouth.ac.uk*

## ABSTRACT

In cybersecurity, efficiency in detecting and responding to threats saves not only time but also the integrity of the digital world. This emphasizes the importance of acting quickly and efficiently in the cybersecurity domain. By detecting and responding to cyber threats in a timely and efficient manner, organizations can minimize the potential damage caused by security breaches, protect sensitive data, and maintain trust in the digital ecosystem. Efficiency in cybersecurity is crucial for safeguarding this increasingly interconnected world.

In the ever-changing landscape of cyberspace, securing digital assets has become a top priority for both businesses and individuals. Cybersecurity experts must stay one step ahead of cyber criminals as they employ increasingly sophisticated techniques to exploit vulnerabilities.

Enter the world of automated penetration testing, a novel approach that combines the capabilities of cutting-edge artificial intelligence with the knowledge of cybersecurity experts. In this new era of cyber defence, these digital fortresses not only fortify themselves but also dynamically adapt to the onslaught of threats, ensuring robust testing and vulnerability assessment generating. Here it is discussed how manual penetration testing can be improved using automation to save time and effort while maximizing the quality of the vulnerability assessment. The output of the automation will take time, and to improve the accuracy of the result without human errors, automation will take a lot of time because of the large variety of vulnerability assessment techniques. To improve the efficiency of automation, this is where artificial intelligence comes in and makes automation more effective.

**KEYWORDS**: cybersecurity, exploit, vulnerabilities, automated penetration testing, automation, artificial intelligence

## Table of Contents

## Table of figures

# 1 Introduction

Penetration testing is an important line of defence in the rapidly evolving world of cybersecurity. It is a simulated cyberattack on a computer system, network, or web application to identify vulnerabilities that threat actors could exploit. Traditionally, this process has been carried out manually by skilled penetration testers who use their expertise to mimic cybercriminals' strategies. Despite its critical role in ensuring robust cybersecurity, manual penetration testing is fraught with difficulties that can jeopardize its effectiveness. These include the process's time-consuming nature, limited scalability, inconsistent results due to human error, costly expertise and resource requirements, and difficulties keeping up with the rapidly evolving threat landscape. Manual penetration testing is a labour-intensive process by definition. Each stage, from reconnaissance to track clearing, must be performed manually, significantly increasing the time required, especially when dealing with complex systems. Furthermore, manual testing has limited scalability. As the number of systems or applications to be tested grows, so do the resources required for thorough testing. Another major issue is inconsistency in results, as the quality and accuracy of testing are heavily dependent on the individual tester's skills and knowledge. Variations in vulnerability detection or risk assessments can result, making it difficult to establish a consistent security posture. Furthermore, human error can result in missed vulnerabilities or false positives, jeopardizing the effectiveness of the testing process. Furthermore, the cost of manual penetration testing can be prohibitively expensive for many organizations, particularly small businesses. Hiring skilled penetration testers, maintaining their training, and managing the manual testing process can put a strain on budgets. Finally, the rapidly evolving cyber threat landscape poses a significant challenge. Cybersecurity professionals must constantly update their knowledge of new vulnerabilities, tools, and techniques, which can be a time-consuming and difficult process.

The proposed solution uses automation and machine learning to streamline and improve cyber defence measures such as vulnerability management, patch management, incident detection and response, threat intelligence, and configuration management, among others. The automation of penetration testing, a critical cybersecurity process that identifies potential vulnerabilities in a system, network, or application, is a key component of this solution. This solution entails the creation of a machine-learning model capable of predicting potential vulnerabilities based on existing data. This model is then integrated into an automated penetration testing system that can identify and test potential vulnerabilities quickly and consistently. A user-friendly interface also allows users to configure the system, launch penetration tests, and interpret the results. This method provides numerous benefits, including faster and more consistent testing, increased coverage, reduced human error, and better resource allocation. However, for comprehensive coverage and robust security assessments, automation should be used in conjunction with manual testing.

# 2 Background

## 2.1 Problem Statements

Manual penetration testing is a critical step in identifying vulnerabilities in a system, network, or application. While manual penetration testing has its advantages, it also has several drawbacks, including,

1. Time-consuming
2. Limited scalability
3. Inconsistent results
4. Human error
5. Limited coverage
6. Costly
7. Lack of documentation
8. Bias
9. Difficulty in keeping up with evolving threats
10. Training and expertise

Manual penetration testing can be time-consuming because testers must perform each step of the process by hand. This can result in a long testing cycle, particularly for large and complex systems. Reconnaissance, scanning, gaining access, maintaining access, and clearing tracks are all part of manual penetration testing. Each stage necessitates manual execution, lengthening the overall process, especially when dealing with large networks and complex applications(Dalalana Bertoglio & Zorzo, 2017) The time and resources required for manual testing grow significantly as the number of systems, networks, or applications to be tested grows, making scaling difficult. As the number of systems, networks, or applications grows, allocating sufficient resources and time to perform thorough penetration testing becomes more difficult. Manual testing consumes a lot of resources and does not scale well with the expansion of the target infrastructure(Dalalana Bertoglio & Zorzo, 2017; Ghanem, n.d.).Because manual testing is dependent on the knowledge and skills of individual testers, the quality and accuracy of the results can vary. This inconsistency can make identifying all vulnerabilities and accurately assessing risk difficult. Because of differences in expertise, experience, and methodology, results may differ between testers. These inconsistencies can result in insufficient vulnerability detection or inconsistent risk assessments, making it difficult to establish a consistent security posture. During manual penetration testing, testers can make mistakes that lead to missed vulnerabilities or false positives. These errors can jeopardize the testing process's effectiveness. During manual testing, testers can make mistakes such as misconfiguring tools, overlooking vulnerabilities, or misinterpreting results. These mistakes can result in false positives, false negatives, or insufficient testing(Dalalana Bertoglio & Zorzo, 2017) Manual testers may lack the time and resources to test every possible attack vector, resulting in insufficient testing coverage. This can lead to unknown vulnerabilities and potential security risks. Manually testing every potential attack vector and vulnerability takes time and is often impractical. As a result, testers may focus on high-risk areas while overlooking less obvious flaws, leaving the system vulnerable to potential attacks. Hiring and retaining skilled penetration testers can be costly, particularly for small businesses and organizations with limited resources. Skilled penetration testers demand competitive salaries, which can put a strain on smaller organizations' budgets. Furthermore, ongoing training and certification costs can add to the financial burden. Manual testers may fail to thoroughly document their testing process and findings, making it difficult to track progress, evaluate testing effectiveness, and implement corrective actions. Manual testing may not result in standardized, detailed documentation of the testing process, vulnerabilities discovered, and remediation recommendations. This lack of documentation can make it difficult for an organization to track progress and implement effective security measures(Altulaihan et al., 2023; Tudosi et al., 2023) Personal biases or preferences may influence testing methodology, potentially focusing on specific vulnerabilities or attack vectors while overlooking others. Personal biases or preferences may cause testers to focus on particular vulnerabilities or tools while ignoring other potential attack vectors. This selective focus can lead to an insufficient security assessment. Manual testers may find it difficult to keep up with the latest tools, techniques, and vulnerabilities, which may reduce the effectiveness of their testing. Manual testers must keep up with the latest vulnerabilities, tools, and techniques as cyber threats evolve. This process of continuous learning can be difficult and time-consuming. Manual penetration testing necessitates a wide range of technical skills, including knowledge of network protocols, programming languages, operating systems, and security tools. Acquiring and maintaining these skills can be difficult for testers and organizations alike.

Despite these issues, manual penetration testing remains an important component of a comprehensive security testing strategy. Combining manual and automated testing can assist organizations in mitigating these issues and increasing the overall effectiveness of their penetration testing efforts. To address these issues, it is common to combine manual penetration testing with automated tools and processes, implement standardized testing methodologies, and invest in ongoing tester training and development. This method ensures a more thorough and effective security assessment. In conclusion, despite its importance in identifying vulnerabilities in systems, networks, and applications, manual penetration testing has several inherent challenges. Time-consuming processes, limited scalability, inconsistency, human errors, limited coverage, high costs, inadequate documentation, personal biases, difficulty keeping up with evolving threats, and the need for continuous training and expertise development are examples of these. Organizations should consider combining manual testing with automated tools and processes, implementing standardized methodologies, and investing in ongoing training for their security professionals to address these issues. Organizations can improve their security posture and better protect their assets from cyber threats by taking a comprehensive and balanced approach to penetration testing(Altulaihan et al., 2023; Dalalana Bertoglio & Zorzo, 2017; Tudosi et al., 2023) Manual penetration testing issues can have a significant impact on cyberspace, affecting the overall security of systems, networks, and applications. Among the potential consequences are,

1. Increased vulnerability
2. Inaccurate risk assessment
3. Slower incident response
4. Increased costs
5. Ineffective security strategies
6. Erosion of trust
7. Legal and regulatory consequences
8. The wider impact on cyberspace
9. credibility issues

Inadequate or inconsistent testing can allow vulnerabilities to go undetected, giving cybercriminals opportunities to exploit these flaws and compromise systems. Inconsistencies in testing results can lead to an incorrect assessment of the risk associated with identified vulnerabilities. This can lead to resource misallocation, with low-risk issues receiving unnecessary attention while high-risk issues go unaddressed. The financial burden of manual penetration testing, which includes salaries and training costs, can put a strain on organizational budgets, potentially limiting the frequency and depth of testing or forcing organizations to reduce other critical security measures. Human biases, limited coverage, and challenges in keeping up with evolving threats can all lead to the implementation of suboptimal security measures that may not fully address an organization's risks. If organizations are unable to provide a secure environment for their customers, partners, and employees, trust in their ability to protect sensitive data and systems may be eroded, potentially leading to reputational harm or business loss. Inadequate penetration testing may cause organizations to fail to meet industry standards, regulations, or legal requirements related to cybersecurity, exposing them to fines, penalties, or legal action. Vulnerabilities in one organization's systems that go undetected can have far-reaching consequences for the entire cyberspace ecosystem, as cybercriminals may use these flaws to launch attacks on other organizations or infrastructures(Altulaihan et al., 2023; Samant, 2011; Tudosi et al., 2023). Penetration testing, like any manual process, is prone to human error. Mistakes can occur at any stage of the process, from incorrectly configuring testing tools to failing to identify vulnerabilities or misinterpreting results. These errors can result in false positives or negatives, undermining the testing results' credibility. Personal biases or preferences of testers may influence their testing methodology. They may concentrate on specific types of vulnerabilities or employ specific tools, potentially ignoring other vulnerabilities. This can result in insufficient testing and raise concerns about the thoroughness and credibility of the results.

To mitigate the impact of these issues on cyberspace, organizations should take a comprehensive and balanced approach to penetration testing, combining manual testing with automated tools and processes, implementing standardized methodologies, and investing in ongoing security professional training and development. This will contribute to a more effective security assessment and a more secure and resilient cyberspace.

## 2.2 project scope

### *2.2.1 Penetration Testing Automation*

Penetration testing is the practice of simulating cyberattacks on a computer system, network, or web application to identify potential vulnerabilities. Traditionally, this was a largely manual process involving experienced security professionals attempting various methods of breaking into the system. However, the process can be time-consuming and heavily reliant on the tester's skill and experience. Automation can help to improve the process's efficiency and consistency. An automated penetration testing system can perform tests much faster than a human, and it can do so at any time of day or night, not just during business hours. It can also repeatedly run the same tests, ensuring that any changes in the system's security are detected.

First, the system must determine what to test. This could be a specific web application, a computer network, or another target. The system would identify potential targets based on predefined criteria. The target would then be analyzed by the machine learning model to predict potential vulnerabilities. This would entail investigating the target's operating system, installed software, configuration settings, and so on. The system would choose potential exploits to test based on the predicted vulnerabilities. These could be existing exploits linked to the predicted vulnerabilities or new exploits generated by the machine learning model. The system would then use the selected exploits to try to exploit the predicted vulnerabilities. This could include sending specially crafted data to the target, attempting to access restricted resources, or employing other techniques. Each exploit attempt would be collected and recorded by the system. This includes whether or not the exploit was successful, what data or resources were accessed, and any other pertinent information. If an exploit is successful, the system may also take corrective actions, depending on how it is configured. Patching the vulnerability, changing configuration settings, or alerting human administrators could all be part of this. The results of each penetration test would be fed back into the machine learning model, allowing it to improve over time by learning from its successes and failures.

### *2.2.2 Machine Learning Model Development and Training*

Machine learning is an artificial intelligence branch in which a model learns patterns from existing data and then applies those patterns to make predictions or decisions without being explicitly programmed to do so. A machine learning model could be trained on data about previous security vulnerabilities in the context of a project like this. This could include information about the vulnerability's characteristics, the type of system it was discovered in, and how it was exploited. The model would learn to recognize patterns and correlations in the data and then apply that knowledge to predict vulnerabilities in new, untested systems. Gather information about previous vulnerabilities. This could include data from internal security audits or scraping public vulnerability databases. Identify the relevant features in the data using feature extraction. In this case, features could include the type of system, software version, configuration settings, and any other factors that may influence the presence of a vulnerability. Train a model on the data using a machine learning algorithm. The model learns to associate the features with the presence or absence of a vulnerability during training. The algorithm used is determined by the task at hand as well as the nature of the data. Test the model on a separate set of data to ensure that it accurately predicts vulnerabilities. This is a necessary step to avoid overfitting, which occurs when a model learns the training data too well and performs poorly on new data. Once trained and validated, the model can be used to predict vulnerabilities in new, untested systems. This is accomplished by analyzing the system's features and predicting the likelihood of a vulnerability based on what it learned during training. If a vulnerability is predicted, the tool could use pre-defined exploitation techniques to determine whether or not the vulnerability is exploitable.

### *2.2.3 User Interface Development*

A user interface is an important part of most software because it allows users to interact with the system in a meaningful and efficient manner. The UI is the medium through which users configure the system, initiate penetration tests, and interpret the results for a tool like this. Users must be able to enter information about the system or application they wish to test. This could include your IP address, system type, software versions, and so on. A good user interface would have clear, intuitive fields or options for these inputs. Users may be able to customize the nature of the tests to be performed, depending on the sophistication of the tool. For example, they may be able to specify which types of vulnerabilities to look for or set a time limit for the test to run. Users must be able to initiate and terminate tests. A good user interface would have clear, prominent controls for these actions, as well as the ability to pause or adjust the intensity of the tests. The UI should provide some indication of what is going on during the tests. This could be a progress bar, a count of the number of tests run, or a live log of the actions taken. When the tests are finished, the UI should display the results in a clear and meaningful manner. This could include a list of vulnerabilities discovered, information about the tests that discovered them, and possibly even recommendations for how to fix them. A good user interface will also provide assistance and documentation, guiding the user through the process and explaining any technical terms or concepts.

## 2.3 objectives & deliverables

### *2.3.1 objectives*

1. develop a tool that can automate the exploitation process in penetration testing.
2. integrate the framework with the Metasploit Framework, a widely used penetration testing tool, for seamless identification and exploitation of vulnerabilities.
3. improve the efficiency and accuracy of vulnerability exploitation by learning from past experiences and continuously refining the machine learning models
4. to reduce the time and effort required by security professionals during penetration tests by automating repetitive tasks.
5. demonstrate the effectiveness of machine learning in cybersecurity and encourage further research and development in this area.

This is a one-of-a-kind and driven project that aims to use machine learning to revolutionize the field of penetration testing and vulnerability assessment. The project's goals are to address several challenges that security professionals face daily, as well as to demonstrate the potential of machine learning in cybersecurity. We will delve into the technical aspects of the project's objectives and their meaningful implications in this detailed explanation. This tool uses advanced machine learning techniques, such as deep learning and reinforcement learning, to create models that can predict the likelihood of successful exploitation for specific vulnerabilities. Security professionals can focus on more complex tasks and strategic decision-making by automating the exploitation process. The tool's machine learning models are trained using historical data from previous exploitation attempts, allowing the framework to intelligently select the most effective exploits and payloads for a given target system. This project integrates seamlessly with the Metasploit Framework, a well-known and comprehensive penetration testing tool. This integration is accomplished through the use of Metasploit's robust database of exploits and payloads, which allows the tool to automatically test and exploit identified vulnerabilities in target systems. This tool's framework, when used in conjunction with Metasploit, has access to a large collection of exploits and payloads, making it a powerful tool for security professionals. One of the most important aspects of this project is its ability to learn from previous experiences and improve over time. The framework's machine learning models are designed to be continuously refined based on new data and feedback, resulting in more efficient and accurate vulnerability identification and exploitation. This continuous learning process allows the framework to adapt to the ever-changing cybersecurity landscape, in which new vulnerabilities are constantly discovered and existing ones are patched. This tool aims to save security professionals time and effort by automating many of the repetitive tasks involved in penetration testing, such as scanning for vulnerabilities, selecting exploits, and testing payloads. This allows them to devote more resources to tasks that require human expertise and intuition, such as analyzing complex attack scenarios, devising defence strategies, and assessing an organization's overall security posture. This project exemplifies how machine learning can be used to solve real-world cybersecurity problems like vulnerability exploitation. The project encourages further research and development in this area by demonstrating the potential benefits and effectiveness of machine learning-based solutions, potentially leading to new and innovative applications of machine learning in cybersecurity. This project relies on a variety of technical components and methodologies to achieve these goals, Python scripts are used to implement the main framework as well as the machine learning algorithms. Using pre-trained machine learning models to predict the success rate of exploiting specific vulnerabilities, which aids in the prioritization of exploits and payloads. Integrating with the Metasploit Framework to gain access to its vast database of exploits and payloads, thereby increasing the framework's capabilities and effectiveness. Providing configuration files that allow for the customization of the framework's behaviour and settings, tailoring it to the specific needs and testing scenarios of users. By providing datasets for training and testing machine learning models, the framework can learn and improve its performance over time. Including libraries and helper scripts to aid the main framework's operation and ensure a modular, organized structure. Creating log files to track the progress of the framework and assist users in analyzing the results and identifying areas for improvement. Incorporating additional tools and utilities to supplement the main framework, thereby improving the overall penetration testing process. This is a project that aims to use machine learning to transform penetration testing and vulnerability assessment. The framework aims to provide a more efficient and accurate solution for security professionals by automating many of the repetitive tasks involved in the process and continuously refining its performance through learning.

### *2.3.2 primary deliverables*

The primary deliverables of this project are intended to give users a comprehensive and adaptable framework for automating the process of identifying and exploiting vulnerabilities in target systems. By dissecting each deliverable, we can gain a better understanding of its technical aspects as well as the implications for security professionals.

