Setting the Standard: Implementing a Standardised Report Writing Tool

Adam Board

BSc Ethical Hacking (Hons), 2023-2024

Supervised by

Mr. Jamie O'Hare

Abstract

Penetration testing is an important aspect of information assurance for determining if an organisation has implemented appropriate security measures to protect any sensitive data. However, the penetration test reports which organisations receive fail to communicate critical information to a non-technical audience. To combat these problems, standardised penetration test reports and tools were designed, but only a limited number of open-source tools have been identified. A survey conducted reported 50% of the participants found a lack of open-source report writing tools. A further 69% of participants emphasised report writing as an essential factor in penetration testing tools. This project aims to design and implement a standardised penetration test report writing tool and evaluate its effectiveness.

To understand what makes a good penetration test report, three report templates were examined. Aspects of each template were merged to develop a standardised template. To identify the current quality of penetration test reports, twenty reports from various organisations were analysed against the standardised template. Based on the critical analysis, a finalised report template was developed. To ensure the guidelines of the report template are being adhered to, a performance review report was produced to give feedback to the penetration tester. A report writing tool with all the necessary features to produce a penetration test report was created based on report writing tools available online.

Overall, this project resulted in the creation of an open-source report writing tool that combines the features from other report writing tools and builds upon them to produce an effective tool for improving the quality of penetration test reports. Similarly to PlexTrac, the integrated vulnerability databank improves the efficiency of report writing. A component that largely affects the consistency and effectiveness of a report is the penetration testers writing the report. To resolve this issue, the report writing tool is bundled with a performance review report to provide feedback after each penetration test report created. However, there are limitations to the report writing tool which would require more resources and better knowledge of the programming languages used. A concern is that the tool has not been designed with secure coding practice in mind. This could lead to the report writing tool potentially containing vulnerabilities. The tool also does not factor in environmental or temporal scores with the built-in CVSS calculator. The calculator could be replaced with a custom-built calculator component that includes base, environmental, and temporal group scores to improve severity accuracy.

Acknowledgements

I would like to thank my phenomenal supervisor Jamie O'Hare, for his unrelenting support and advice over this past year. The substantial feedback provided by Jamie has made the process of completing my dissertation considerably easier with less issues than what would have occurred had I been completing this on my own.

I would also like to thank my partner, Melina Garcia Ayala, for always supporting me throughout the entirety of university. Especially during deadline season when stress was at its highest point. I could not have accomplished many of the achievements that I have earned through university without my partner pushing me to be the best version of myself.

Finally thank you to my friends and family for listening to me discuss my dissertation ideas and reminding me to take breaks occasionally. Especially during the final few months of university, I could not have accomplished as much as I have without my friends and families' support. An especially huge thank you to Allan as this project would not have been accomplished without their guidance and support during the development of the report writing tool.

Table of Contents

Table of Figures

1. Introduction

1.1. Background

Penetration testing is an important aspect of information assurance for determining if an organisation has implemented appropriate security measures to protect any sensitive data relating to customers and staff. Conducting regular penetration tests allows an organisation to mitigate potential financial or reputational damage that typically occurs after a cyber-attack. To aid in discovering vulnerabilities a penetration tester will introduce automated tools into the testing process. Which improves the effectiveness of penetration tests and reduces the time to identify vulnerabilities (Farah Abu-Dabaseh, 2018). Upon completion of a penetration test, the tested organisation receives a report to relay the key findings from the penetration test, such as vulnerabilities discovered and possible mitigations. The penetration test report will go through development stages and multiple iterations of reviewing and finalisation before reaching a client to reduce the chance of errors appearing within the report, such as spelling mistakes and formatting errors. Figure 1 shows the development stages a typical penetration test report will undergo.

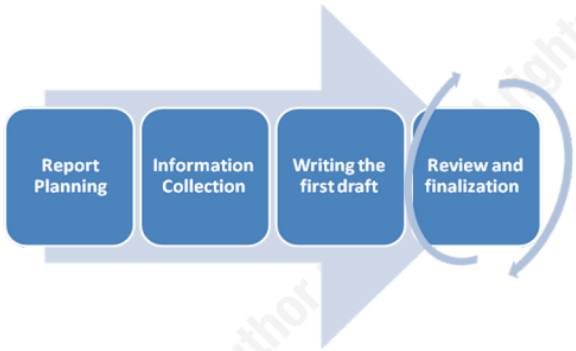


Figure 1 Report Development Stages (Alharbi, 2010).

The importance of penetration testing continues to grow as organisations rely on information technology resources to expand their development. However, an ongoing problem with penetration testing is the vocabulary used to communicate information. The penetration test reports which organisations receive fail to communicate critical information to a non-technical audience. Top-level management in organisations on several occasions have struggled to comprehend the criticality and severity of vulnerabilities listed within penetration test reports (Mohd Nizam Zakaria, 2019). This lack of comprehension can potentially lead to critical vulnerabilities being left unresolved within an organisation's system. Conversely, previous research has suggested that penetration testers find report writing tedious, complex, and time consuming (Alghamdi, 2021). To combat the current problems with report writing, numerous standardised penetration test reports and tools were designed. However, the effectiveness of the proposed designs remains unclear. There has not been a study to determine the impact of implementing the mentioned report formats.