- Python-based tool for automated pentest
  secondary deliverables:
  1. Pre-trained machine learning models
  2. Configuration files
  3. Datasets for training and testing
  4. Detailed documentation and guides
  5. Libraries and helper scripts
  6. Log files

main.py is the project's central component, serving as the framework's core. It is a Python script that controls the entire exploitation process, from scanning target systems to launching specific exploits and payloads. The script employs machine learning algorithms and pre-trained models to select the most effective exploits and payloads based on their predicted success rates. DeepExploit.py significantly reduces the time and effort required for manual vulnerability assessment and exploitation by automating this process-trained machine learning model in the project to predict the success rate of exploiting specific vulnerabilities. These models allow the framework to prioritize which exploits and payloads to use, reducing trial and error significantly. With more data, the models can be retrained to improve their accuracy and performance, allowing the tool to become more efficient and effective over time. This adaptability is critical for staying current with the ever-changing landscape of cybersecurity threats and vulnerabilities. The tool includes several configuration files that allow users to modify the framework's behaviour and settings. Target systems, exploit modules, payloads, and other preferences are all defined in these files. Users can tailor the framework to their specific needs and testing scenarios by adjusting these settings, giving them flexibility and control over the testing process. The datasets used for training and testing the machine learning models are included in the project. These datasets contain information about previous attempts to exploit vulnerabilities, which the models use to learn and predict. The datasets can be updated as new vulnerabilities are discovered and exploited, allowing the models to be continuously retrained and refined. This ongoing learning process allows the tool to stay current and effective in identifying and exploiting vulnerabilities. The tool provides extensive documentation and guides to assist users in understanding how to use the framework effectively. These resources, which can be found in the "doc" directory, cover topics like installing the framework, customizing the configuration files, using the various tools and utilities, and interpreting the results. By providing extensive documentation, users can make the most of the framework and its features. During its operation, the tool generates log files, which can be found in the "logs" directory. These files contain a detailed record of the framework's actions and results, allowing users to track progress, assess the effectiveness of machine learning models, and identify areas for improvement. The availability of log files enables users to optimize their testing processes and make informed security decisions.

### *2.3.3 automation on cyber security*

Automation can help to mitigate a variety of cybersecurity issues by increasing the efficiency, accuracy, and effectiveness of security measures. Here are some examples of how automation can assist in addressing common cybersecurity challenges:

1. Vulnerability management
2. Patch management
3. Incident detection and response
4. Threat intelligence
5. Security orchestration and automation response
6. Configuration management
7. Continuous monitoring
8. Access control
9. Security testing
10. Compliance monitoring

Automated vulnerability scanners can identify and assess vulnerabilities in systems, networks, and applications faster and more consistently than manual methods, allowing organizations to better prioritize and address security risks(Samant, 2011). Patch management can be streamlined with automation, ensuring that security updates are applied consistently and promptly across the entire infrastructure. This narrows the window for attackers to exploit known vulnerabilities. Intrusion detection systems, security information and event management systems, and endpoint detection and response solutions are examples of automated security tools that can monitor and analyze network traffic, log data, and system activities in real-time. This allows for faster detection and response to security incidents, reducing potential damage. Automated threat intelligence platforms collect, analyze, and share data on emerging threats and vulnerabilities, ensuring that security teams are aware of the most recent threats and can adjust their defences accordingly. Security orchestration and automation response platforms can automate and streamline incident response processes, saving time and effort in investigating, remediating, and reporting security incidents. Automation can help an organization's infrastructure enforce consistent security configurations, reducing the risk of misconfigurations that can lead to security vulnerabilities. Automated tools can monitor and assess an organization's security posture in real-time, providing real-time feedback and enabling proactive security measures. Automation can help to simplify and strengthen access control processes, ensuring that only authorized individuals have access to sensitive resources and that access is quickly revoked when it is no longer needed. As previously discussed, automation can improve penetration testing efforts by increasing speed, scalability, and consistency, allowing organizations to more effectively identify and address security issues. Compliance monitoring tools that are automated can assist organizations in maintaining compliance with industry regulations and standards, lowering the risk of fines, penalties, and reputational damage.

While automation can significantly improve cybersecurity defences, it should be noted that it is not a cure-all. To ensure comprehensive protection against cyber threats, organizations should implement a layered security approach that combines automation with manual efforts, as well as foster a culture of security awareness among employees(Ghanem, n.d.).

# 3 Literature review

## 3.1 Automated penetration testing

The use of automated tools and techniques to simulate attacks on a system or network to identify vulnerabilities and assess their impact is known as automated penetration testing (APT). APT is becoming more popular because it has several advantages over manual penetration testing, such as increased speed, accuracy, and consistency. Automated tools can perform penetration tests much more quickly than manual methods, reducing the amount of time systems are vulnerable to attack. Furthermore, automated tools are less prone to errors and oversights than manual methods, which can aid in identifying vulnerabilities that human testers may overlook. The use of automated tools also ensures that tests are performed consistently and repeatably, making it possible to compare results over time. Furthermore, automated penetration testing can help reduce penetration testing costs by eliminating the need for human testers, who can be costly to hire and train. It is important to note, however, that automated tools have limitations, such as the inability to replicate specific types of attacks and the requirement for human expertise to interpret and respond to test results. As a result, automated penetration testing should be used in conjunction with other methods and performed by skilled professionals who are aware of the tools' limitations and potential risks(Ghanem, n.d.; Samant, 2011).

## 3.2 Machine Learning in Cybersecurity

Machine learning is becoming a more important tool in cybersecurity. Anomalies in network traffic and system logs that may indicate a cyber-attack can be detected using machine learning algorithms. These algorithms can learn to recognize patterns of normal behaviour and flag any deviations as potential threats. Rather than relying on signatures, machine learning can be used to identify and classify malware based on its behaviour. This improves the accuracy and effectiveness of detecting new and unknown malware. Machine learning algorithms can be used to analyze user behaviour and identify anomalies that could indicate insider threats or compromised accounts. Machine learning can be used to analyze large amounts of threat intelligence data to identify patterns and trends that could indicate new or emerging threats. Machine learning can be used to automate the response to cyber-attacks, making incident response faster and more effective.

## 3.3 Deep Learning Techniques for Cybersecurity

Deep learning techniques are being used more frequently in cybersecurity to detect, prevent, and respond to cyber threats. By analyzing malware's behaviour and structure, deep learning techniques can be used to detect it. This can be accomplished through the use of techniques such as convolutional neural networks or long short-term memory networks, which can learn to recognize patterns in malware code or detect malicious behaviour. Deep learning techniques can be used to detect network traffic anomalies that may indicate an attack. This can be accomplished through the use of techniques such as recurrent neural networks or autoencoders, which can learn to recognize normal patterns in network traffic and flag any deviations as potential threats. Deep learning techniques can be used to analyze user behaviour and spot any anomalies that could indicate insider threats or compromised accounts. This can be accomplished through the use of techniques like reinforcement learning, which can learn to recognize patterns of normal behaviour and flag any deviations as potential threats. Deep learning techniques can be used to analyze large amounts of threat intelligence data to identify patterns and trends that could indicate new or emerging threats. This can be accomplished through the use of techniques like convolutional neural networks or long short-term memory networks, which can learn to recognize patterns in threat intelligence data. By analyzing code or system logs, deep-learning techniques can be used to identify potential vulnerabilities in software or systems. This can be accomplished by employing techniques like convolutional neural networks or long short-term memory networks, which can learn to recognize patterns in code or system logs that may indicate vulnerabilities(Nikolov & Slavyanov, 2017).

## 3.4 Vulnerability Detection and Exploitation

Vulnerability detection and exploitation are two critical cybersecurity processes used to identify and remediate vulnerabilities in a system or network. Vulnerability detection is the process of identifying flaws in software, hardware, and network infrastructure by using tools and techniques. This process may include vulnerability scanning, vulnerability assessment, and penetration testing, which are all intended to identify and quantify the potential risks associated with specific vulnerabilities. Once identified, vulnerabilities can be prioritized based on severity and remediated using patches, configuration changes, or other mitigation strategies.

Exploitation, on the other hand, is the process of exploiting identified vulnerabilities to gain unauthorized access to a system or network. This can include the use of exploit code, which is intended to exploit specific vulnerabilities to gain access to sensitive data or execute malicious code. While the goal of vulnerability detection is to identify and remediate vulnerabilities before they can be exploited, the exploitation process can be useful in understanding the potential impact of a vulnerability and assessing the effectiveness of mitigation strategies.

Both vulnerability detection and exploitation are critical processes in cybersecurity, and they are frequently used in tandem to provide a comprehensive understanding of a system's security posture. Organizations can reduce the risk of successful cyberattacks and protect sensitive data by identifying and correcting vulnerabilities(Altulaihan et al., 2023; Ghanem, n.d.).

## 3.5 Ethical and Legal Considerations

To ensure that security measures are implemented responsibly and legally, ethical and legal considerations are critical in cybersecurity. Individuals' privacy must be protected, as well as their rights to access and control their data, according to ethical considerations. This includes handling data breaches responsibly and notifying affected individuals on time. It also entails maintaining the confidentiality of sensitive information and ensuring that only authorized individuals have access to it. Compliance with applicable laws and regulations, such as data protection laws and industry-specific regulations, is a legal consideration. This includes making certain that personal data is collected and processed following applicable laws, and that appropriate safeguards are in place to prevent unauthorized access or disclosure. It also entails ensuring that systems and processes adhere to relevant cybersecurity regulations, such as those governing the use of encryption, access controls, and incident response. Furthermore, ethical and legal considerations include using cybersecurity tools and techniques responsibly, such as penetration testing and vulnerability scanning. These tools must be used in a way that respects individuals' and organizations' rights while causing no harm or disruption. This includes obtaining appropriate authorization and consent, implementing appropriate safeguards to prevent unauthorized access or disclosure, and disposing of data and equipment properly(Reddy Mamilla, n.d.).

# 4 Method of approach

## 4.1 Planning phase

The problems that the project will address should be clearly defined at this stage. The project's scope could include automating penetration testing and vulnerability discovery. This phase would entail a deep dive into existing penetration testing methods as well as the current state-of-the-art in machine learning as it relates to cybersecurity. Define the metrics that will be used to assess the project's success. For this project, this could include things like the number of successfully discovered vulnerabilities or the time it took to find them and identify the potential risks and challenges that may arise during the project's execution.

## 4.2 Tool Design and Development

The first step is to determine what the tool must do. This includes determining which systems or networks the tool will be used on, which vulnerabilities it must detect, and which exploits it must be capable of executing. The tool is built with a mix of programming languages and libraries. Python, for example, is a popular choice due to its powerful networking and machine-learning libraries (such as Scikit-learn, TensorFlow, or PyTorch). Because of its extensive networking capabilities and efficiency, C++ is also used. The tool's architecture must be designed. This includes how it will communicate with the network or system, process data, and execute exploits. If the tool employs machine learning, the architecture of the machine learning model must also be created. The tool will then be implemented following the design. This entails writing code for the tool, integrating it with required libraries, and ensuring that it can successfully interface with a network or system. The tool should be integrated with known vulnerabilities and exploit databases, such as Metasploit. This enables the tool to detect these vulnerabilities in the system or network being examined.

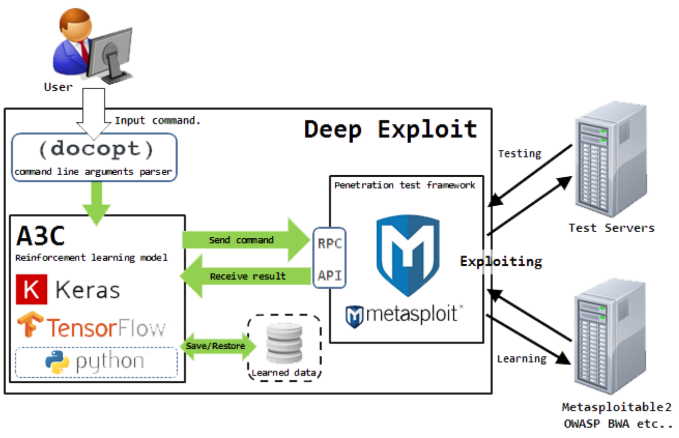


*Figure 1 - Final data flow chart*

To ensure that the tool works as expected, it should be thoroughly tested. This could entail running it on a controlled system or network to ensure that it correctly identifies vulnerabilities and executes exploits. Any problems or bugs discovered during testing should be resolved. If the tool utilizes machine learning, the model must be trained on a set of exploits. This entails feeding the model data on various exploits and their outcomes, allowing it to learn to predict the success of a given exploit based on its characteristics and those of the target system.

Once the tool has been thoroughly tested and refined, as well as the machine learning model (if used), it is ready to be used for automated penetration testing.

## 4.3 Penetration Test Execution

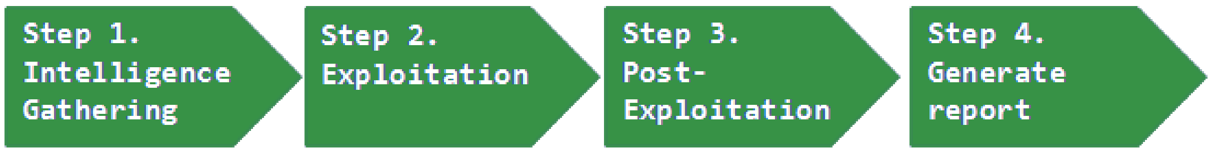


*Figure 2 - Penetration test flow*

The tool is used in a supervised environment. This could be a separate test network or system from the live environment. This controlled setup helps to avoid inadvertent network or system damage or disruption.

The tool first scans the network or system. This entails identifying the network devices or system components, as well as the software that runs on them. The tool may collect information such as IP addresses, operating systems, open ports, running applications and services, and so on. The tool then identifies potential vulnerabilities based on the data gathered during the scanning phase. This is accomplished by comparing the gathered data to known vulnerabilities in its database (for example, vulnerabilities listed in Metasploit). It searches for similarities between the system's characteristics and the conditions that allow certain vulnerabilities to be exploited. After identifying potential vulnerabilities, the tool selects appropriate exploits to test these vulnerabilities. If the tool employs machine learning, it may predict the success rate of various exploits based on its training and then select the most promising. The tool then tries to carry out these exploits.

The tool records the outcomes of its actions throughout the testing process. This includes the vulnerabilities it discovered, the exploits it tried, and the results of those attempts. These results are critical for later analysis and providing a test report. After the test is finished, the tool should clean up if necessary and possible. This could imply removing any payloads installed as part of the test on the system or network.

Finally, the tool generates a penetration test report. This report should include detailed information about the discovered vulnerabilities, attempted exploits, success rates, and any other pertinent observations. This report is then used to conduct additional analysis and to guide remediation efforts.

## 4.4 Results Analysis and Refinement

After the penetration test, the results are thoroughly examined. This entails reviewing the tool's report, which should include information about the identified vulnerabilities, attempted exploits, success rates, and any other relevant observations. The goal is to evaluate the tool's effectiveness - how well it identified vulnerabilities, how well it exploited them, and whether it missed any vulnerabilities. Any flaws or areas for improvement are identified during the analysis. For example, the tool could have overlooked certain vulnerabilities or failed to exploit identified vulnerabilities. Other potential flaws include inefficiencies in the scanning process, errors in the tool's reports, and interface issues. Once the weaknesses are identified, the tool is refined accordingly. Depending on the identified flaws, this could entail a variety of activities.

- If the tool employs machine learning and its predictions were not sufficiently accurate, it may require additional training. This could entail collecting more training data, adjusting the machine learning model's parameters, or employing a different machine learning algorithm.
- The scanning algorithm may need to be fine-tuned if the scanning process was inefficient or inaccurate. Making it more thorough, adjusting it to reduce false positives or negatives, or optimizing it for speed are all possibilities.
- If the tool lacked an appropriate exploit for a particular vulnerability, additional exploits may need to be added. This could entail adding these exploits to the tool's database and ensuring the tool can correctly execute them.
- If there were problems with the tool's interface or reports, these aspects of the tool may need to be improved. Making the interface more user-friendly, adding more detailed explanations to the reports, or fixing any errors in the reports could all fall under this category.

The tool should be tested again after it has been refined to ensure that the refinements were effective. This testing, analyzing, and refining process should be repeated until the tool performs satisfactorily.

## 4.5 Limitations and Future Work

### *4.5.1 Acknowledging Limitations*

It is critical to recognize and disclose the tool's limitations. One common limitation is that the tool's effectiveness is heavily reliant on the completeness and accuracy of the vulnerability and exploit databases it employs. As a result, if the databases are out of date, the tool may fail to detect new vulnerabilities or provide appropriate exploits for them. Another limitation is that the tool cannot handle unknown or "zero-day" vulnerabilities because it relies on known vulnerabilities in its databases. The tool may also be less effective in complex or highly customized situations. Finally, if the tool employs machine learning, it may be limited by the algorithm or model used, such as overfitting or underfitting, lack of interpretability, or sensitivity to the quality and quantity of training data.

### 4.5.2 Proposing Future Work

Updating the vulnerability and exploiting databases regularly can help the tool stay current with the latest security threats. It may also be advantageous to integrate additional databases to broaden the range of vulnerabilities and exploits that the tool can handle. Anomaly detection and behavioural analysis techniques could be integrated into the tool to better handle unknown or zero-day vulnerabilities. It could be designed to be more customizable or adaptable to improve the tool's effectiveness in different or complex environments. It could, for example, allow users to input specific information about their environment or use reinforcement learning techniques to gradually adapt to new environments. If the tool makes use of machine learning, future work could concentrate on improving the machine learning model. This could include experimenting with different algorithms, adjusting parameters, or increasing the quality or quantity of training data. Efforts could also be made to improve the predictability of the model's predictions, making the tool more reliable and useful. To keep up with the ever-changing landscape of cybersecurity threats, it is critical to continuously improve and adapt the tool.

# 5 Requirements

## 5.1 Operating System: Kali Linux

Kali Linux is regarded as essential for automated penetration testing programs due to its comprehensive suite of tools designed specifically for penetration testing and ethical hacking. Automated penetration testing programs frequently necessitate the use of a diverse set of tools to perform tasks such as vulnerability scanning, network mapping, password cracking, and exploit development. Kali Linux includes a large number of pre-installed and pre-configured tools, making it simple for users to begin performing various penetration testing tasks. Furthermore, Kali Linux is a Linux distribution based on Debian that is specifically designed for penetration testing and digital forensics. Offensive Security, a leading provider of information security training and certification, maintains and develops it. Kali Linux includes several features that are important for automated penetration testing programs, such as the ability to run from a USB drive or DVD, which is useful when testing on different machines. Kali Linux also has extensive documentation and a large online community to assist users with any problems they may encounter while testing. Overall, Kali Linux is an important tool for automated penetration testing programs because it provides a comprehensive suite of tools, is specifically designed for penetration testing, and has features important for automated testing programs.