To date, only a limited number of open-source penetration test report writing tools have been identified. However, there are a variety of report writing tools which are closed source or developed by an in-house team for a penetration testing organisation. But these tools are all vastly different in layout and functionality which makes it difficult for employees due to the constant switching of report writing tools. A survey conducted about penetration testing tools reported 50% of the participants found a lack of open-source report writing tools. A further 69% of participants emphasised report

writing as an essential factor in penetration testing tools due to potential compliance requirements that are needed to satisfy regulation and industry needs (Kousik Barik, 2021). Implementing report writing tools increase both the efficiency and consistency of writing reports. Although, report writing tools have no evidence in aiding the formatting of reports or the vocabulary used within reports. Examining existing penetration testing tools and penetration test reports may aid in identifying the current limitations in the report writing stage of penetration testing.

1.2. Aim

This project aims to design and implement a standardised penetration test report writing tool and evaluate its effectiveness.

1.3. Research Questions

1. How do current tools increase the consistency and efficiency of penetration test report writing?
2. What current standards and components should be taken into consideration to produce consistent and effective penetration test reports?
3. How can implementing a standardised penetration test report writing tool ensure consistency and increase the effectiveness in communicating findings to clients?

1.4. Objectives

- Conduct an extensive review of penetration test reports to identify a penetration tests key components and the standards for scoring vulnerabilities.
- Propose a standardised template report, informed by the extensive review.
- Develop a standardised report writing tool. The tool will make use of the template to ensure each report follows the same procedure.
- Evaluate the effectiveness of the standardised tool and relate the findings to previous work found within the literature review.

1.5. Structure

The next section in this dissertation is a literature review, which provides a background and critical analysis of previous and currently used technologies in the field of report writing, automated tools for penetration testing, and methods of standardisation. The methodology section focuses on the explanations and justifications for why each artefact was created to carry out this project. It also explains the methods used to develop the three artefacts. Results are presented in section three, aligning with the project's aims and being analysed accordingly. The remaining sections are the discussion and conclusion. The discussion evaluates the findings from this project and comments on its significance in relation to the literature review. The conclusion showcases the implications that were discovered throughout developing the project and how this project could be continued in the future.

2. Literature Review

2.1. Automation Tools

Automated tools have been implemented in numerous ways to speed up the process of completing a penetration test. This reduces the complexity and difficulty of penetration testing for organisations without a dedicated security expert (Farah Abu-Dabaseh, 2018). Manual penetration testing is time-consuming, complex, and entirely dependent on the skillset of the penetration tester. Automated tools have aided in bridging the skill gap that is present among penetration testers (Kousik Barik, 2021). automated tools have their positives and drawbacks compared to manual penetration testing. This can be seen in Table 1.

Table 1 Comparison between manual and automated penetration testing (Kousik Barik, 2021)

Phases	Manual	Automated
Testing Phases	Manual nonstandard process, high cost in customization. process, high cost in customization.	Standard process, fast in time.
Attack Database Management	Dependencies on the public database and essential to maintain the database manually.	Updates are available for the attack database
Reporting	Manual process of collection of data.	Customization of reports based on requirements and available centrally.
Network Identification	No change is needed in systems.	Different systems modifications are required.
Auditing	Slow, complex to manage, and often inaccurate process.	Records all activity automatically
Exploit Development and Management	Maintaining an exploit database is very difficult, and public exploits can be unsafe to run.	Product vendors maintain exploits and are easier to manage. In addition, exploits are developed by professional experts and thoroughly tested.
Training	More straightforward, to train users compared to manual testing.	Training can be customized but time-consuming

While the table highlights the benefit of customisability for reports through automation, it is still ineffective at report generation. Automated reports provide an overload of information without structuring it out in a format that is easy for a non-technical user to read and understand (Salim, et al., 2022). Nessus, an automated scanner and report generator, is an example of an automated tool

producing reports that are overloaded with information. Non-technical users would struggle to go through a Nessus report and gather the relevant information required to introduce appropriate mitigations for any discovered vulnerabilities. Although, the ineffectiveness could be due to no standard formatting being present for penetration test reports.

To combat the limitations of current automation tools, new tools should be created with the specific purpose of overcoming these limitations, or the current tools should be updated and refined to provide solutions to their limitations (Farah Abu-Dabaseh, 2018). This is especially important for tools which are focused on aiding in report writing. There are very few studies which investigate systematically enhancing remediation and reporting features to improve a tools usage (Harrell, et al., 2018). Focusing on improving the remediation and reporting features could mitigate the limitation mentioned previously regarding Nessus' overload of information when generating reports.

Dradis is an example of a tool which focuses on aiding the report writing section of a penetration test. A study on "Red Teaming Service Learning" discovered that students on the study managed to improve the quality and efficiency in their report writing by using Dradis to collate information centrally and generate a partial report they can add onto to complete it (Young, 2020).

2.2. Report Writing

The penetration test report given to organisations after a penetration test can be created through various methods: from automated generation, to manually typing up the report. For creating reports there is a guideline that a client can follow for requesting information: the ISO 29147, which focuses on the information that should be requested in a report. This includes the risk assessment with details of identified threats including a qualitative level of risk (low, medium, high, or critical), the time and date of discovery, the version information, and a technical description of what was performed to discover the threat (ISO, 2018). However, research into penetration testing had discovered that many penetration test reports did not follow these standards, with comments about the structure of the report from clients such as "Generally hit and miss" and "Appalling". (Knowles, et al., 2016).

Penetration testers could look towards software development for improving report writing quality and structure due to the overlapping topics and issues found in both penetration reports and bug reports. In a good bug report the tester should provide a full description of how and where the bug was found with steps to reproduce the bug, as well as the impact of the bug (Bettenburg, et al., 2010). Similarly, penetration test reports should provide a full description of how and where a vulnerability was found with steps to reproduce the vulnerability, as well as the severity of the vulnerability. Both penetration test and bug reports suffer from poor structure and the use of complicated terminology that a top-level executive would not be able to understand.

Penetration test reports should be able to satisfy both top-level executives and security system personnel to be an effective report. However, current reports were not meeting this goal and top-level executives could not understand the severity and impact the vulnerabilities could cause (Mohd Nizam Zakaria, 2019). The lack of understanding happens when complex and technical language is used within high-level sections, such as the executive summary. To combat this, penetration testers should focus on using language suited for a non-technical audience consistently.

The consistency and structure of reports typically relates to the capability of an individual and their ability to relay technical information to a non-technical audience (Knowles, et al., 2016). This issue

could be mitigated by implementing a form of standardised template which novice penetration testers could follow to reduce the formatting errors.

2.3. Standardisation

To establish a level of quality that is similar between different organisations, standards such as ISO 29147 and CHECK were introduced. ISO 29147 provides guidelines for disclosure of potential vulnerabilities in products and online services. It also details how to handle potential issues which may appear from disclosing a vulnerability to an organisation or the public (ISO, 2018). CHECK is the list of National Cyber Security Centre (NCSC) approved penetration test organisations and the methodology they use to conduct penetration tests. The employees conducting the penetration tests must also hold NCSC approved qualifications and produce reports to a specific standard of quality (National Cyber Security Centre, 2023). While these standards include guidelines on report writing, many penetration testers still find difficulty translating the findings of a penetration test to qualitative information that is useful and comprehensible for top level executives (Sharma, et al., 2023).

Critical and high severity vulnerabilities with possible mitigations should be clearly identifiable as urgent by top-level executives to ensure they are dealt with immediate action (McGready, 2023). The Common Vulnerability Scoring System (CVSS)is a method of calculating the severity of a vulnerability and displaying it as a simple qualitative rating of low, medium, high, or critical. The rating is calculated through three metric groups which contain questions about the context of the vulnerability. This can be seen in Figure 2.

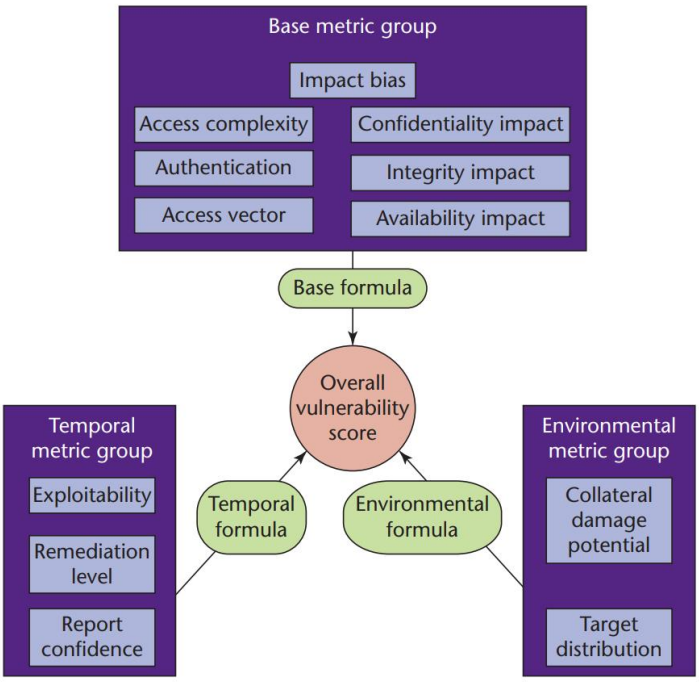


Figure 2 CVSS Metric Groups (Mell, et al., 2006).

Nessus utilises CVSS as its method of calculating the severity of a vulnerability because of clients often mandating it to be included in a penetration test report (Knowles, et al., 2016). However, the severity ratings provided for vulnerabilities in a Nessus report are not a fair representation of severity as Nessus

relies solely on the base metric group and does not factor in temporal or environmental metric groups (Kumar, 2014). This could lead to a vulnerability being deemed more critical than necessary and resources could potentially be wasted remediating a vulnerability which is not as severe as it says through Nessus.