## 5.2 Python

Python, because of its simplicity, flexibility, and versatility, is an essential programming language for automated penetration testing tools. Python's readability and ease of use make it a popular choice for penetration testers looking to quickly create scripts and tools to automate their testing procedures. Python is a high-level language that is simple to read and write, allowing penetration testers to quickly develop custom scripts and tools. Its syntax is concise and straightforward, making it an excellent choice for task automation. Python is a versatile programming language that can be used for a wide range of tasks including web development, data analysis, and automation. Its extensive library of modules and packages offers penetration testers a wide range of tools and functionalities for developing custom scripts and automating testing procedures. Python is a cross-platform language, which means that scripts written on one platform can be run on another with little modification. This is especially important for penetration testers who must test multiple operating systems and platforms. Python is easily integrated with other penetration testing tools and frameworks such as Nmap, Metasploit, and Shodan. This makes it simple to combine various tools to create a more powerful and automated testing environment. Python is an open-source language, which means it can be customized and extended to meet the unique needs of penetration testers.

## 5.3 Dependencies

### *5.3.1 TensorFlow*

TensorFlow can aid in automation by providing tools and techniques for developing machine-learning models capable of performing complex tasks with minimal human intervention. TensorFlow can automate a wide range of tasks that would otherwise require significant manual effort, thanks to its capabilities for predictive modelling, image recognition, natural language processing, and more. TensorFlow, for example, can automate decision-making processes such as detecting potential fraud in financial transactions or predicting equipment failures in manufacturing. TensorFlow image recognition can automate tasks like quality control in manufacturing or identifying objects in images, reducing the need for manual inspection. TensorFlow's natural language processing can automate tasks like sentiment analysis, language translation, and chatbots.

### *5.3.2 Nmap*

Because it is a powerful and versatile network scanning tool that provides information about network hosts and services, Nmap is an essential tool for automated penetration testing programs. Nmap is an essential tool for automated penetration testing programs for the following reasons.

Nmap can scan large networks and identify active hosts, allowing penetration testers to focus their efforts on active hosts. Nmap can detect open ports on a host and provide useful information about the services that are running on that host. This can aid penetration testers in identifying flaws and potential attack vectors. Nmap can determine the version of services running on a host, which allows penetration testers to determine whether those services are vulnerable to known exploits. Nmap can detect the operating system on a host, providing useful information for further testing and exploitation. Nmap integrates easily into automated penetration testing frameworks, allowing testers to automate the scanning process and concentrate on analyzing the results.

### *5.3.3 Metasploit*

Because it provides a wide range of capabilities for identifying, testing, and exploiting vulnerabilities in systems and applications, Metasploit is an essential tool for automated penetration testing programs. Here are a few reasons why Metasploit is such an important tool for automated penetration testing programs.

Metasploit provides a platform for creating and testing custom exploits, allowing penetration testers to identify and exploit vulnerabilities that other tools may miss. Metasploit integrates easily into automated testing frameworks, enabling testers to automate the process of identifying and exploiting vulnerabilities. Metasploit offers a variety of payloads that can be used to gain remote access to systems and execute commands, making it an important tool for testing and demonstrating the impact of vulnerabilities. Metasploit offers a diverse set of modules for testing and exploiting vulnerabilities in a wide range of systems and applications, including web applications, databases, and operating systems. Metasploit includes reporting capabilities that enable testers to generate detailed reports on vulnerabilities discovered and actions taken to exploit them.

### 5.4 Exploit Database

This tool uses a large exploit database to find suitable exploits for target systems. To keep the tool up to date with the latest vulnerabilities and exploits, the database must be updated regularly.

### 5.5 Hardware Requirements

Deep learning models can be computationally demanding to train and run. It is recommended that you have a powerful CPU, sufficient RAM, and, if possible, a GPU with CUDA support, which can significantly accelerate model training. A powerful CPU is required for deep learning. Look for a processor with a high clock speed, multiple cores, and multithreading support. Intel Core i7, i9, or Xeon processors are excellent options. Deep learning models can consume a lot of memory, so having enough RAM is critical. A minimum of 16 GB of RAM is recommended, with 32 GB or more recommended for larger models. A GPU can significantly speed up model training, especially if it supports CUDA, NVIDIA's parallel computing platform. Look for a GPU with at least 8 GB of memory, such as an NVIDIA GeForce RTX or NVIDIA Quadro card.

# 6 Solution

To address the challenges of manual penetration testing, consider leveraging automation to streamline and enhance the process. Because they quickly identify vulnerabilities and assess risks, automated tools can help manage time and resource constraints. By incorporating automation into your workflow, you can maintain a consistent testing methodology and more effectively prioritize high-risk areas. Furthermore, automated solutions can help you stay up to date on emerging threats while freeing up time for your skilled professionals to focus on complex, manual testing tasks that require human intuition and expertise. While automation can greatly improve the process, it should be used in tandem with manual testing to ensure comprehensive coverage and robust security assessments.

Automation refers to the use of technology, such as software and machines, to perform tasks with minimal human intervention. Automation in the context of penetration testing can help make the process easier and more efficient by automating repetitive tasks, reducing human error, and increasing testing speed.

Many of the issues associated with manual testing can be mitigated by automating penetration testing. Here are some examples of how automation can help with these issues:

1. Faster testing
2. Scalability
3. Consistent results
4. Reduced human error
5. Improved coverage
6. Cost-effectiveness
7. Better documentation
8. Continuous learning

Automated tools can perform repetitive tasks and scan large networks or applications faster than human testers, reducing testing time and allowing for more frequent assessments(Ghanem, n.d.; Nikolov & Slavyanov, 2017). Automation allows organizations to test multiple systems, networks, or applications at the same time, increasing scalability and ensuring consistent coverage across their entire infrastructure. Automated tools adhere to a standardized methodology, which aids in the production of consistent results across different tests and testers, thereby improving the accuracy of risk assessments and vulnerability detection. Automated tools can test a broader range of attack vectors and vulnerabilities than manual testing alone, providing more comprehensive coverage. While there are some upfront costs associated with acquiring and deploying automated tools, they can be more cost-effective in the long run by reducing the need for additional human resources and allowing for more frequent and efficient testing. Typically, automated tools generate detailed, standardized reports that document the testing process, findings, and remediation recommendations, which can improve incident response and overall security strategy. Many automated tools incorporate the most recent threat intelligence, ensuring that they remain effective in the face of evolving threats and vulnerabilities. Automation has become a critical component of effective penetration testing strategies, providing numerous benefits that assist organizations in overcoming the challenges associated with manual testing. Organizations can streamline their testing processes, improve efficiency, and maximize resource allocation by leveraging automated tools such as vulnerability scanners, web application scanners, code analysis tools, and network configuration analysis tools and integrating them into the CI/CD pipeline(Nikolov & Slavyanov, 2017). Faster testing, scalability, consistent results, reduced human error, improved coverage, cost-effectiveness, better documentation, and continuous learning are some of the key benefits of incorporating automation into penetration testing. These advantages allow organizations to not only detect and remediate vulnerabilities more quickly but also to adapt to evolving threats and maintain a strong security posture. It is critical, however, to recognize that automation should be used in tandem with manual testing to ensure comprehensive coverage and robust security assessments. Human intuition and expertise are invaluable in identifying complex vulnerabilities, validating the findings of automated tools, and dealing with potential false positives or negatives. For optimizing penetration testing processes and achieving the most accurate results, a balanced approach that combines the strengths of both automation and manual testing is required(Nikolov & Slavyanov, 2017). To successfully incorporate automation into your organization's penetration testing strategy, you must first select the appropriate tools and technologies, train your security team on their use and limitations, establish a process for maintaining and updating the tools, and create a methodology that effectively combines the results of automated and manual testing efforts. Organizations can improve their security assessments by addressing these technical aspects and striking a balance between automation and manual testing. This ensures that their infrastructure remains secure against ever-evolving threats.

# 7 End-project report

## 7.1 summary and achievements

This tool automates the penetration testing process, which previously required a significant amount of manual labour and expertise. This reduces the time and resources required to identify system vulnerabilities. Once identified, it can intelligently exploit vulnerabilities in a system using machine learning algorithms. By learning from previous attempts, the tool improves its success rate over time, allowing it to adapt to new vulnerabilities and security measures. The project integrates successfully with the Metasploit Framework, one of the most popular tools for penetration testing and vulnerability discovery. This enables the tool to make use of Metasploit's extensive database of exploit modules. This can generate reports about the vulnerabilities it discovers and the exploits it employs automatically. This is critical for documenting the penetration testing process and providing testing evidence for compliance purposes. One of this tool's most significant accomplishments is its contribution to the advancement and awareness of the use of artificial intelligence in cybersecurity. It demonstrates machine learning's potential for improving and automating complex cybersecurity tasks.

## 7.2 Realization of business objectives

Using a tool like this to automate penetration testing can significantly reduce the amount of manual labour required, resulting in cost savings. This could assist businesses in meeting budget and operational efficiency goals. The tool can assist organizations in better understanding their security risks by identifying and exploiting vulnerabilities. This can help with risk management and compliance objectives. This project can assist organizations in continuously improving their security posture by providing an automated and intelligent way to conduct penetration tests. This can help them achieve their cybersecurity and data protection goals. The tool's ability to learn from previous attempts and improve over time can make the penetration testing process more efficient. This could help achieve operational efficiency goals. The reporting capabilities of the tool can aid in demonstrating compliance with various cybersecurity regulations, which may be a significant business goal for many organizations.

## 7.3 Changes made during the project

### *7.3.1 Switching from MulVal to Keras and TensorFlow*

TensorFlow and Keras both include a plethora of pre-built functions, models, and layers that can be used to design, train, and validate nearly any machine learning model imaginable. Because of their adaptability, they are suitable for a wide range of tasks, including cybersecurity. TensorFlow is designed for scalability. It is ideal for handling large datasets and complex computations because it can run on multiple CPUs, GPUs, and even distributed setups. Keras, which is built on top of TensorFlow, provides an intuitive interface for developing and training machine learning models. This allows for faster and easier prototyping of different models, which can be a significant advantage in a field like machine learning where iterative development and testing are common. And TensorFlow and Keras have a large and active developer and research community. This means that new features and improvements are added regularly, and help or advice is frequently available via community forums or resources.

# 8 Project post-mortem

## 8.1 Description of the project

The goal of this project was to create an automated penetration testing tool using machine learning technology. The tool's goal was to identify and exploit vulnerabilities in systems, networks, or web applications, with machine learning to improve its effectiveness over time.

javelin is a cutting-edge, machine learning-based penetration testing tool that intelligently identifies and exploits vulnerabilities in target systems. By automating the vulnerability detection and exploitation process, the project's primary goal is to provide cybersecurity professionals with a more effective and efficient method for assessing a system's security posture. In an increasingly complex security landscape, this advanced tool enables penetration testers to make data-driven decisions and streamline their workflows. Several key components at the heart of this tool's architecture work together to provide an intelligent and comprehensive penetration testing solution, The tool's deep learning model, which predicts the success rate of a given exploit against a specific target system, is central to its functionality. The model factors in various features, such as target system characteristics, service versions, and exploit metadata, by training on historical data that includes both successful and unsuccessful exploit attempts. As a result, the tool can make informed decisions when selecting exploits, increasing the likelihood of detecting vulnerabilities while decreasing the number of false positives. The tool is based on a large exploit database that contains a wide range of exploits targeting various vulnerabilities. The tool stays current with the latest threats and can adapt to the ever-changing security landscape by regularly updating its knowledge base with newly discovered exploits.

## 8.2 Project Achievements

- The tool was successful in automating many penetrations testing tasks, increasing efficiency.

  Javelin uses machine learning to automate penetration testing, which is an important aspect of cybersecurity. Traditional penetration testing often necessitates a significant amount of time and manual labour, but Javelin can complete these tasks much more quickly and consistently. This automation saves human testers time on routine tasks, allowing them to focus on more complex aspects of security testing. As a result, overall testing efficiency is significantly increased.
- The machine learning component allowed the tool to improve its ability to identify and exploit vulnerabilities by learning from each test.

  Javelin's machine learning component is one of its most important features. To learn the characteristics of vulnerabilities and improve its ability to exploit them, the tool employs a multi-layer perceptron, a type of artificial neural network. With each test, the tool learns more about the system being tested and how it responds to various types of attacks. As a result, Javelin's performance improves over time, identifying and exploiting vulnerabilities more effectively with each subsequent test.

- The tool could generate detailed reports on the vulnerabilities it discovered, which provided valuable information to the security team.

  Javelin not only finds vulnerabilities but also provides detailed reports on each one. These reports contain information about the vulnerability's nature, how it can be exploited, and potential mitigation strategies. This is critical for the security team because it allows them to comprehend the situation and prioritize their response. Javelin's detailed reporting feature is thus a valuable source of actionable intelligence for the security team.
- The tool's user-friendly interface and accurate results received positive feedback from users.

  Despite its sophisticated technology, Javelin has a user-friendly interface that makes it accessible to users of varying technical expertise. This ease of use, combined with the tool's ability to produce accurate results, has helped it gain popularity among users. The interface is simple to use, making it simple for users to set up and run tests, and the results are presented in an easy-to-understand format. This combination of usability and accuracy has resulted in positive user feedback.

## 8.3 Areas for Improvement

- Despite its machine-learning capabilities, the tool occasionally overlooked vulnerabilities that a human tester would have discovered. This emphasized the importance of using a hybrid approach that combines automated and manual testing.

  While Javelin is very good at automating penetration testing tasks and learning from them, it is not without flaws. There are times when it may miss vulnerabilities that a skilled human tester would notice. This could be because certain vulnerabilities are complex, or because they involve elements that the tool's machine-learning model hasn't been trained to recognize. It emphasizes the importance of combining the efficiency of automated testing with the nuanced understanding and adaptability of human testers.
- The tool produced some false positives, requiring the security team to do additional work to verify the results.

  While Javelin is generally accurate, it can produce false positives from time to time, flagging harmless elements as vulnerabilities. This can add to the security team's workload because they must verify the results, which can be time-consuming and divert resources away from addressing actual threats. Improving the tool's machine learning model, specifically its ability to distinguish between true and false positives, would help to alleviate this problem.
- The tool required a significant amount of computational power to function properly, which may be prohibitively expensive for some users.

  Javelin requires a significant amount of computational power due to its reliance on machine learning and the complex tasks it performs. This means that it may require a high-end system to run efficiently, which could be a barrier for smaller organizations or individuals with limited budgets. Investigating ways to improve the tool's algorithms or make it more efficient could help reduce this requirement, making it more accessible to a broader range of users.

## 8.4 Lessons Learned

- While automation can boost efficiency, it is not a replacement for human expertise. Future projects should think about combining automated and manual testing.

  Javelin has proven the effectiveness and efficiency of automation in penetration testing. However, as evidenced by its occasional misses, it also demonstrated that human expertise remains critical in this process. Because of their understanding of the system and ability to think creatively, automated tools like Javelin may miss certain vulnerabilities that a human tester would catch. This highlights the importance of a hybrid approach that takes advantage of automation's speed and consistency while also benefiting from the adaptability and insight of human testers.

- To reduce false positives, machine learning models must be trained with high-quality data.

  With Javelin, the occurrence of false positives is a challenge. False positives can create extra work and cause real threats to go unnoticed. The quality of data used to train machine learning models is critical in reducing these. More comprehensive and diverse data sets enable the model to distinguish between genuine threats and harmless anomalies, lowering the likelihood of false positives.

- System requirements should be considered early in the development process to ensure that the tool is accessible to all potential users.

  Javelin, while powerful, necessitates significant computational resources to function properly. This may make it inaccessible to smaller organizations or individuals who lack such resources. As a result, system requirements should be considered early in the development process. This could entail optimizing the tool's performance or developing lighter versions that can run on less powerful systems, allowing such tools to be accessible to a broader range of users.

## 8.5 The Recommendations for Future Projects

- Include a feedback loop that allows human testers to correct the tool's errors, allowing the machine-learning model to be improved further.

  In the case of Javelin, a feedback loop in which human testers can correct the tool's errors would be advantageous. Human testers could review the tool's findings, identify any errors or oversights, and then feed this information back into the tool. This feedback could then be used by the machine learning model to improve its future performance by learning from its mistakes and refining its vulnerability detection and exploitation techniques. This would result in a continuous cycle of improvement and adaptation, eventually leading to a more effective and accurate tool.

- To reduce false positives, consider investing in more advanced machine learning models or techniques.

  False positives, as seen with Javelin, can add to the security team's workload while potentially distracting from real threats. Future projects could consider investing in more advanced machine learning models or techniques to address this issue. This could include employing more complex models, such as deep learning, or implementing techniques such as anomaly detection to distinguish between actual vulnerabilities and harmless anomalies. Furthermore, using high-quality, diverse training data can help improve model accuracy and reduce false positives.

- To reduce the computational power required, consider optimizing the tool's code or using more efficient algorithms.

  Some users may be put off by Javelin's high computational requirements. Future projects could look into ways to reduce these requirements to make the tool more accessible. This could entail optimizing the tool's code to make it run faster, or employing more efficient algorithms that can perform the same tasks with less computational power. Reducing these requirements would not only make the tool more accessible to a broader range of users, but it would also potentially improve the tool's performance, making it faster and more responsive.

# 9 Conclusions

The project to develop an automated penetration testing tool using machine learning technology was successful in achieving its objectives. The tool was able to automate many tasks related to penetration testing, improving efficiency and allowing the security team to focus on more complex tasks. The machine learning component was a significant advantage, enabling the tool to learn from its actions and improve over time. This aspect played a crucial role in enhancing the accuracy of identifying and exploiting vulnerabilities. The tool was also able to generate detailed reports on the vulnerabilities it discovered, providing valuable information that aided in refining the security measures. The user-friendly interface was well-received by users, making the tool accessible and easy to use even for those who aren't tech-savvy.

However, the project also highlighted some areas for improvement. The tool occasionally missed vulnerabilities that a human tester might have spotted, and it produced some false positives. This reaffirms the importance of combining automated tools with manual testing for a comprehensive penetration testing strategy. Additionally, the high computational power required by the tool could be a barrier for some users. This suggests that future iterations of the tool could benefit from code optimization or the implementation of more efficient algorithms.

In conclusion, the project successfully developed an automated penetration testing tool using machine learning, demonstrating the potential of AI in cybersecurity. Nevertheless, the project also underscored the importance of balancing automation with human expertise and considering user constraints such as computational power requirements. Future work should aim to address these issues while continuing to leverage and expand upon the significant benefits of machine learning for cybersecurity.