To ensure vital information is relayed appropriately, guidelines and templates for report writing should be suggested, similarly to CHECK, to ensure penetration testers within organisations keep up their quality of writing. Although, strict standards should not be set as organisations would oppose this, stating that the variation in reports is the unique selling point of each organisation and provides flexibility (Knowles, et al., 2016).

2.4. Penetration Test Report Template

Organisations that provide penetration testing typically use their own in-house template that they have developed over time. It is important to develop an effective and easy to use template as the service penetration testers provide is useless without a comprehensive and understandable report. Much of the time spent on a penetration test is the report writing, as this is the tangible product provided to organisations to convey the findings and requires multiple drafts before it reaches another organisation (Alharbi, 2010).

Numerous research papers have found that an effective penetration test report is broken down into the executive summary, the objectives and scope, and the detailed findings. The executive summary provides a summary of the key findings in two short paragraphs. The language used must be accessible for executives as they are the ones acting upon any recommendations and mitigations for the discovered vulnerabilities (Mohd Nizam Zakaria, 2019). The objectives and scope aid the client in understanding what type of penetration test was conducted and what devices were included as potential targets for the scope (Alharbi, 2010). The detailed findings describe the vulnerabilities regarding their impact against the organisation should it be exploited, the likelihood of the vulnerability being exploited, and the recommendations to mitigating the vulnerability (Kousik Barik, 2021). Writing these sections can be difficult and tedious, which can lead to varying standards and differences in reports between organisations. Table 2 showcases the different additions organisations implement into the primary sections of their reports to try and relay the findings to a client.

Table 2 Sections within a penetration test report from various organisations

Pen Test Report	Executive Summary						Methodology	Detail Finding				Conclusion	References	Appendixes		
	Scope of Work	*Project Objective*	*Timeline*	*Summary Of Finding*	*Summary Of Recommendation*	*Chart*		*Vulnerabilities Detail & Description*	*Impact*	*Likelihood*	*Recommendation*			*Scanning Result*	*Vulnerability Assessment Result*	*Risk Rating*
[10]	✓	✓		✓		✓	✓	✓			✓		✓		✓	✓
[11]	✓	✓		✓			✓	✓	✓	✓	✓	✓				
[12]	✓						✓	✓	✓		✓	✓				
[13]	✓	✓		✓	✓			✓			✓	✓			✓	
[14]	✓	✓		✓				✓	✓	✓	✓	✓				✓
[15]		✓	✓	✓		✓	✓	✓	✓	✓	✓		✓	✓		
[16]	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓				✓
[17]	✓			✓	✓			✓			✓				✓	✓

The design and formatting of a penetration test report can affect the effectiveness of relaying vital information. Using hard to read fonts and poor colour combinations can lead to unappealing and

ineffective reports, whereas using a simple colour combination with dyslexic friendly fonts can increase the accessibility of the penetration test report. The layout of information is especially important as it influences how a reader will navigate the information present on the report, which means a clear and concise layout will be easier to understand. Additionally, the results from a survey about whether data visualisation improves business insight shown that 74% of 210 respondents agreed that it majorly improves insight if visual aspects are included (Kyrölä, 2022). Based on this, implementing graphs and charts to convey findings can aid the reader in remembering key information. Finally, the report design should be revolved around the target audience to ensure the information is communicated effectively (Murchie & Diomede, 2020).

2.5. Report Performance Review

The quality of the report directly relates to the skillset of the penetration tester writing the report. To improve the skills of a penetration tester, a personal performance review report that identifies strengths and weaknesses can be implemented (Bertoglio, et al., 2022). This would allow a penetration tester to self-evaluate their work with feedback from their line-manager to improve their own skills and become a greater asset to their organisation.

Feedback from the line-manager should showcase the goals the penetration tester should be aiming for to improve their writing capabilities. To effectively provide feedback, the areas of improvement should be written with consideration to the following factors: Relevance, Accuracy, Timeliness, Specificity, and Ease of Understanding (Baker, 2010). Should the penetration tester be able to meet these 5 factors in their improvements, they will potentially increase their performance, which increases the standard of quality in penetration test reports.

2.6. Standardised Report Writing Tool

The implementation of standardised report writing tools has shown improvements in report completeness in other industries. In medical science, “randomised trial” reports typically lead to wasted research time due to reports being incomplete and lacking detail (Barnes, et al., 2015). The completeness of reports is rated on an average scoring from 1-10. Introducing a writing aid tool led to an average increase of 42% in report completeness, from 5.0 to 7.1. Penetration testing could benefit in similar ways to the medical industry by introducing report writing tools., especially in ensuring reports are not lacking in detail.

To standardise the quality of reports, report writing tools have been developed previously. Dradis acts as a central repository to enable information sharing between team members of what has been done and what shall be done during a penetration test. It features report generation, with support for attachments of screenshots and logs. Dradis has shown to improve the efficiency of report writing by importing report writing guidelines from The Penetration Testing Execution Standard (PTES) package that penetration testers can follow (Young, 2020). Snippets of Dradis’ user interface can be seen in Figure 3 and 4.

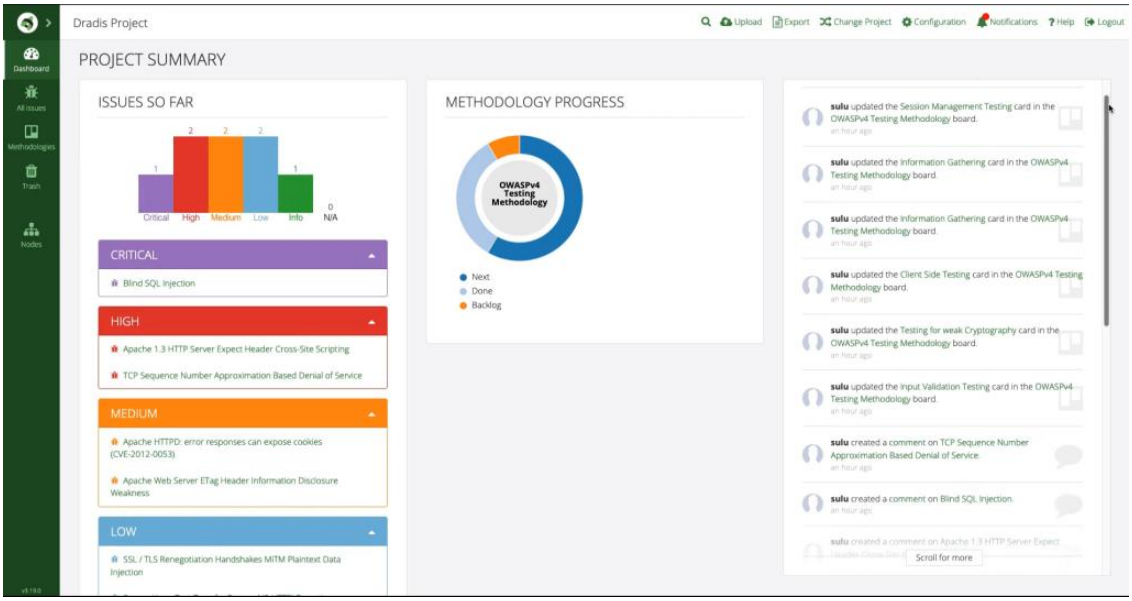


Figure 3 - Dradis User Interface Landing Page

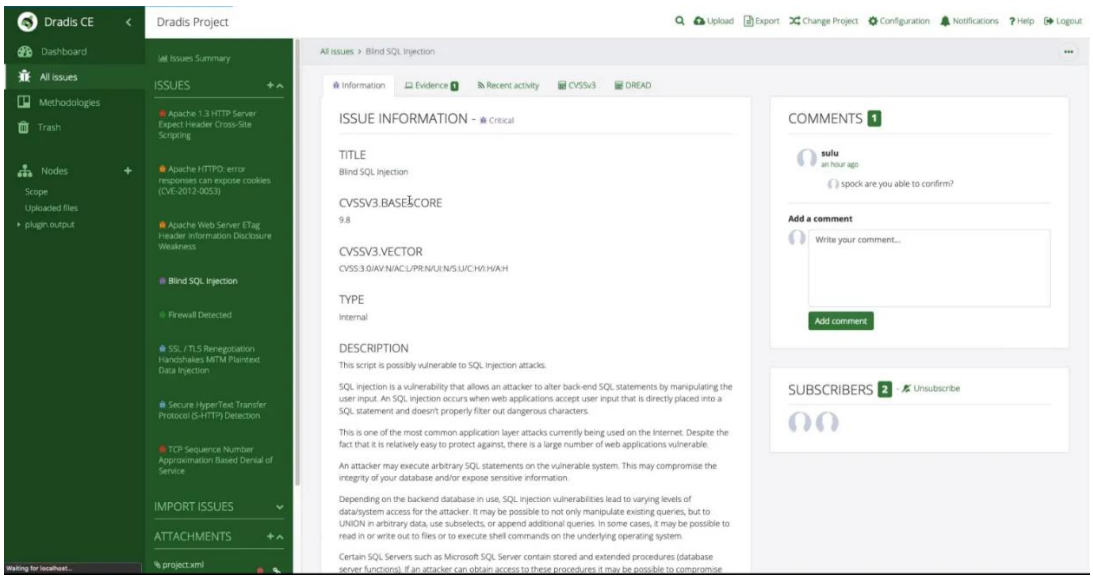


Figure 4 Dradis User Interface Issues Page

However, as it can be seen in Figure 4, Dradis suffers from the same issue of information overload as Nessus does in automated report generation. A solution to the issue would be to allow the penetration testers to write and save copy-and-paste answers for different sections of the report, as well as for specific vulnerabilities. This could allow for an increase in quality while keeping the same level of efficiency as automated generation.