## 10 Reference List

Altulaihan, E. A., Alismail, A., & Frikha, M. (2023). A Survey on Web Application Penetration Testing. In *Electronics (Switzerland)* (Vol. 12, Issue 5). MDPI. https://doi.org/10.3390/electronics12051229

Dalalana Bertoglio, D., & Zorzo, A. F. (2017). Overview and open issues on a penetration test. *Journal of the Brazilian Computer Society*, *23*(1). https://doi.org/10.1186/s13173-017-0051-1

Ghanem, M. C. (n.d.). *Towards an efficient automation of network penetration testing using model-based reinforcement learning. Anti-Forensics: A Script-Based Tool for Extracting Evidence Hidden by Cryptographic and Steganographic Techniques View project*. http://openaccess.city.ac.uk/

Nikolov, L., & Slavyanov, K. (2017). *ON THE CONTEMPORARY CYBERSECURITY THREATS Cybersecurity Test Lab View project Artificial intelligence procedures for ISAR object recognition. View project*. https://www.researchgate.net/publication/328355728

Reddy Mamilla, S. (n.d.). *A Study of Penetration Testing Processes and Tools*. https://scholarworks.lib.csusb.edu/etd/1220

Samant, N. (2011). *AUTOMATED PENETRATION TESTING* [San Jose State University]. https://doi.org/10.31979/etd.fxpj-pt6k

Tudosi, A. D., Graur, A., Balan, D. G., & Potorac, A. D. (2023). Research on Security Weakness Using Penetration Testing in a Distributed Firewall. *Sensors*, *23*(5). https://doi.org/10.3390/s23052683

## 11 Bibliography

"Metasploit: The Penetration Tester's Guide" by David Kennedy, Jim O'Gorman, Devon Kearns, and Mati Aharoni. No Starch Press, 2011.

"The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy" by Patrick Engebretson. Syngress, 2013.

"Penetration Testing: A Hands-On Introduction to Hacking" by Georgia Weidman. No Starch Press, 2014.

"Automated Security Analysis of Android and iOS Applications with Mobile Security Framework" by Ajin Abraham and Henry Dalziel. Packet Publishing, 2015.

"Automated Penetration Testing: a Framework Approach" by J. J. Shoup. NPS Thesis, 2007.

"Evaluating the Effectiveness of Automated Penetration Testing Tools" by S. Hazarika, D. P. Bhattacharyya, and J. K. Kalita. In the International Journal of Network Security, 2016.

"The Future of Automated Penetration Testing" by R. J. Seacord and C. D. Hines. In the Software Engineering Institute, Carnegie Mellon University, 2018.

"OWASP ZAP: The Zed Attack Proxy Project" by the Open Web Application Security Project.

"Metasploit Unleashed" by the Metasploit Team.

"The Nessus Vulnerability Scanner" by Tenable.

"OpenVAS - Open Vulnerability Assessment Scanner" by Greenbone Networks.

## 12 Appendices

### 12. 1 Project initiation document

UNIVERSITY OF PLYMOUTH

# PUSL3119 Computing Individual Project

## Project Initiation Document (PID)

Automated Penetration Testing

Supervisor: Mr. Chamindra Attanayake

Name: Rathnayaka Rathnayaka
Plymouth Index Number: 10747919
Degree Program: BSc (Hons) Computer Security

# ABSTRACT

## *i -Introduction*

Penetration testing can test the current system using various attacks. After doing these kinds of tests, developers can add security patches to the system. If the receiver of the attack is related to the healthcare sector, small system offline time can cause huge damage - this is why cyber security is so important.

The goal of the project is to automate the current process of penetration testing. In this process program must scan the ports automatically and give the data back to the program itself. After getting the output of the previous process program will automatically try to the penetration of the given system.

## *ii -Project objectives*

Intelligence gathering is one of the most crucial steps in a successful penetration test. This typically contains information about the target's business, people, property, and other relationships. Numerous sources, including but not limited to social sites, can be used to find out this information.

In theory, penetration tests are meant to be performed by humans but can and ought to be automated in some cases. They aim to find known software problems, such as a server missing security updates or using a default password.

Even for smaller businesses, penetration testing is becoming a crucial activity. The cost of such tests is one of the most significant economic pressures. Pentest can be automated with the help of tools like Metasploit. However, post-exploitation operations are difficult to automate.

## *iii -METHOD OF APPROCH*

Python scripts can automate a wide range of repetitive operations that consume your time and drain your energy. The most monotonous, time-consuming jobs may be sped up and simplified with Python automation.

Pen testers are unable to complete their work without automated tools. The most popular operating system for penetration testing is Kali, a free program created by Offensive Security. Nmap displays which ports are open, and what programs are running on them, helps you understand network pathways, and does an asset inventory on a target network.

Python is arguably the most popular programming language for penetration tester pen testing. Laying the foundation for the attack is the first stage; this entails specifying the range of your attack as well as the overarching concept. The second stage consists primarily of research to identify any potential system flaws.

Pen testers map out the attack surface and potential vulnerabilities of the target using this information. The type of reconnaissance varies depending on the goals and parameters of the pen test. Pen testers decide the appropriate tools and tactics to acquire access to the system, whether through a weakness such as SQL injection or malware.

## Table of Contents

# 1 – INTRODUCTION

## 1.1 Background of the project

In the current technological era, cyber security is mostly needed due to various factors, such as various kinds of offline and online attacks and mostly human errors. Because of these kinds of malicious attacks, so many technologies are going offline for a long time. If the receiver of the attack is related to the healthcare sector, small system offline time can cause huge damage.

To prevent these kinds of disasters security experts can use passive and active security systems, yet if there is a new attack these systems will not pick up the vulnerability at the first attempt. To identify the vulnerabilities in the system, cyber security experts can stress test the system using various penetration testing methods.

Penetration testing can test the current system using various attacks. After doing these kinds of tests, developers can add security patches to the system according to the identified vulnerabilities.

## 1.2 Problem statement

Penetration testing can be time consuming and it is very hard to track every single outcome while concentrating on the testing attack. Also, penetration testing on multiple systems manually is very hard because the outcomes of the tests can be the same. Because of this reason, some vulnerabilities cannot be found in time. If the vulnerability is a critical one, attackers can easily gain the access to the system even from a small vulnerability. Because of this reason penetration testing of the system must be fast and accurate. The current ability of multitasking on penetration testing is very low and because of that reason productivity of the test will get very low. To increase the productivity of the testing above problems must be fixed.

## 1.3 Solution for the problem

To reduce human errors and high consumption of time, Automation is a good solution.

To automate the penetration testing process, each and every element must be automated. In this process program must scan the ports automatically and give the data back to the program itself. After getting the output of the previous process program will automatically try to the penetration of the given system automatically according to the input data of the current process. Likewise, this program will not do the penetration correctly every time. The goal of the project is to automate the current process of penetration testing.

## 1.4 Project outcome

The outcome of this project is that at the end of this project the program must be able to do penetration testing automatically. Also, the penetration testing process using this program must be very user friendly and any user with little experience should use this program to do penetration testing.

# 2 – PROJECT OBJECTIVES

## 2.1 Automate Reconnaissance

Information collecting is one of the most crucial phases of a penetration test. The more knowledge obtained, the higher the likelihood of success, which is a widely held maxim. As a result, a considerable amount of time is invested in learning as much as possible about the target. This typically contains information about the target's business, people, property, and other relationships. Numerous sources, including but not limited to social sites, professional profiles, business directories, and corporate blogs, can be used to find out this information.

## 2.2 Automate Scanning

In practice, penetration tests involve a variety of tasks, some of which must be performed manually but which may and ought to be automated. For instance, while guessing passwords, a human tester may consider the employees of a company and adapt part of their guesses based on birthdays or pet names discovered online. They may even modify the company name or office address in the hopes of coming up with an intriguing result. However, this can and ought to be automated when it comes to finding known software problems, such as a server missing security updates, using a default password, or being unintentionally exposed to the internet.

## 2.3 Automate Vulnerability Assessment

Servers, laptops, firewalls, printers, containers, and other network assets are all identified by vulnerability scanners, and their operating data is continuously gathered. These scanning systems also include auditing, logging, threat modelling, reporting, and remediation capabilities, enabling enterprises to evaluate their various levels of security vulnerabilities at any given time.

Government requirements and industry standards frequently need vulnerability scanners to maintain data protection and improved security, in addition to being recognized as a standard best practice in enterprise networks.

## 2.4 Automate Exploitation

Even for smaller businesses, penetration testing is becoming a crucial activity. The price of such tests is one of the most significant economic pressures. Pentest can be automated with the help of tools like Metasploit. However, post-exploitation operations are difficult to automate. Our contribution entails expanding Meterpreter-scripts to enable scripting of post-exploitation. Additionally, by employing a multi-step strategy called pivoting, we may automatically attack devices that are not immediately routable. After the first machine is compromised, the script keeps going and launches an assault on the second machine, etc.

# 3 LITERATURE REVIEW

## 3.1 Penetration Testing

In a security exercise called penetration testing, a computer system's weaknesses are sought after and exploited. This simulated attack is meant to find any vulnerabilities in a system's defenses that an attacker might exploit.

This is comparable to a bank hiring a thief to try to break into their building and open the vault. The bank will learn vital information about how to tighten its security systems if the "burglar" is successful in breaking in and taking over the bank or the vault.

It is ideal to have a pen test carried out by someone who has little to no prior knowledge of how the system is protected since they may be able to reveal blind spots missed by the system's engineers. Due to this, outside contractors are typically hired to conduct the tests. Since they are employed to hack into a system with consent and to increase security, these contractors are frequently referred to as "ethical hackers."

When it comes to the cyber security penetration testing plays a unique part. To identify the vulnerabilities of the current system and make patches for the current system plays a huge part in the cyber security sector. When penetration testing is done correctly developers can get the insight of the system security hierarchy, this will allow them to make secured system. When it comes to the penetration testing, it will simulate the patterns of the intruder or malwares like botnets, DDoS, etc. Because of the knowledge availability of the hacking and malware creation, security experts and developers are trying to find any problems of their systems before launch. When it comes to this matter penetration testing plays a huge role.

## 3.2 Automation

With an increasingly complicated cyber threat landscape and an unprecedented shortage of skilled security personnel, many firms are seeking for solutions to improve and simplify security operations. Even though staffing and cybersecurity budgets are expanding, they aren't keeping pace with the increase in threats. Therefore, businesses are having trouble locating solutions that offer sufficient protection at the scale they require and at a fair price. to this problem, the solution lies in automation.

The notion of security automation has evolved over the past few years. A typical description in the past would have been the automation of cybersecurity safeguards. But today's definition of security automation does not accurately reflect it. These two words have now developed into a phrase with definite associations to particular traits and skills.

There is a better approach to specify security automation now.

the automation of threat detection and penetration testing systems while also adding to the system overall threat database to prepare for and protect against future malwares and attacks. It is made to automatically carry out best practices specified by your SecOps team at high machine speeds to expedite resolution, easy network communications, and reduce huge risks.

## 3.3 Automated Penetration Testing

Automated penetration testing is the practice of analyzing a system's security architecture utilizing cutting-edge testing technologies. Automated penetration testing is most effective when used frequently to thwart online threats and attacks.

An improved version of manual penetration testing is automated penetration testing. It quickly identifies vulnerabilities by using machine learning and algorithms to detect holes in systems.

# 4 METHOD OF APPROCH

## 4.1 Automation using python

The most monotonous, time-consuming jobs may be sped up and simplified with Python automation. there must be a less complicated way to go about and complete a simple activity. In other aspects of your career and life, repetitive tasks can be a drag and hinder your creativity and productivity. If only you could automate some of the repetitious duties that the system does.

Python's simple syntax is another factor contributing to its popularity among software engineers. Compared to many other programming languages, Python code closely resembles the English language. In comparison to other languages, Python tends to be more compact and takes fewer lines of code to complete the same tasks. Python is very simple to learn for both novice and seasoned programmers.

Python's user-friendliness enables programmers of all experience levels to start writing practical Python scripts with a minimal investment of time and effort. Python scripts can automate a wide range of tedious, repetitive operations that consume your time and drain your energy.

## 4.2 Utilizing the basic penetration testing tools

Penetration testers, like attackers, are unable to complete their work without automated tools. In order to carry out their simulated assault and automatically scan a website for vulnerabilities, pen testers need tools. Here are a few powerful tools that are frequently employed in penetration examinations.

Information is more exposed than ever, and each new technology development creates a new security risk that calls for fresh security measures. By carefully exposing its weaknesses, penetration testing is used to assess the security of an IT infrastructure. It aids in determining how effective the defense systems, methods, and policies put in place are. To discover hazards and control them to meet better security standards, penetration testing is constantly carried out. In this essay, we cover the significance of penetration testing, the factors and elements taken into account when performing a penetration test, a survey of the tools and techniques used, the role of penetration testing when implementing IT governance in an organization, and, finally, the professional ethics that the penetration test team should possess.

The most popular operating system for penetration testing is Kali, a free program created by Offensive Security. Over 500 cybersecurity tools are included with Kali, which can be used for information gathering, vulnerability analysis, exploitation, wireless attacks, forensics, web application attacks, stress testing, sniffing, password attacks, and more. The free tool Nmap displays which ports are open, and what programs are running on them, helps you understand network pathways, and does an asset inventory on a target network. The fact that Nmap is a legal tool that is frequently employed on business networks for legal purposes is advantageous.

In order to recognize compromised accounts and track lateral movement across various IT systems and user accounts, traditional security technologies are insufficient.

The Metasploit Cybersecurity Management Platform offers Advanced Analytics of the system that can assist your business in quickly identifying insider risks and sophisticated attack methods. Metasploit will assist in triaging, investigating, and blocking threats with a little investment of security analyst resources, whether it be an external attacker, an insider threat, or a penetration tester.

## 4.3 Automate penetration testing using python

When it comes to pen testing, Python is arguably the most popular programming language. This is partially attributable to the abundance of available external Python libraries. The penetration tester's life is made much simpler by these libraries.

Python's simplicity is an added benefit. Python differs from most programming languages in that it doesn't take much research to learn the fundamentals. You may rapidly catch up with some of the language's more sophisticated concepts and implementations thanks to how straightforward and well-designed it is.

Additionally, Python may be executed quite quickly if used correctly, which can make a significant difference over time, especially for a project of this size. Laying the foundation for the attack is the first stage. This entails specifying the range of your attack as well as the overarching concept and desired outcome.

Discovery, usually referred to as fingerprinting, speaks for itself quite well. The goal is to learn as much as you can about the system you wish to break into. This section consists primarily of research to identify any potential system flaws.

Naturally, the reporting process requires you to produce a report of all known and possible vulnerabilities. Every report must be quite particular and in-depth; otherwise, it won't be of much use in enhancing system security.

## 4.4 Testing

To guide the attack approach, assemble as much information as you can on the target from both public and private sources. Internet searches, the recovery of domain registration data, social engineering, network scanning, and occasionally even trash diving are sources. The attack surface and potential vulnerabilities of the target are mapped out by pen testers using this information. The type of reconnaissance varies depending on the goals and parameters of the pen test; it could be as straightforward as making a phone call to go through a system's features.

Attacker goals may include data theft, alteration, or deletion; the transfer of funds; or even just reputational harm to a business. To complete each test case, pen testers decide the appropriate tools and tactics to acquire access to the system, whether through a weakness such as SQL injection or using malware, social engineering, or anything else. When testing the program, it will scan the given system or website and give the details of the system and if there is a malfunction, it will be patched in this testing phase according to the above methods.

# 5 INITIAL PROJECT PLAN

## 5.1 Planning and analysis of the project

In this phase, main goal is that make the design of the project solid and create the basic structure of the program according to the main project idea. After the planning stage, project will ascend to the analysis phase that can determine what kinds of feathers and performance will end up in the final project structure.

## 5.2 Design the base program

In this phase, project will have a solid base program according to the basic penetration testing tools and penetration operations. In this phase, main goal is that project

has the basic functionalities of the manual penetration testing tool like Metasploit Meterpreter.

## 5.3 Testing manual penetration testing using base program

This phase is the first testing phase and, in this phase, program will go through some manual penetrating testing. Base program will have scanning and attacking options according to the planning phase and analysis of the current structure. To make testing easy, this phase will have some basic CLI (command line interface). In this phase project will not have bug fixes or additional implementations.

## 5.4 Implementation of the automation

This is the final development phase of this project. In this phase project will gain the ability of the automating. To implement automation project will have python programming. To ensure automation various python libraries will add to the project and project will backup the current progress logs of the current penetration test.

## 5.5 Testing and bug fixing

After developing the final program, project will have many simple and advanced testing phases using some capture the flag targets via Hackthebox website. This test will give the basic idea of the capabilities of the program about automation and penetration capabilities. if there is any bugs or malfunctions, these bugs will get a patch alongside with previous bugs.

# 6 RISK ANALYSIS

## 6.1 System Outages

To circumvent security precautions and take advantage of flaws, employers engage penetration testers. We are the ones that break things. It shouldn't come as a surprise that your main worry is the possibility that we might damage a crucial item. Anyone with penetration testing experience will be able to tell you about an important thing that went wrong.

System failures during penetration tests are typically caused by two things:

irresponsibility and unique or unexpected events.

## 6.2 Unusual Circumstances

Even the most vigilant penetration testers cannot always prevent outages or breaks. Software bugs in an application could lead to a Denial of Service situation. It is possible for a network device to be incorrectly configured, which would result in poor network traffic handling. In our testing, we have all encountered problems of this nature.

Unfortunately, there is no foolproof solution to solve this kind of issue. Best practices like patching, change management, and in-depth code reviews can help to reduce it. By continuously monitoring the systems being tested and being ready to stop automated tools at the first indication of danger, damage from this type of issue can be reduced. Making sure the testers are knowledgeable about how the systems being tested operate can also help.

## 6.3 Masking of Attacks

The organization being tested becoming complacent is another frequent risk of a penetration test. This Security Operations program may be disregarding signs of an actual assault if it ignores alarms because a penetration test is still running. This does not imply that the penetration test must be kept under wraps. On the contrary, it is best for the SOC and the penetration testing team to work together on the test and have open lines of communication. The outcomes can aid the SOC in enhancing alerting.

## 6.4 Inadvertent exposure

The unintentional disclosure of sensitive data or system access is the next important risk to take into account. Your penetration testers are executing exploits and looking for weaknesses. For instance, they might discover a weakness that enables them to access a backdoor. If they do this but leave the backdoor unprotected, a true attacker might find it and utilize it to their advantage. If the tester is accessing data insecurely, there is even another sort of exposure. They could transfer malwares over an unencrypted channel, for instance.

# 7 FUTURE IMPLEMENTATIONS

## 7.1 Machine learning and AI

To make sure that a firm has a strong cybersecurity network, penetration testing is really essential. Cybersecurity professionals test a software application's resistance using both manual and automated pen-testing methods.