PlexTrac is a report writing tool that implements a solution to the typical automated report generation issue. PlexTrac allows the penetration tester to create or insert a custom template of headings into the tool to allow modularity. As seen in Figure 5, PlexTrac implements a databank solution of vulnerability descriptions that a penetration tester can write up for later use to ensure they can be copy and pasted when required in newly created reports. This mitigates the information overload caused by auto generated vulnerability descriptions. Additionally, PlexTrac provides a built in CVSS calculator that can

attach the CVSS scores to the vulnerabilities added to the report (PlexTrac, 2022). However, the tool is close sourced which means it cannot be modified to suit different organisations requirements.

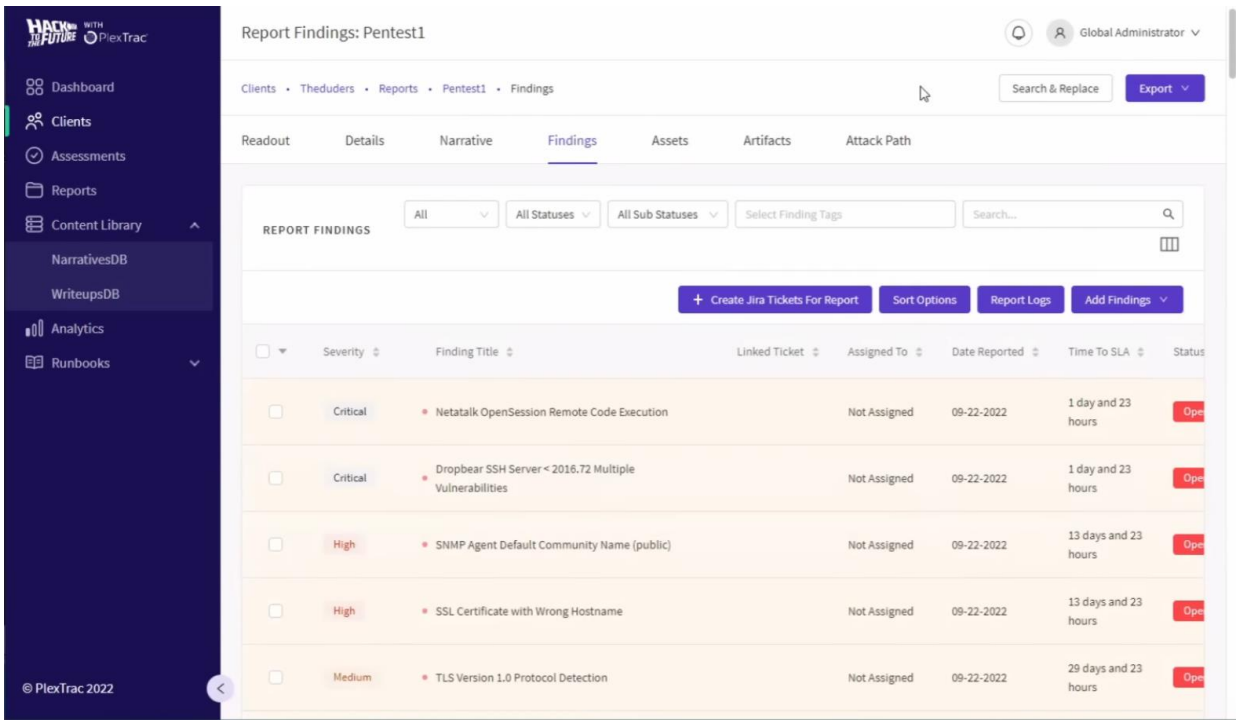


Figure 5 PlexTrac Findings Page

3. Methodology

The methods utilised for developing the three artefacts are described within this section of the paper. Additionally, analysis of penetration testing reports and justifications for decisions made while developing the artefacts can be found in the methodology.

3.1. Analysis of current Penetration Test Reports

To understand what makes a good penetration test report, three report templates were examined from three different research papers (Kousik Barik, 2021) (Mohd Nizam Zakaria, 2019) (Alharbi, 2010). Various aspects of each template that overlapped were merged to develop an effective template with concise descriptions for each section. In Figures 6, 7, and 8, the title of the research paper can be seen at the top of the column. The sections that appear in all three templates can be seen in the colour green. The colour yellow represents the sections that appear in two of the templates. Red represents the sections that appear in only one template. The final report template includes multiple sections from the three templates and can be seen in the results section of this paper.

Penetration Testing Analysis with Standardized Report Generation
Executive Summary
Scope of work
Project Objectives
Timeline
Summary of Findings
Summary of recommendations
Methodology
Planning
Exploitation
Reporting
Detailed Findings
Details of Vulnerabilities
Impact
Likelihood
Recommendations

Figure 6 Report template 1 from penetration testing analysis with standardized report generation

A Review of Standardization for Penetration Testing Reports and Documents
Executive Summary
Scope of work
Project Objectives
Timeline
Summary of Findings
Summary of recommendations
Methodology
Detail Findings
Vulnerabilities
Impact
Likelihood
Risk Evaluation
Recommendations
What should be included through metholdology
Passive OSINT
Covering Tracks
Small section on Industry Practice
Example: CREST