The majority of manual testing is conducted by humans. Even in automated penetration testing, the human element is still a significant aspect and essential to the process' success.

Automation typically just removes redundancy; it does not address the cognitive component of testing. The effectiveness of the exam, therefore, rests more on the knowledge and conviction of the testing professionals than it does on the efficiency of the equipment.

However, by incorporating ML and AI into the automated penetration testing process, cognitive automation enters the picture and significantly reduces the need for human involvement. Additionally, by including accuracy and contextual knowledge, machine learning for penetration testing might change the endpoint.

***end of the PID***

## 12.2 interim report

# PUSL3119 Computing Individual Project

## Project Interim Report

# Automated Penetration Testing

Supervisor: Mr. Chamindra Attanayake

Name: Rathnayaka Rathnayaka

Plymouth Index Number: 10747919

Degree Program: BSc (Hons) Computer Security

# Table of Contents

## Table of Figures

# Chapter 01 Introduction

## Introduction

The use of software tools and scripts to simulate an attack on a computer system or network in order to identify vulnerabilities and weaknesses that could be exploited by real attackers is referred to as automated penetration testing.

Automated penetration testing tools typically scan the target system or network for known vulnerabilities and attempt to exploit them through a variety of techniques such as brute-force attacks, injection attacks, and buffer overflows. The testing results are typically presented in a report, which includes a list of the vulnerabilities discovered as well as recommendations for remediation.

While automated penetration testing can be useful for identifying vulnerabilities, it should be noted that it is not a substitute for manual penetration testing. Certain types of vulnerabilities may be missed by automated tools, resulting in false positives or false negatives. As a result, it is recommended that automated penetration testing be used as part of a larger security testing strategy that also includes manual testing by experienced security professionals.

## problem definition

Manual penetration testing, in which human testers use their knowledge and expertise to identify vulnerabilities, is an important component of ensuring the security of computer systems, networks, and applications. It is not without flaws, however. The following are some of the potential disadvantages of manual penetration testing

- Manual penetration testing can be time-consuming, particularly when testing large and complex systems. This can cause delays in identifying and addressing vulnerabilities, leaving systems vulnerable to potential threats.
- Manual penetration testing can only cover a small portion of the system being tested. This is because testers can only test what they know, and may not be able to identify all potential attack vectors.
- Human testers may bring biases and assumptions to the testing process, which can influence the results. This can result in false positives or false negatives, wasting time and resources.
- Manual penetration testing can be costly because it necessitates the use of skilled and experienced testers. This can be a challenge for smaller organizations or those with limited resources.
- Manual penetration testing can be inconsistent in terms of testing quality and rigor. This is due to the fact that different testers may approach the testing process in different ways, resulting in inconsistent results.

By providing a more consistent, efficient, and comprehensive testing process, automated penetration testing can help to address some of these issues. It is important to note, however, that automated testing should not completely replace manual testing, as human testers can provide context and insight that automated tools may miss.

## project objectives

Some of the flaws associated with manual penetration testing can be addressed with automated penetration testing. Here are some examples of how automated testing can be used to address the shortcomings of manual penetration testing.

### *Time efficiency*

- When compared to manual testing, automated penetration testing tools can test large and complex systems quickly and efficiently, saving time and resources.

### *Comprehensive testing*

- Automated penetration testing tools can detect a wide range of vulnerabilities, including those that human testers may overlook. This can result in a more thorough testing procedure.

### *Consistency*

- Automated penetration testing tools can provide a consistent testing process, which can help to reduce the possibility of bias or inconsistency that can occur during manual testing.

### *Cost-effectiveness*

- Because automated penetration testing tools can be run continuously and do not require as much human intervention, they can be more cost-effective than manual testing.

### *Reduced false positives*

- By using a variety of techniques to confirm the presence of vulnerabilities, automated penetration testing tools can help to reduce the number of false positives.

automated penetration testing can be a valuable tool for organizations seeking to strengthen their security posture. It is important to note, however, that automated testing should not completely replace manual testing, as human testers can provide context and insight that automated tools may miss. The most comprehensive testing process can be achieved by combining automated and manual testing.

# Chapter 02 System analysis

## Facts gathering techniques

### *Vulnerability databases*

Vulnerability databases are repositories that store data on known flaws in software, hardware, and network systems. These databases are managed by a variety of organizations, including security firms, government agencies, and industry associations. Some examples of vulnerability databases are as follows:

#### *National Vulnerability Database (NVD)*

- The National Institute of Standards and Technology (NIST) maintains the NVD, which contains information about known vulnerabilities in software, hardware, and other systems. The database contains descriptions of vulnerabilities, severity ratings, and links to patches and other fixes.

#### *Common Vulnerabilities and Exposures (CVE)*

- It is a dictionary of common names for publicly known cybersecurity vulnerabilities and exposures. Each CVE entry includes a unique identifier, a description of the vulnerability, and links to more information.

#### *The Exploit Database*

- The Exploit Database is a repository of exploits and vulnerabilities maintained by Offensive Security. The database contains exploits for a wide range of software and hardware systems, as well as vulnerability descriptions and links to more information.

These databases can be useful resources for security professionals, researchers, and organizations looking to identify and mitigate system vulnerabilities. It is critical to keep these databases up to date in order to have the most up-to-date information on vulnerabilities.

## Existing system

The following steps can be taken to perform a system analysis of the current manual penetration testing process

### *Identify the manual penetration testing process's objectives and goals*

- The first step is to identify the manual penetration testing process's objectives and goals. This could entail identifying and assessing system vulnerabilities, testing the effectiveness of security controls, and identifying potential areas for improvement.

### *Examine the current testing process*

- The next step is to examine the current manual penetration testing process for flaws and areas for improvement. This could include looking over testers' testing methodologies, tools, and techniques, as well as the documentation and reporting process.

### *Identify potential areas for improvement*

- Potential areas for improvement can be identified based on the review results. Introduce new testing methodologies or tools, improve the documentation and reporting process, or address any training or skill gaps among testers.

### *Create an improvement plan*

- Once potential areas for improvement have been identified, a plan for implementing these changes can be created. Identifying the resources needed, establishing timelines and milestones, and defining the roles and responsibilities of team members involved in the testing process are all examples of this.

### *Monitor and evaluate progress*

- It is critical to monitor and evaluate the effectiveness of the new manual penetration testing process after it has been implemented. This may include reviewing test results, gathering tester feedback, and identifying areas for further improvement.

a system analysis of the current manual penetration testing process can assist in identifying potential areas for improvement and ensuring that the testing process is effective in identifying and mitigating system vulnerabilities.

## Use case diagram

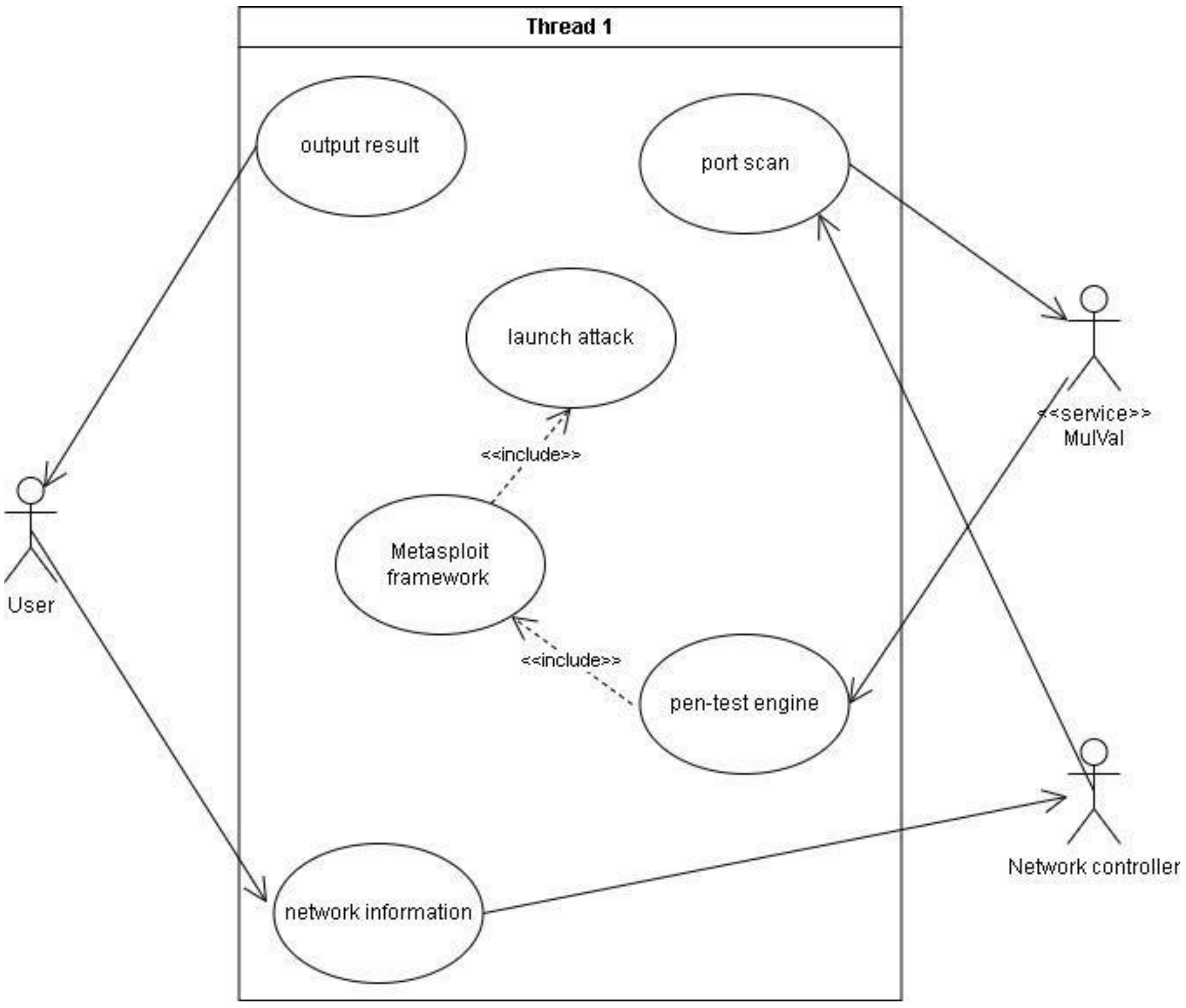


*Figure 1-Use Case*

## Drawbacks of the existing system

While automated penetration testing has many advantages, it is not without drawbacks. Here are some of the potential disadvantages of automated penetration testing

### *False positives and false negatives*

- Automated penetration testing tools can occasionally generate false positives or false negatives, resulting in wasted time and resources or missed vulnerabilities.

### *Lack of context*

- Human testers can provide context that automated penetration testing tools cannot. This can make interpreting testing results difficult, leading to incorrect conclusions or missing vulnerabilities.
- Automated penetration testing tools may have a limited scope and may not be capable of testing all areas of a system or application. This may result in some vulnerabilities going undetected.

### *Inability to detect new or emerging threats*

- Automated penetration testing tools may be incapable of detecting new or emerging threats that have yet to be discovered or documented.

### *Dependence on tools*

- The effectiveness of automated penetration testing tools is limited by the tools and techniques they employ. The testing process may be jeopardized if these tools become obsolete or ineffective.

### *Lack of creativity*

- Automated penetration testing tools may be lacking in the creativity and problemsolving abilities that human testers bring to the testing process. This can make identifying complex vulnerabilities or attack scenarios difficult.

it is critical to recognize that automated penetration testing should not completely replace manual testing, as human testers can provide context and insight that automated tools may overlook. The most comprehensive testing process can be achieved by combining automated and manual testing. Furthermore, it is critical to review and update automated testing tools and techniques on a regular basis to ensure that they continue to be effective in identifying and mitigating vulnerabilities.

# Chapter 03 -Requirements specification

## Functional requirements

The functional requirements of automated penetration testing tools will vary depending on the organization's specific needs and goals. However, the following are some common functional requirements that businesses may look for in automated penetration testing tools:

### *Vulnerability scanning*

- Automated penetration testing solutions should be able to scan for known vulnerabilities in software, applications, and systems.
- Organizations may need to customize automated penetration testing tools to meet their specific needs, such as targeting specific applications or systems or employing custom attack scenarios.

### *Reporting and documentation*

- Automated penetration testing tools should provide comprehensive testing results reporting and documentation, including vulnerability descriptions, recommended fixes, and severity rankings.

### *Integration with other tools*

- Automated penetration testing tools should be able to integrate with other security tools and systems, such as vulnerability management systems or SIEM platforms.

### *Continuous testing*

- Automated penetration testing tools should be able to run indefinitely in order to detect new vulnerabilities or changes in the system or application being tested.

### *Attack simulation*

- Penetration testing tools should be able to simulate a wide range of attack scenarios, such as network attacks, web application attacks, and social engineering attacks.

### *Performance testing*

- In addition to security testing, automated penetration testing tools may perform performance testing to evaluate system and application scalability and availability.

the functional requirements of automated penetration testing tools should align with the overall security strategy of the organization.

## Nonfunctional requirements

In addition to functional requirements, organizations should consider a number of non-functional requirements when selecting automated penetration testing tools. These non-functional requirements help ensure that the tools meet the organization's overall usability, performance, and security requirements. Here are some common non-functional requirements of automated penetration testing tools.

### *Usability*

- For testers with varying levels of experience and technical expertise, automated penetration testing tools should be user-friendly and simple to use. The tools should have a simple and intuitive user interface, as well as documentation and training resources to assist users.

### *Performance*

- Penetration testing tools that are automated should be able to scan and test large and complex systems and applications in a reasonable amount of time. The tools should also be scalable and capable of handling increasing volumes of data and traffic.

### *Security*

- Secure communication protocols, secure storage of sensitive information, and protection against attacks or vulnerabilities in the tools themselves should all be included in automated penetration testing tools.

### *Compatibility*

- Automated penetration testing tools should be compatible with the systems and applications being tested, as well as the organization's other security tools and platforms.

### *Customization*

- Automated penetration testing tools should be easily customizable to meet the organization's specific needs and requirements, such as the ability to create custom tests and attack scenarios.

### *Support*

- The vendor or provider of automated penetration testing tools should provide dependable and timely support to assist users with any issues or questions that may arise.

### *Cost*

- The automated penetration testing tools should be reasonably priced and fit within the organization's budget, while also providing sufficient functionality and capabilities to meet the needs of the organization.

organizations should carefully evaluate and select automated penetration testing tools that meet both functional and non-functional requirements in order to ensure that the tools are effective, secure, and simple to use.

## Hardware/ software requirements

Hardware requirements for automated penetration testing tools will vary depending on the size and complexity of the system or application being tested. The tool should, in general, be run on a system with sufficient processing power, memory, and storage. Some automated penetration testing tools may also necessitate the purchase of specialized hardware, such as network taps or additional servers.

### *Operating system*

- Some automated penetration testing tools, such as Windows, Linux, or macOS, may require a specific operating system. Furthermore, some tools may require specific operating system versions, such as Windows 10 or Ubuntu 20.04.

### *Dependencies*

- Automated penetration testing tools may rely on third-party software or libraries such as Python or Ruby. Before installing the tool, organizations should ensure that all required dependencies are installed and up to date.

### *Network access*

- Network access to the system or application being tested may be required by automated penetration testing tools. Access to specific ports, protocols, or network devices may be included. Before installing and using the tool, organizations should ensure that the necessary network access is available.

### *Integration with other tools*

- Automated penetration testing tools may require integration with other security tools and platforms in an organization's environment, such as vulnerability management systems or SIEM platforms. The tool should be interoperable with the other tools and platforms, and it should be able to exchange data with them as needed.

### *Licensing*

- Organizations should ensure that they have the appropriate licenses or subscriptions before installing and using automated penetration testing tools.

before installing and using automated penetration testing tools, organizations should carefully review the hardware and software requirements to ensure that they have the necessary resources and environment to support the tool's functionality.

# Chapter 04- feasibility study

## operational feasibility

The operational feasibility of automated penetration testing tools refers to how well the tool integrates into the organization's existing security operations and processes. Consider the following factors when assessing the operational feasibility of automated penetration testing tools:

Automated penetration testing tools should be able to integrate seamlessly with the organization's existing security tools and processes, such as vulnerability scanning, incident response, and risk management. The tool should also be able to generate reports that security teams and other stakeholders can easily consume.

### *User training and expertise*

- Automated penetration testing tools require technical expertise to set up and use effectively. Organizations should provide adequate training and support to security teams so that they can use the tool and interpret the results effectively.

### *Scalability*

- The automated penetration testing tool should be scalable to meet the organization's needs. It should be able to handle large and complex systems, applications, and networks, as well as generate timely reports and results.

### *Maintenance and support*

- Automated penetration testing tools require regular maintenance and updates to ensure that they remain effective against new threats and vulnerabilities. To ensure that the tool remains functional and secure, the tool's vendor or provider should provide dependable support and regular updates.

### *Regulatory compliance*

- The use of automated penetration testing tools may be subject to HIPAA, PCI-DSS, or GDPR compliance requirements. Organizations should ensure that the tool satisfies any relevant compliance requirements and can provide auditors with the necessary documentation and evidence.

before implementing automated penetration testing tools, organizations should carefully evaluate their operational feasibility to ensure that the tool can effectively integrate into the organization's security operations and processes and provide value in terms of risk reduction and mitigation.

## Technical feasibility

The technical feasibility of an automated penetration testing tool refers to whether the tool can perform the necessary functions while also meeting the technical requirements of the organization's security environment. Consider the following factors when assessing the technical feasibility of automated penetration testing tools

### *Compatibility with target systems*

- The tool should be able to communicate with the systems and applications being tested, including the operating system, programming language, and architecture. A variety of network protocols and applications should also be supported by the tool.

### *Functionality and capabilities*

- The tool should have the necessary functionality and capabilities to meet the security testing requirements of the organization. This includes the ability to identify and exploit vulnerabilities as well as generate detailed reports.

### *Accuracy and effectiveness*

- The tool should be capable of identifying vulnerabilities and simulating real-world attacks. The tool should be able to provide actionable results that can be used to improve the security posture of the organization.

### *Scalability and performance*

- The tool should be scalable and perform well in order to meet the needs of the organization. The tool should be able to handle large and complex systems, applications, and networks, as well as generate timely reports and results.

### *Ease of use and deployment*

- The tool should be simple to use and deploy, with simple documentation and user interfaces. The tool should also be able to be customized to meet the specific needs of the organization.

### *Security and dependability*

- The tool must be both secure and dependable, with built-in security controls and the ability to withstand attacks. The tool should also be capable of recovering from failures or errors, as well as providing adequate logging and auditing capabilities.

before implementing automated penetration testing tools, organizations should carefully evaluate their technical feasibility to ensure that the tool can effectively meet the organization's security testing requirements while also providing value in terms of risk reduction and mitigation.

# Chapter 05-System architecture

## class diagram

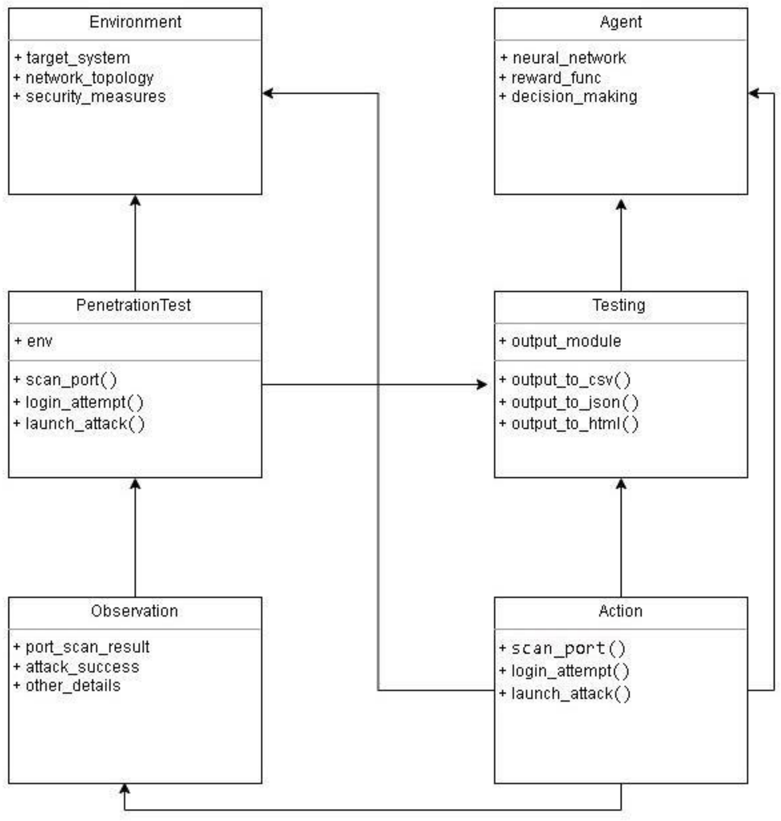


*Figure 2 - Class Diagram*

## ER diagram

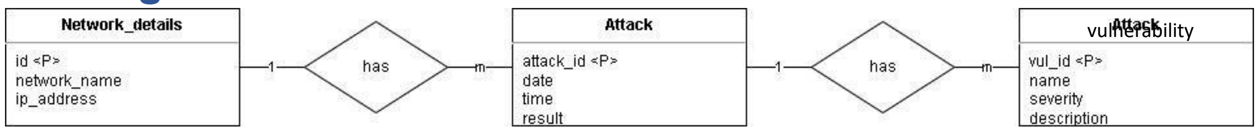


*Figure 3 - ER diagram*

# Chapter 06- Development tools and technologies

## development methodology

Creating a penetration testing tool typically entails adhering to a structured software development methodology that consists of several stages

### *Gathering Requirements*

During this phase, the developer determines the requirements for the penetration testing tool. This may entail analyzing the target systems, comprehending the types of attacks that must be carried out, and identifying the features required to carry out those attacks.

Gathering requirements is a critical step in developing a penetration testing tool. Following are some steps to take for effective requirements gathering:

#### *Define the Scope*

- The first step is to define the tool's scope. Determine the target environment, which includes things like the operating system, network infrastructure, and web applications.

#### *Identify the Risks*

- Identify potential threats and vulnerabilities in the target environment. Determine the types of attacks that the tool should be capable of performing in order to detect these risks.

#### *Define the Tool's Functionality*

- Based on the identified risks and attacks, define the tool's functionality. Find out what the tool can do, such as port scanning, vulnerability scanning, password cracking, and network traffic analysis.

#### *Consider Legal and Ethical Considerations*

- Think about the legal and ethical implications of using the tool. Check that the tool complies with all applicable laws and ethical standards.

#### *Document the Requirements*

- Clearly and concisely document the requirements. To help explain the requirements, use diagrams, flowcharts, and other visual aids. Ensure that the requirements can be tested and measured.

#### *Examine the Requirements*

- Discuss the requirements with stakeholders such as end users, developers, and project managers. Check that the specifications are complete, accurate, and feasible.

Following these steps will allow you to effectively collect requirements for a penetration testing tool and ensure that it meets the needs of the target environment and its users.

## *Design*

Once the requirements are gathered, the developer designs the architecture of the tool, including its components, data flow, and user interface. The design should address the tool's performance, reliability, and scalability.

A penetration testing tool's design phase is critical to ensuring that the tool is efficient, reliable, and meets the requirements. Here are some steps to take for an effective penetration testing tool design:

### *System Architecture*

- Define the tool's system architecture. Determine the tool's major components, such as the user interface, database, and analysis engine.

### *Data Flow*

- Determine the tool's data flow. Determine the input data, the processing logic, and the output data. Decide how the data will be stored, analyzed, and displayed to the user.

### *User Interface*

- Create the tool's user interface. Make sure the user interface is simple, intuitive, and user-friendly. To present the results, use visual aids such as graphs and charts.

### *Performance and Scalability*

- Consider the tool's performance and scalability. Check to see if the tool can handle large amounts of data and complex analysis. Improve tool performance by employing efficient algorithms and data structures.

### *Security*

- Make certain that the tool is secure. To prevent unauthorized access, use security features such as authentication and access control. To protect sensitive data, use encryption.

### *Testing*

- Define the tool's testing strategy. Determine the types of testing that will be performed, such as unit testing, integration testing, and acceptance testing. Create test cases to ensure that the tool meets the requirements and functions properly.

### *Documentation*

- Create tool documentation. User manuals, installation guides, and technical documentation should be provided. Check that the documentation is correct, complete, and up to date.

## *Implementation*

The developer then writes the code for the tool based on the design. The code should be welldocumented, modular, and efficient.

A penetration testing tool's implementation phase entails writing code and developing software components. Here are some steps to take for an effective penetration testing tool implementation:

### *Code Development*

- Use the programming language and development tools specified in the design phase to write the tool's code. To ensure code quality, readability, and maintainability, use coding standards and best practices.

### *Component Development*

- Create the tool's software components, such as the user interface, database, and analysis engine. Make sure the parts are modular, reusable, and scalable.
- Integrate the tool's components to create a cohesive and functional tool. To ensure that the components work together as expected, test the integration.

### *Security*

- Include security features in the code to ensure the tool's safety. To prevent vulnerabilities and attacks, use secure coding practices such as input validation and data encryption.

### *Testing*

- Thoroughly test the tool to ensure that it meets the requirements and performs as expected. To validate the tool's functionality and performance, use testing tools such as automated testing tools and manual testing techniques.

### *Debugging*

- Use debugging to identify and resolve any issues that arise during testing. To identify and isolate bugs, use debugging tools such as debuggers and logging mechanisms.

### *Documentation*

- Document the tool's code and software components. Documentation should be clear and concise, including code comments, user manuals, and technical documentation.
- You can effectively implement a penetration testing tool that meets the requirements, is secure, and works as expected if you follow these steps.

## *Testing*

In this phase, the developer tests the tool to ensure that it meets the requirements and works as expected. The testing should involve both functional and non-functional testing, including unit testing, integration testing, and performance testing.

### *Deployment*

- The tool is deployed in the target environment after it has been tested and validated. The deployment process may include configuring the tool and providing end-user documentation.

### *Plan the Deployment*

- Based on the target environment and user requirements, plan the tool's deployment.
  Determine the installation procedure, including the necessary hardware and software.

### *Install the Tool*

- Follow the installation instructions in the documentation to install the tool on the target environment. Test the installation to ensure that the tool is properly installed.

### *Configure the Tool*

- Set up the tool in accordance with the target environment and user requirements. Set the tool's parameters, such as the target IP address and port number.

### *Test the Tool*

- Run the tool in the target environment to ensure it works properly. Using test cases and scenarios, evaluate the tool's functionality and performance.

## *Maintenance*

After the tool is deployed, the developer should provide ongoing maintenance and support. This could include bug fixes, new features, and updating the tool to address new security threats.

### *Monitor the Tool*

- Keep an eye out for performance issues, errors, and security flaws. To identify and resolve issues, use monitoring tools and techniques.

### *Update the Tool*

- Add new features, bug fixes, and security patches to the tool. To avoid any negative impact on the tool's performance, ensure that the updates are tested before deployment.

### *Backup and restore*

- Make regular backups of the tool's data and configuration. Create a disaster recovery plan and test the backup and restore processes on a regular basis.

### *Provide Support*

- Provide end-user support, including technical assistance and troubleshooting. Ascertain that the support team is properly trained and equipped to handle any issues that may arise.

### *Examine the Tool*

- Examine the tool on a regular basis to ensure that it meets the requirements and is effective. Collect feedback from end users and stakeholders and make changes as needed.

Following these steps will allow you to deploy and maintain a penetration testing tool that meets the requirements, is reliable and secure, and adds value to the end-users.

## programming languages and tools

### *Basic programing concepts that used in this project*

*Nmap*

*implementatio n*

```
import nmap
def
port_scan(targ
et):
   scanner = nmap.PortScanner()
   scanner.scan(target, arguments='-p-')

   open_ports = []     for
host in
scanner.all_hosts():
        print("Host: %s (%s)" % (host,
scanner[host].hostname()))        print("State:
%s" % scanner[host].state())        for proto in
scanner[host].all_protocols():
print("Protocol: %s" % proto)            port_list =
scanner[host][proto].keys()            for port in
port_list:                if
scanner[host][proto][port]['state'] == 'open':
                    open_ports.append(port)
                    print("Port: %s\tState: %s" % (port,
scanner[host][proto][port]['state']))     if len(open_ports) == 0:
        print("No open ports
found.")     else:
        print("Open ports: %s" %
open_ports) target = '127.0.0.1'
port_scan(target)
```

*Metasploit implementation from metasploit*.

```
msfrpc import MsfRpcClient
msf_user = 'msf'
msf_pass = 'msfpass'
msf_host = 'localhost'
msf_port = 55552

client = MsfRpcClient(msf_pass, username=msf_user,
port=msf_port, server=msf_host) target_host = '192.168.0.1'
target_port = 445
exploit = client.modules.use('exploit',
'windows/smb/ms17_010_eternalblue') exploit['RHOSTS'] =
target_host exploit['RPORT'] = target_port if exploit.check():
    print(f'Target {target_host} is vulnerable to
{exploit.name} exploit!') else:
    print(f'Target {target_host} is not vulnerable to {exploit.name} exploit.')
```

### *basic Machine learning implementation*

```
import pandas as pd
import numpy as np
from sklearn.model_selection import
train_test_split from sklearn.preprocessing
import StandardScaler from
sklearn.ensemble import
RandomForestClassifier from
sklearn.metrics import confusion_matrix

data =
pd.read_csv('network_traffic.c
sv') X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0) sc = StandardScaler()
X_train =
sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = RandomForestClassifier(n_estimators=10,
criterion='entropy', random_state=0) classifier.fit(X_train, y_train) y_pred
= classifier.predict(X_test) cm = confusion_matrix(y_test, y_pred)
print(cm)
```

## *Machine learning*

Machine learning can improve the accuracy and effectiveness of vulnerability detection and exploit generation, making it a valuable tool for automated penetration testing. Here are some applications of machine learning in automated penetration testing:

Machine learning algorithms can be trained to detect anomalies in network traffic, application behavior, or system logs that could indicate a potential security issue or attack. These algorithms can assist in identifying zero-day vulnerabilities or unknown attack patterns that traditional rule-based systems may miss.

Machine learning algorithms can be used to analyze historical data and forecast future security risks or attacks. This can assist organizations in proactively mitigating security risks and preventing attacks from occurring.

Machine learning algorithms can analyze user behavior patterns in order to detect potential insider threats or malicious activities. These algorithms can assist organizations in identifying and preventing attacks from within, such as employees or contractors.

Machine learning algorithms can be taught to generate automated exploits for known vulnerabilities or common attack patterns. These algorithms can help to accelerate testing and free up resources for more complex or customized attacks.

Prioritization of vulnerabilities: Machine learning algorithms can prioritize vulnerabilities based on severity and likelihood of exploitation. This can assist organizations in concentrating their resources on the most critical vulnerabilities first, thereby reducing the overall attack surface.

While machine learning can be a useful tool for automated penetration testing, it should be noted that it is not a panacea. Machine learning algorithms require a large amount of data to effectively train, and they may produce false positives or false negatives. Furthermore, machine learning should not be used to replace human expertise and analysis, but rather as a supplement to improve the overall effectiveness of automated penetration testing.

## *python for automation*

Python is a popular programming language for automation because of its ease of use, versatility, and large library. Here are some examples of common Python automation use cases:

### *Web scraping*

- Using libraries such as BeautifulSoup, Scrapy, or Selenium, Python can be used to scrape data from websites.

### *File Management*

- Using built-in modules such as os and shutil, Python can automate tasks such as file renaming, copying, and moving.

### *Task Scheduling*

- Using Python's built-in module "sched," you can schedule tasks to run at regular intervals.
- Python can be used for test automation by utilizing frameworks such as PyTest or unittest.
- Python's natural language processing libraries, such as NLTK and spaCy, make it a popular choice for developing chatbots.

### *Image Processing*

- Image resizing, cropping, and filtering can be accomplished using Python libraries such as Pillow and OpenCV.
- Python is widely used in data analysis and can automate tasks such as data cleaning, manipulation, and visualization by utilizing libraries such as Pandas and Matplotlib.

### *Network Automation*

- Using libraries such as Netmiko or Paramiko, Python can automate network tasks such as configuring routers and switches.

These are just a few applications of Python for automation. Python, with its extensive libraries and resources, is an excellent choice for automating a variety of tasks.

## Third party components and libraries

### *MulVAL*

The Multi-stage Vulnerability Assessment and Exploitation Framework is a free and open-source automated penetration testing tool that combines vulnerability assessment, automated exploit generation, and network mapping. It identifies vulnerable targets using network reconnaissance techniques, assesses their security posture, and generates automated exploits to test them for vulnerabilities.

MulVAL employs network scanning techniques to identify the devices, services, and applications that are running on a target network. Following that, it creates a detailed map of the network architecture and identifies potential attack vectors.

Assessment of vulnerabilities: MulVAL scans the target network for known vulnerabilities and security flaws. It can detect vulnerabilities in operating systems, applications, and services and provides detailed information about each vulnerability's severity and exploitability.

MulVAL can generate automated exploits for vulnerabilities that have been identified. It exploits vulnerabilities and tests for system weaknesses by combining pre-defined exploit modules and customized scripts.

Reporting and visualization: MulVAL generates comprehensive reports on vulnerabilities, exploits, and network topology. It also generates graphical visualizations of the network architecture and identified vulnerabilities, which can help analysts and security teams understand the security posture of the target network.

MulVAL is a powerful tool for automated penetration testing, but it requires some technical knowledge to set up and use properly. It is best suited for experienced security professionals who are well-versed in network scanning, vulnerability assessment, and exploit generation techniques. Furthermore, as with any automated penetration testing tool, using MulVAL ethically and with proper authorization is critical to avoid legal and ethical issues.

# Chapter 07 – Discussion

## Discussion

Both automated and manual penetration testing have advantages and disadvantages, and they are frequently used in tandem to provide comprehensive security testing.

Automated penetration testing can be more efficient, faster, and less expensive than manual testing. Automated tools can quickly scan large amounts of code or network data and identify potential vulnerabilities, whereas manual testing can take days or weeks to complete. Automated tools can also perform tests that a human tester would find too complex or difficult to perform accurately, such as brute-force attacks or fuzz testing.

However, there are some limitations to automated testing tools. They may overlook vulnerabilities that can only be identified manually, such as logical flaws or misconfigurations. Automated tools may also produce false positives or false negatives, necessitating additional human analysis to determine whether or not a vulnerability exists. Furthermore, automated testing tools may be incapable of detecting advanced attacks or zero-day vulnerabilities that have yet to be discovered.

Manual penetration testing, on the other hand, can provide a more comprehensive and accurate assessment of a system's or network's security posture. Manual testing can uncover vulnerabilities that automated tools may overlook, such as social engineering or physical security flaws. Manual testing can also provide more detailed and contextual information about vulnerabilities, such as their impact and likelihood of being exploited.

Manual penetration testing, on the other hand, can be slower, more expensive, and resource-intensive than automated testing. Manual testing also necessitates more human expertise and experience in order to be effective. Furthermore, personal biases or errors in manual testing can result in inaccurate results.

Overall, both automated and manual penetration testing have advantages and disadvantages. Organizations should consider combining both approaches, tailored to their specific needs and resources, to ensure comprehensive security testing.

## Challenges faced

Debugging can be difficult, particularly when working with complex automation scripts. Errors are not always easy to spot and can be time-consuming to correct.

Automation scripts can expose you to security risks, especially if they access sensitive data or systems. It can be difficult to ensure that your Python code is secure and that sensitive information is properly protected.

If automated penetration testing is not combined with manual testing and analysis, it can create a false sense of security. Automation should be viewed as a supplement to manual testing rather than a replacement.