Figure 7 Report template 2 from A review of standardization for penetration testing reports and documents

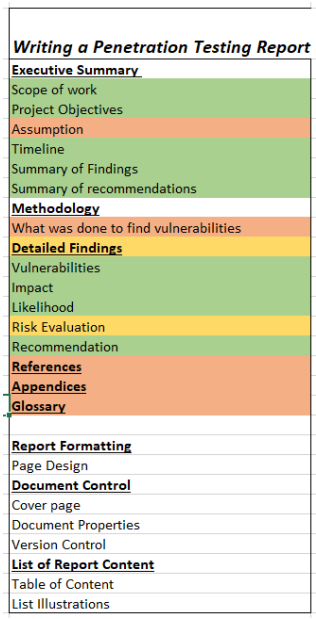


Figure 8 Report template 3 from writing a penetration testing report

To develop a clear understanding of the current quality and standard of penetration test reports, twenty reports from different organisations were obtained through publicly available sources to be analysed thoroughly (Mitev, 2023). A comparison of the proposed template against the twenty reports was conducted to determine the effectiveness of the reports. The reports were given an overall rating out of ten based on a multitude of criteria. The finalised criteria can be seen in the appendix A section of this paper.

The key sections that formulate the overall rating in the criteria include:

- Top-Level Overview
    - Reviews the various summaries of findings such as whether the executive summary is effective at relaying critical information while being concise; Are the objectives of the assessment clear; Is the timeline of the assessment accurate; Does the summary of findings describe the root cause of the different vulnerabilities; Does the summary of recommendations provide enough information to understand how to mitigate the vulnerabilities discovered in the network.

- Methodology
    - A review of how the assessment was carried out, factoring in details such as how the planning of the assessment was completed, what tools were used to exploit or discover the vulnerabilities, and the method of reporting the severity for found vulnerabilities.

- Detailed Findings
    - Reviews the quality of vulnerabilities descriptions within the report, including the description of the vulnerability itself, the impact it could have on the organisation, the likelihood of the vulnerability being exploited, the risk evaluation method, and the recommendations for mitigating the vulnerabilities.

- Additional Importance
    - A review of additional factors such as the use of references and appendices, a glossary for technical terms, the design and formatting of the report, and use of illustrations like graphs to convey vital information.

The different reports' overall ratings can be seen in the results section of this paper.

3.2. Developing the Report Template

Based on the data gathered from the critical analysis of various reports, a report template was developed for implementation into the report writing tool. The next task was to produce a format that is clear and targeted towards non-technical users. Based on the findings from the paper, Fundamentals of graphic design—essential tools for effective visual science communication (Murchie & Diomede, 2020), the report template would employ dyslexic-friendly fonts, a maximum of three fonts, and use illustrations such as graphs, charts, and screenshots to ensure information is easily digestible and clear for the target audience.

The template provided descriptions of what should be included in each section of the report to give guidance to users who are new to penetration test report writing. The descriptions were based on information gathered from the three research papers with templates and the ISO 29147 guidelines for disclosure of potential vulnerabilities in products and online services (ISO, 2018) (Alharbi, 2010) (Kousik Barik, 2021) (Mohd Nizam Zakaria, 2019). Additionally, the subheadings for vulnerability severity were colour coded dark red for critical vulnerabilities, red for high vulnerabilities, orange for medium vulnerabilities, and yellow for low vulnerabilities. The use of colour will highlight the sections as areas of importance within the report to the target audience. This can be seen in Figure 9.

Figure 9 Vulnerabilities Section Colour Coded to Highlight Severity

The font formatting was in accordance with The British Dyslexia Association's guidelines to ensure all information is legible even if the target reader has accessibility requirements. The guidelines state that dyslexic-readable fonts include sans serif fonts such as Arial or Tahoma and should always be font size 12-14 or larger. Additionally, underlining and italics should be avoided within the report as it can make sentences crowded, while bold is endorsed as it can be useful for emphasis without causing crowding (British Dyslexia Association, 2023).

The report was produced on Microsoft Word to confirm the report's design and formatting before being implemented into the application as the default template. The full report can be found in Appendix B.

3.3. Report Performance Review

As stated in numerous research papers, the quality of the report relies heavily on the skills of the penetration tester writing the report (Kousik Barik, 2021) (Bertoglio, et al., 2022). To ensure the skills of the penetration tester improve after each report they produce, a performance review report was developed. Information about the employee such as name, ID, and role can be found on the front page of the report. Additionally, the front page includes the client's name and type of report to aid the employee in identifying which penetration test report the performance review is referencing.

The performance review report contains four sections: The Document control section, the strengths of the report section, the areas of improvement for the report section, and the additional information section. Within each section, a detailed description of what should be included is written. The document control section informs the employee of who reviewed their work with the reviewer's name, role, and contact details included for extra detail. The descriptions within the strengths and areas of improvement sections have been written based on the paper "Employee feedback technologies in the human performance system" to explain how to provide effective feedback that does not negatively impact the employee's motivation and confidence (Baker, 2010). The descriptions also mention to relate the strengths of the report back to weaknesses of a previous report to show that the employee has improved since the last written report. Using the method will boost the employees' confidence and self-fulfilment which can lead to an increase in workflow. Finally, the additional information section is provided as a section for including screenshots that are potentially too large or links to sources that may support the employee's improvement, such as writing courses. The entire performance review report can be seen in Appendix C.