***end of the interim report***

# 13 User Guide

step 1 : open a terminal as sudo and enter following commands

- open a terminal
- enter ‘sudo su’

step 1 : install pip3(python3-pip

- sudo su
- apt-get install python3-pip

step 2 : install python libraries

- pip install beautifulsoup4
- pip install docopt
- pip install jinja2
- pip install Keras
- pip install matplotlib
- pip install msgpack-python
- pip install numpy
- pip install pandas
- pip install Scrapy
- pip install tensorflow
- pip install urllib3
- pip install protobuf

step 3 : insert the host details in to config.ini

- server_host : 192.168.12
- server_port : 55553
- msgrpc_user : admin
- msgrpc_pass : admin

step 4 : start Metasploit framework database

- msfdb init

step 5 : run msfconsole

- msfconsole

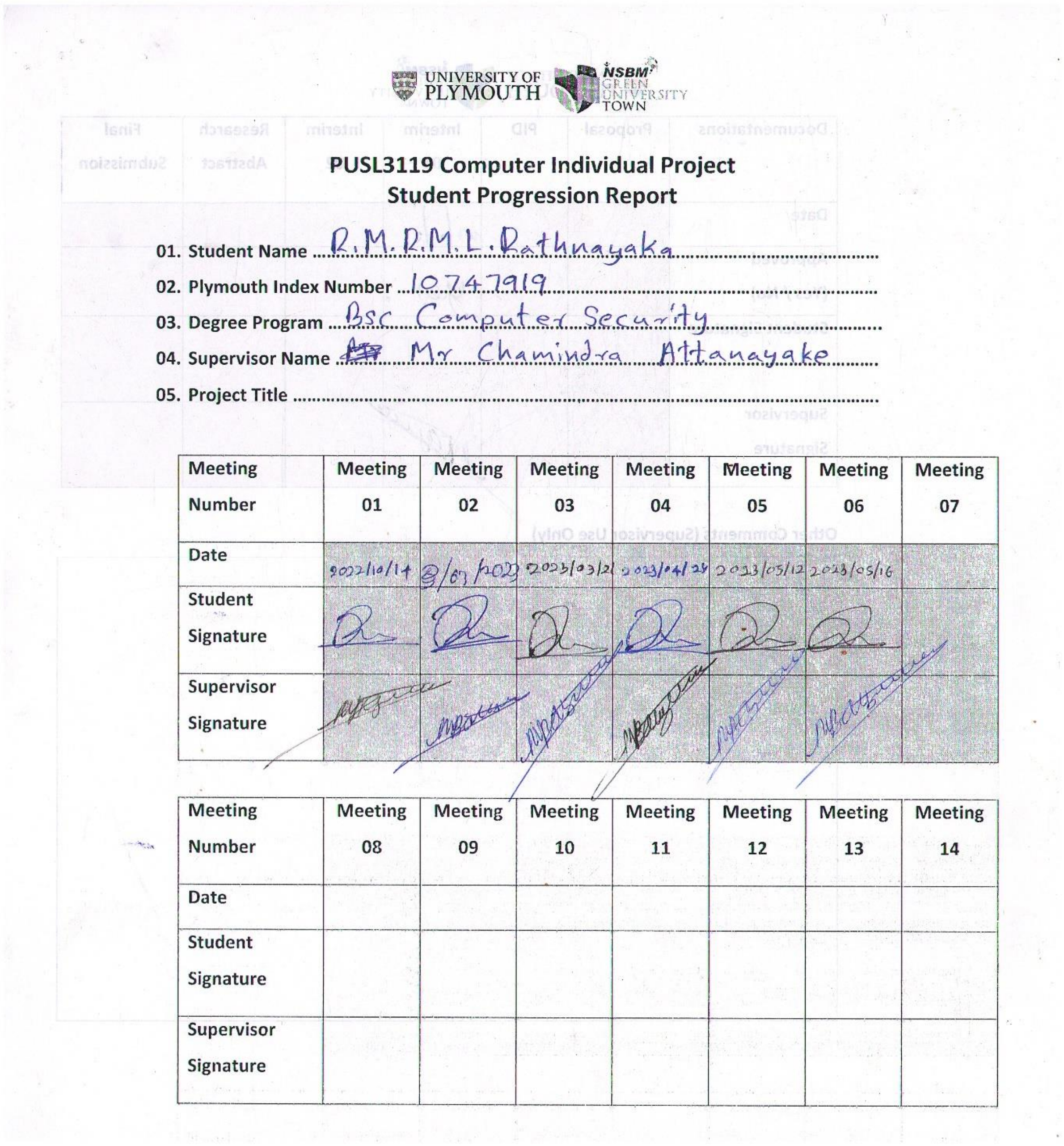
step 6 : configure RPC server according to config.ini

- load msgrpc ServerHost=192.168.220.144 ServerPort=55553 User=admin Pass=admin

step 7 : run the python file using following command

- python3 ./javelin.py <source_ip> -m <mode>
- ex: python3 ./javelin.py 192.168.1.3 -m test

## 14 Records of supervisory meetings



UNIVERSITY OF PLYMOUTH NSBM GREEN UNIVERSITY TOWN

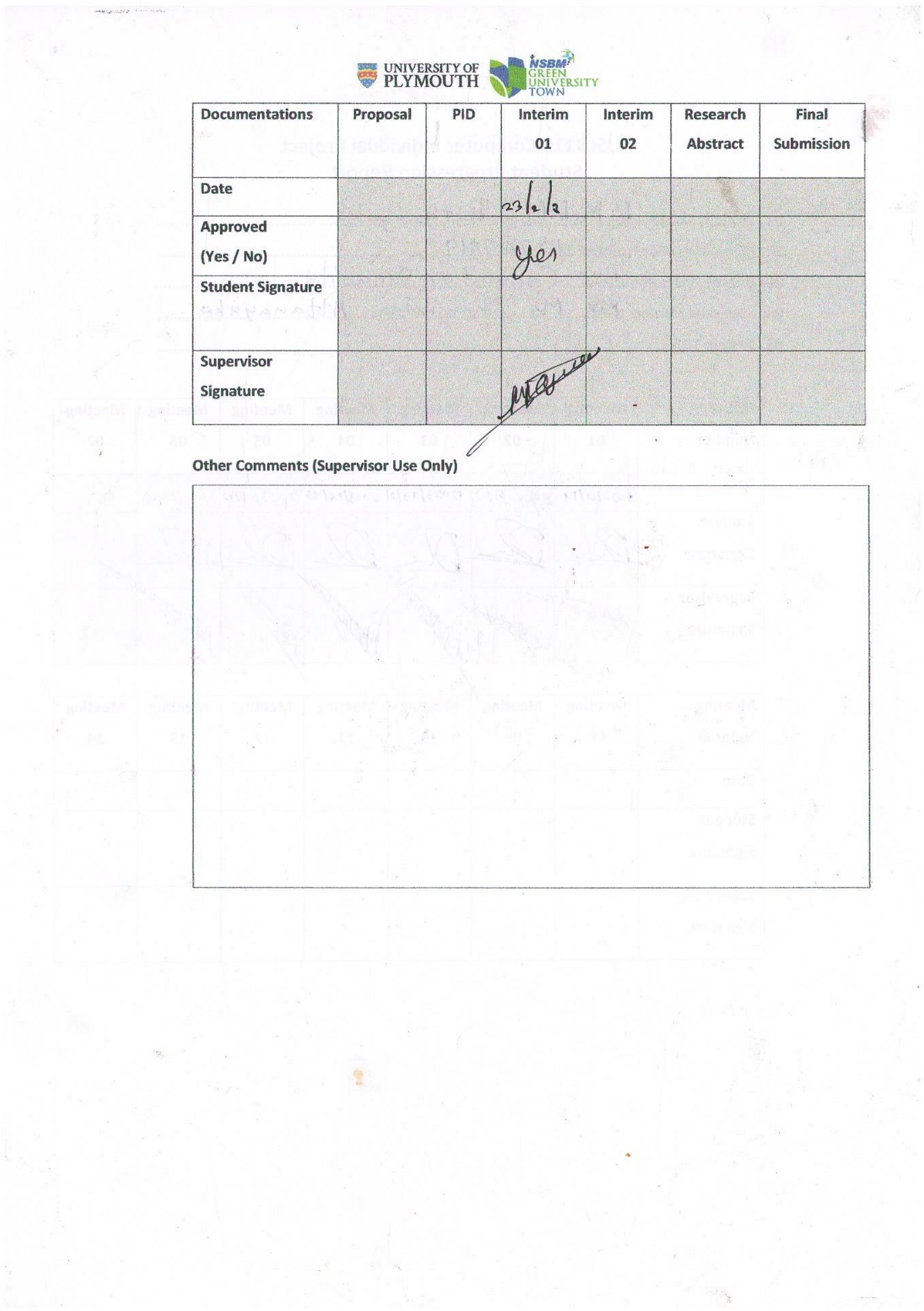
**PUSL3119 Computer Individual Project**
**Student Progression Report**

01. **Student Name** R.M.R.M.L.Rathnayaka
02. **Plymouth Index Number** 10747919
03. **Degree Program** Bsc Computer Security
04. **Supervisor Name** Mr Chamindra Attanayake
05. **Project Title**

| Meeting Number | Meeting 01 | Meeting 02 | Meeting 03 | Meeting 04 | Meeting 05 | Meeting 06 | Meeting 07 |
|---|---|---|---|---|---|---|---|
| Date | 2022/10/14 | 3/03/2023 | 2023/03/21 | 2023/04/28 | 2023/05/12 | 2023/05/16 | |
| Student Signature | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | |
| Supervisor Signature | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | |

| Meeting Number | Meeting 08 | Meeting 09 | Meeting 10 | Meeting 11 | Meeting 12 | Meeting 13 | Meeting 14 |
|---|---|---|---|---|---|---|---|
| Date | | | | | | | |
| Student Signature | | | | | | | |
| Supervisor Signature | | | | | | | |

*Figure 3 - Student Progression Report 1*



UNIVERSITY OF PLYMOUTH

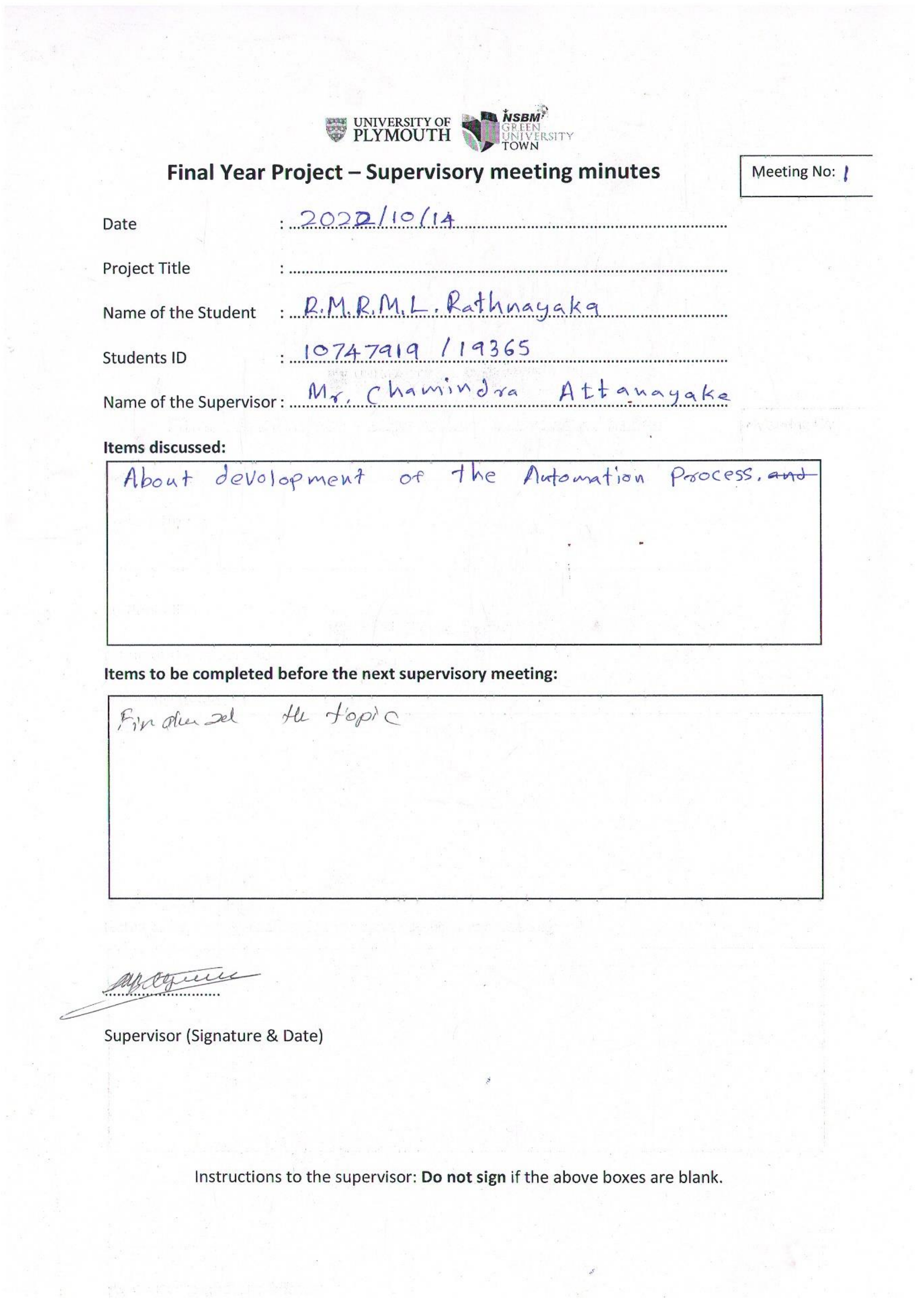
NSBM GREEN UNIVERSITY TOWN

| Documentations | Proposal | PID | Interim 01 | Interim 02 | Research Abstract | Final Submission |
|---|---|---|---|---|---|---|
| Date | | | 23/2/2 | | | |
| Approved (Yes / No) | | | Yes | | | |
| Student Signature | | | | | | |
| Supervisor Signature | | | [illegible] | | | |

Other Comments (Supervisor Use Only)

| |
|---|
| |

Figure 4 - Student Progression Report 2



UNIVERSITY OF PLYMOUTH | NSBM GREEN UNIVERSITY TOWN

**Final Year Project – Supervisory meeting minutes**

Meeting No: 1

| | |
|---|---|
| Date | : 2022/10/14 |
| Project Title | : |
| Name of the Student | : R.M.R.M.L. Rathnayaka |
| Students ID | : 10747919 / 19365 |
| Name of the Supervisor | : Mr. Chamindra Attanayake |

**Items discussed:**

About devolopment of the Automation Process. ~~and~~

**Items to be completed before the next supervisory meeting:**

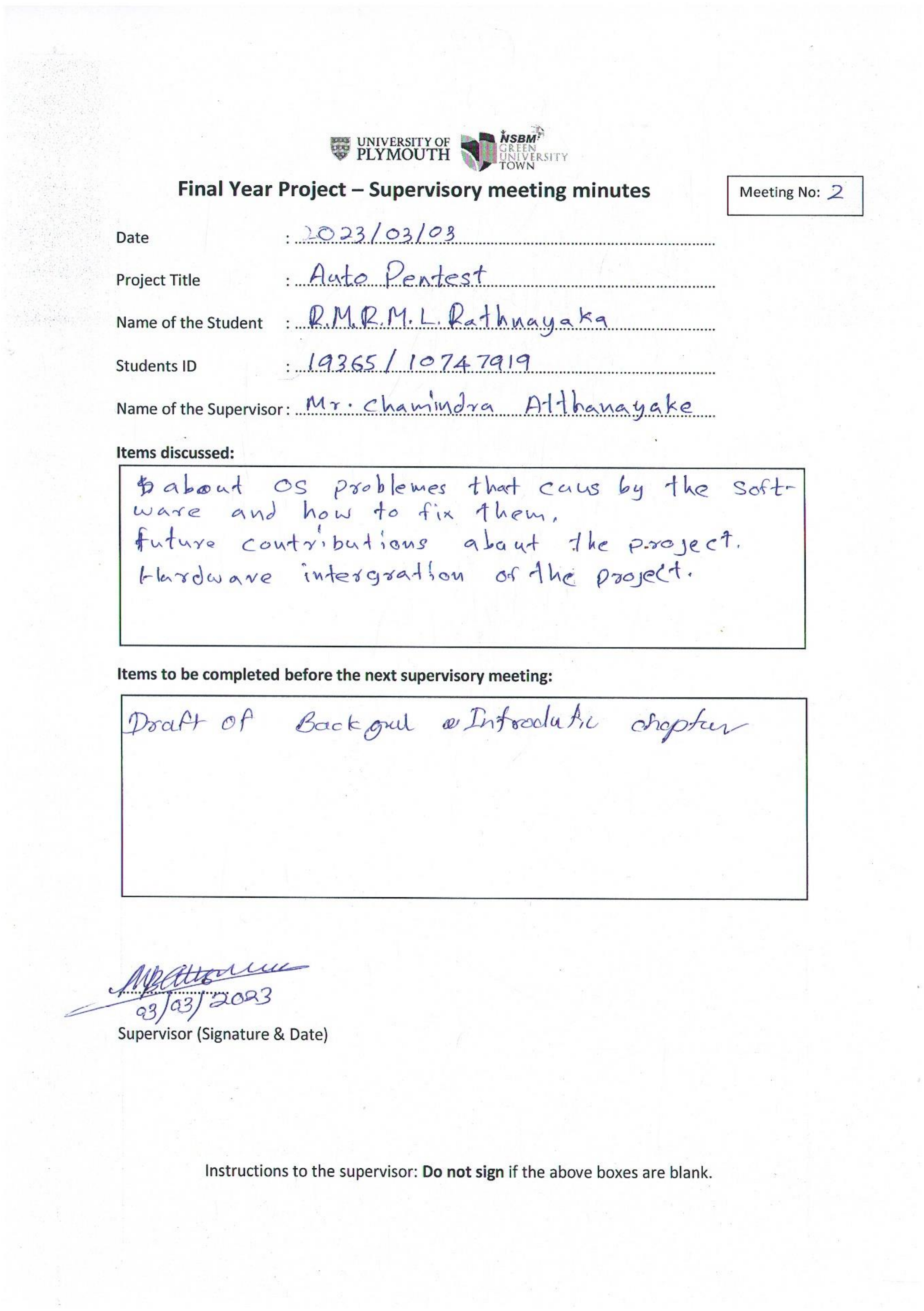
Finalised the topic

[illegible]

Supervisor (Signature & Date)

Instructions to the supervisor: **Do not sign** if the above boxes are blank.

*Figure 5 - Meeting Minutes Document 1*



UNIVERSITY OF PLYMOUTH NSBM GREEN UNIVERSITY TOWN

**Final Year Project – Supervisory meeting minutes**

Meeting No: 2

| | |
|---|---|
| Date | : 2023/03/03 |
| Project Title | : Auto Pentest |
| Name of the Student | : R.M.R.M.L. Rathnayaka |
| Students ID | : 19365 / 10747919 |
| Name of the Supervisor | : Mr. Chamindra Atthanayake |

**Items discussed:**

about OS problemes that caus by the software and how to fix them.
future contributions about the project.
Hardware intergration of the project.

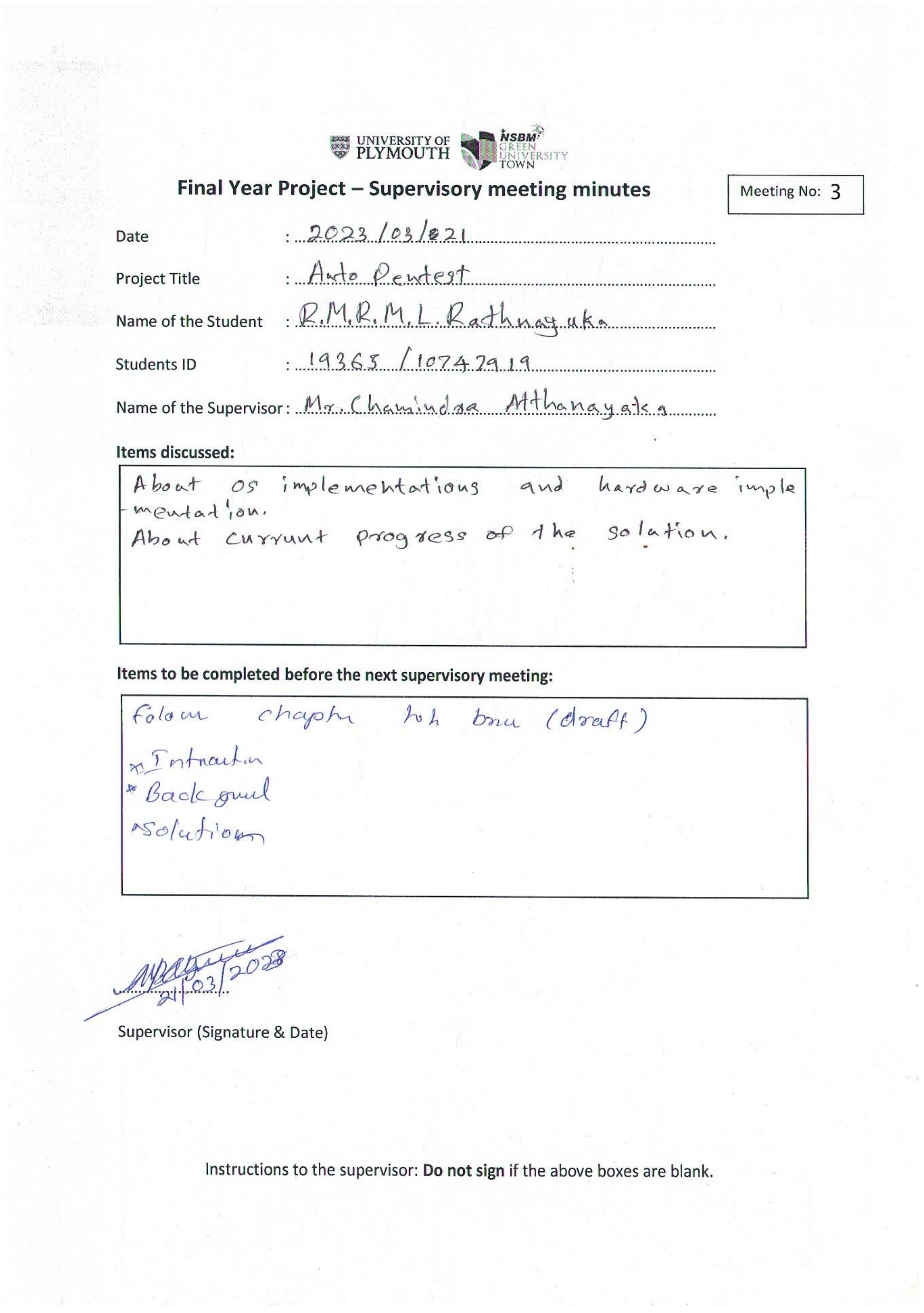
**Items to be completed before the next supervisory meeting:**

Draft of Backgrd & Introductic chapter

[illegible]
03/03/2023
Supervisor (Signature & Date)

Instructions to the supervisor: **Do not sign** if the above boxes are blank.

*Figure 6 - Meeting Minutes Document 2*



UNIVERSITY OF PLYMOUTH NSBM GREEN UNIVERSITY TOWN

**Final Year Project – Supervisory meeting minutes**

Meeting No: 3

| | |
|---|---|
| Date | : 2023/03/021 |
| Project Title | : Auto Pentest |
| Name of the Student | : R.M.R.M.L. Rathnayaka |
| Students ID | : 19363 / 10747919 |
| Name of the Supervisor | : Mr. Chaminda Athanayaka |

**Items discussed:**

About OS implementations and hardware imple-mentation.
About currunt progress of the solution.

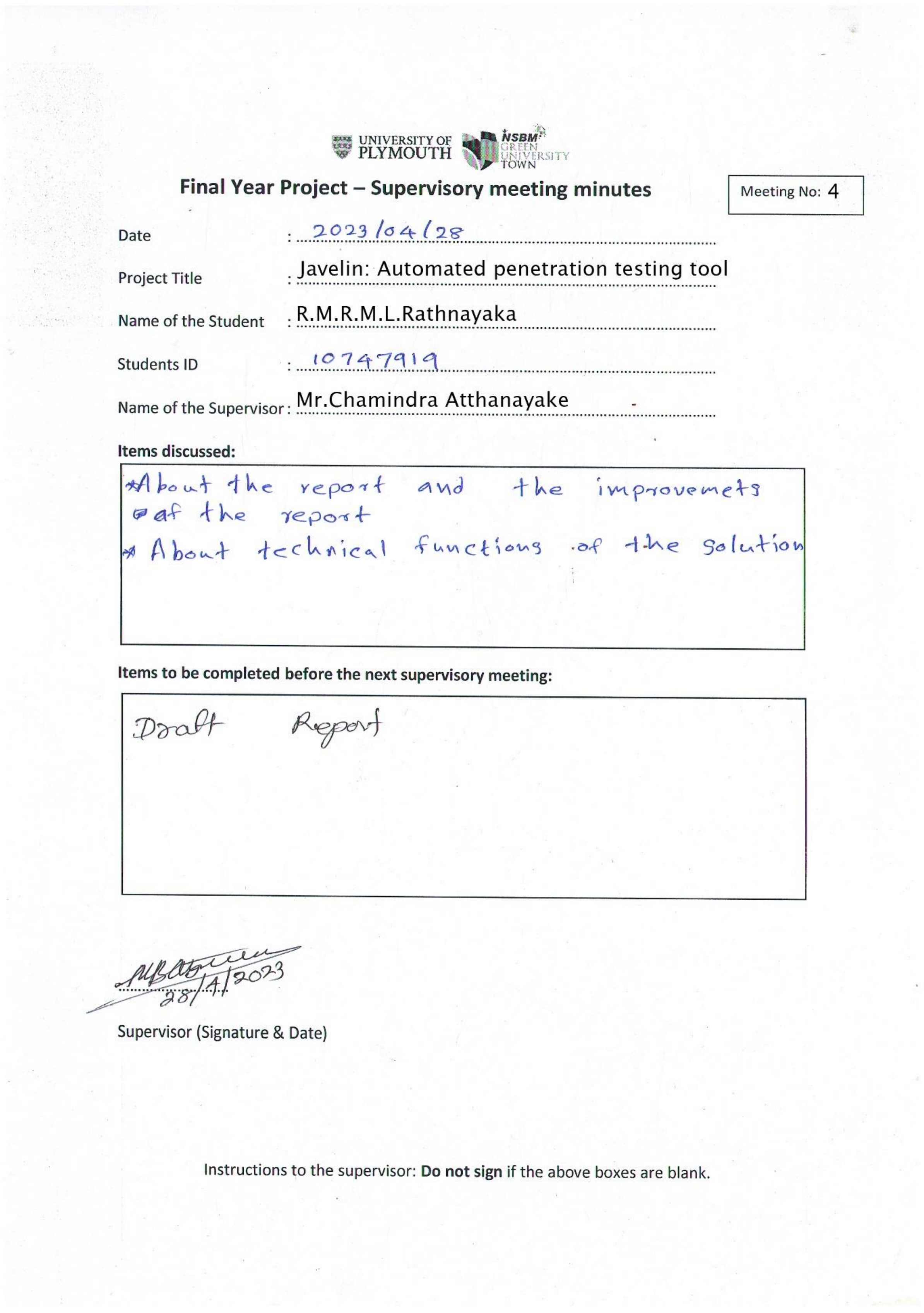
**Items to be completed before the next supervisory meeting:**

Folow chapter [illegible] [illegible] (draft)
* Intraction
* Backgroud
* solution

[illegible] 21/03/2023
Supervisor (Signature & Date)

Instructions to the supervisor: **Do not sign** if the above boxes are blank.

*Figure 7 - Meeting Minutes Document 3*



UNIVERSITY OF PLYMOUTH NSBM GREEN UNIVERSITY TOWN

**Final Year Project – Supervisory meeting minutes**

Meeting No: 4

| | |
|---|---|
| Date | : 2023/04/28 |
| Project Title | : Javelin: Automated penetration testing tool |
| Name of the Student | : R.M.R.M.L.Rathnayaka |
| Students ID | : 10747919 |
| Name of the Supervisor | : Mr.Chamindra Atthanayake |

**Items discussed:**

* About the report and the improvemets of the report
* About technical functions of the solution

**Items to be completed before the next supervisory meeting:**

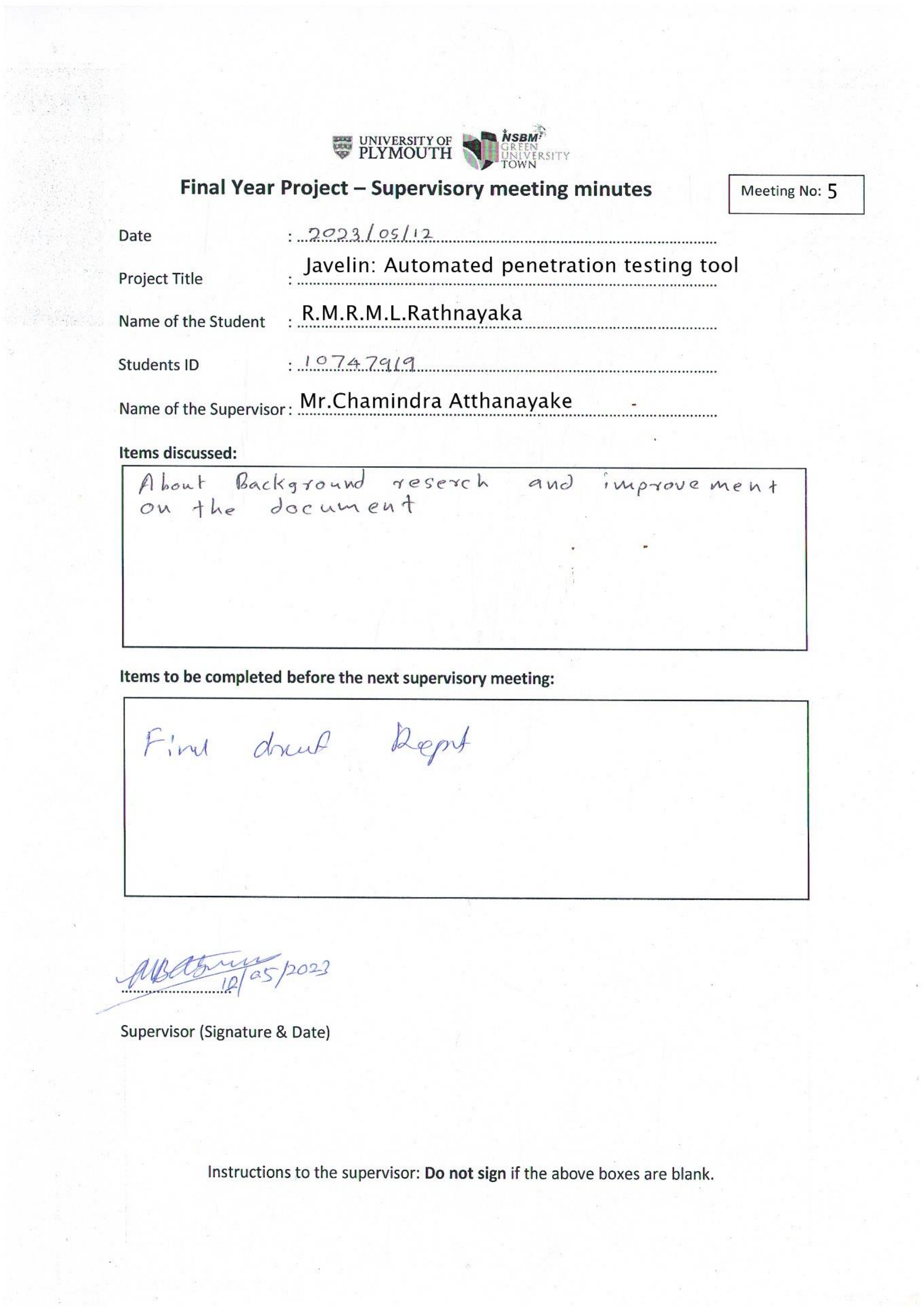
Draft Report

[illegible] 28/4/2023

Supervisor (Signature & Date)

Instructions to the supervisor: **Do not sign** if the above boxes are blank.

*Figure 8 - Meeting Minutes Document 4*



UNIVERSITY OF PLYMOUTH | NSBM GREEN UNIVERSITY TOWN

**Final Year Project – Supervisory meeting minutes**

Meeting No: 5

| | |
|---|---|
| Date | : 2023/05/12 |
| Project Title | : Javelin: Automated penetration testing tool |
| Name of the Student | : R.M.R.M.L.Rathnayaka |
| Students ID | : 10747919 |
| Name of the Supervisor | : Mr.Chamindra Atthanayake |

**Items discussed:**

About Background reserch and improvement on the document

**Items to be completed before the next supervisory meeting:**

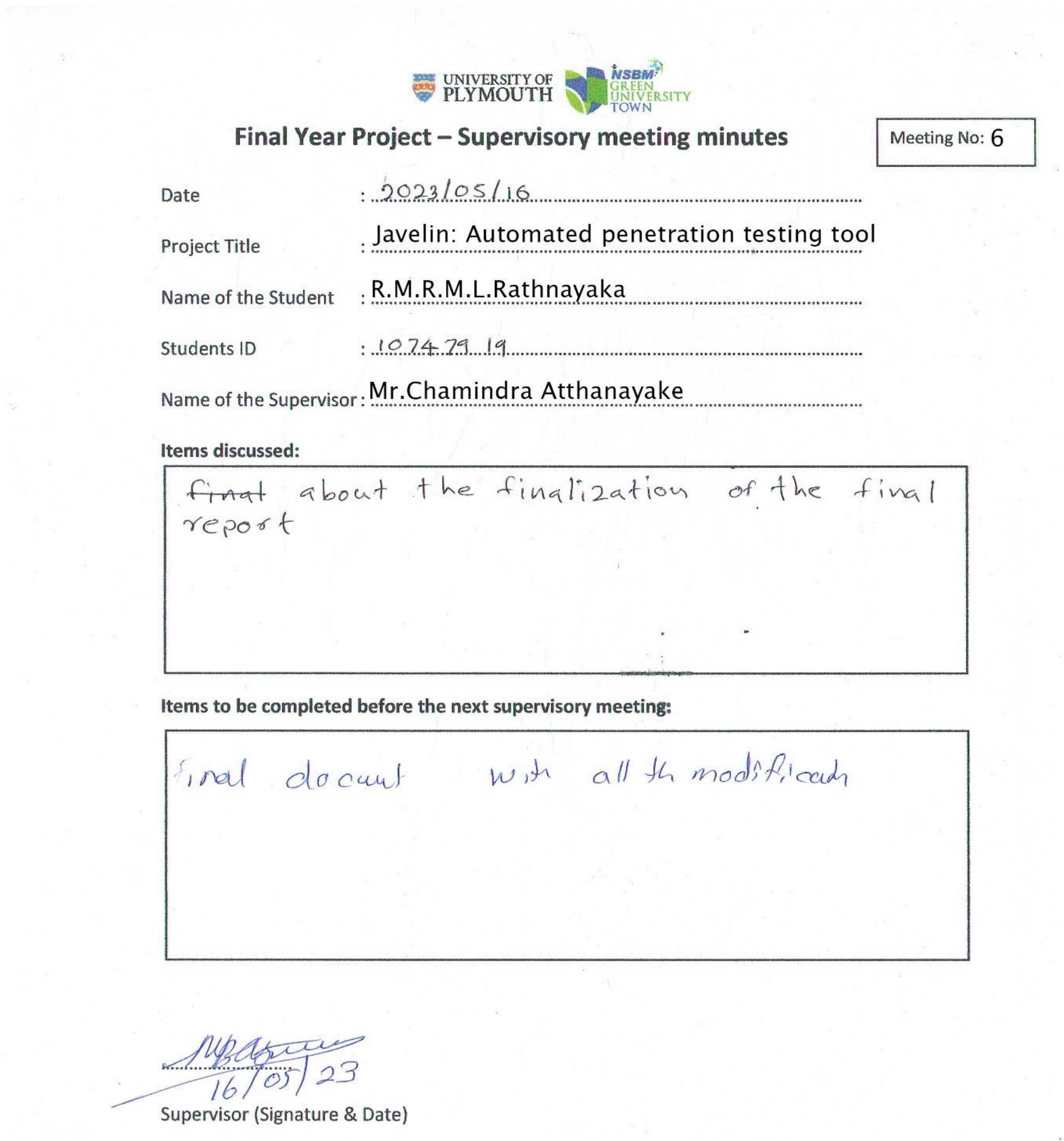
Final draft Report

[illegible] 12/05/2023

Supervisor (Signature & Date)

Instructions to the supervisor: **Do not sign** if the above boxes are blank.

*Figure 9 - Meeting Minutes Document 5*



UNIVERSITY OF PLYMOUTH

NSBM GREEN UNIVERSITY TOWN

**Final Year Project – Supervisory meeting minutes**

Meeting No: 6

| | |
|---|---|
| Date | : 2023/05/16 |
| Project Title | : Javelin: Automated penetration testing tool |
| Name of the Student | : R.M.R.M.L.Rathnayaka |
| Students ID | : 10747919 |
| Name of the Supervisor | : Mr.Chamindra Atthanayake |

**Items discussed:**

~~final~~ about the finalization of the final report

**Items to be completed before the next supervisory meeting:**

final docunt with all th modificadn

[illegible]
16/05/23

Supervisor (Signature & Date)

Instructions to the supervisor: **Do not sign** if the above boxes are blank.

*Figure 10 - Meeting Minutes Document 6*

# 15 Testing

Using Nmap, Javelin collects target server information such as OS type, open port, product name, and product version. As a result of Nmap, Javelin can extract the following information.

| | | |
|---|---|---|
| Linux | 21 | vsftpd |
| Linux | 22 | ssh |
| Linux | 23 | telnet |
| Linux | 25 | postfix |
| Linux | 53 | bind |
| Linux | 80 | apache |
| Linux | 5900 | vnc |
| Linux | 6667 | irc |
| Linux | 8180 | tomcat |

Using trained data and identified product information, the Javelin executes the exploit on the first target server.

It is capable of carrying out exploits at a precise location.

If the exploitation is successful, a session will be established between the Javelin and the first target server.

The pivoting is carried out by Javelin using an open session in exploitation.

Following that, Javelins that do not have a direct connection to the internal server can run exploits via the first server. As a result, Javelin is repeatedly scanning the internal server for post-exploitation via the compromised server.

****end of the final report****