3.4. Standardised Report Writing Tool

Once the reports were finalised the report writing tool could begin its iterative development. However, before developing the report writing tool, source control and backup versions of the project need to be setup in advance. This is because Losing Code due to crashes or corrupted data would have set back the project drastically. multiple repositories and a GitHub project were setup to mitigate the chance of code being unrecoverable due to data corruption or not saving properly. GitHub (GitHub, 2024) project included a kanban board which was utilised for setting up tasks required for this application to be considered feature complete. A screenshot of the Kanban board can be seen in Figure 10.

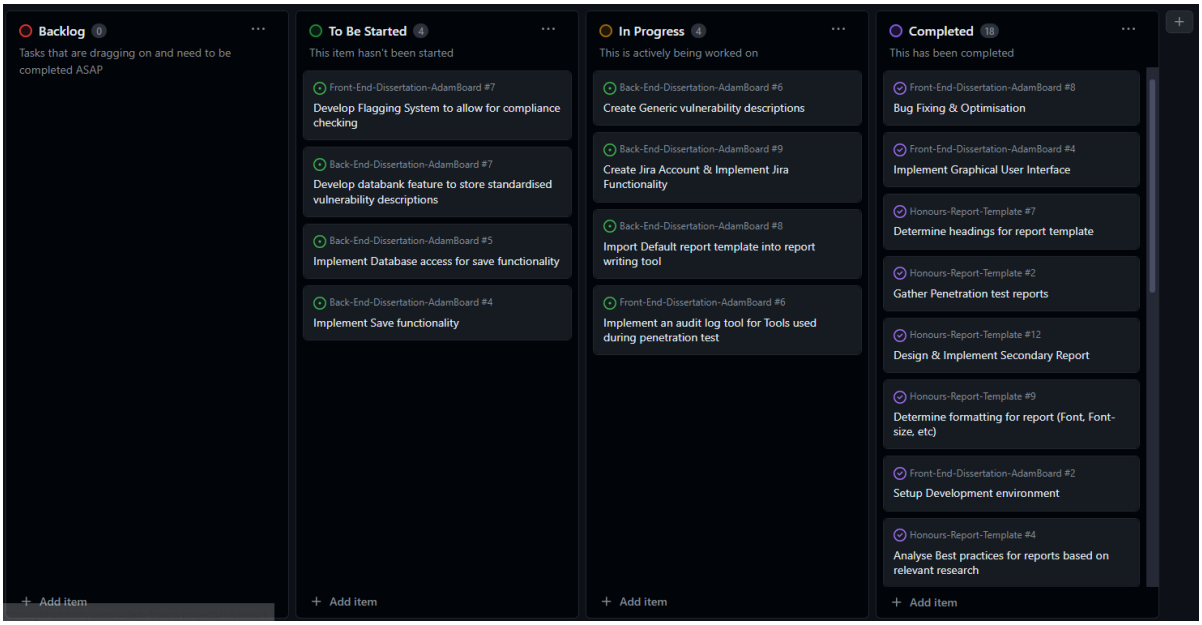


Figure 10 GitHub project Kanban board

The tasks on the GitHub project have been synced up to three different repositories with the names, “Front-End-Dissertation”, “Back-End-Dissertation”, and “Honours-Report-Template”. GitHub repositories allow code to be saved externally from the device which the code is being developed on. A repository also uses a feature called branches. The initial code saved into the repository is known as the master branch. Separate instances of this code known as branches can be created to allow developers to rewrite and experiment with code without affecting the original copy of the code. To mitigate the chance of losing code due to unforeseen circumstances, the repositories were uploaded to after any major changes were implemented. Using Multiple repositories provided flexibility and reduced the size of the download for users since they only need to download the front-end, as the back-end will be on a server that is separate to the user’s machine.

For programming the report writing tool, Visual Studio Code was chosen as the Integrated Development Environment (IDE) (Microsoft, 2024). It was chosen due to the user-friendly layout and customisability available through the vast library of plugins, including GitHub’s plugin for easily creating branches and commits directly through Visual Studio Code’s user interface. Additionally, Visual Studio Code does not require a subscription or one-time payment to access everything it offers. Finally, the built-in debugger and syntax highlighter provides a developer the tools to quickly identify a bug and fix it.

For testing the tool’s functionality, eight vulnerability write-ups ranging from critical to low were produced. The descriptions were based on Nessus’ explanation of the vulnerability and was modified to match the penetration test report template’s vulnerability format. The format and description for each vulnerability was modified to improve the clarity and ensure the description’s language is accessible to the target audience. The report template created previously had also been included for testing various features within the tool, including exporting the headings and vulnerabilities to Microsoft Word as an example report. The example vulnerabilities can be found in Appendix D.

3.4.1. Front-End Development

JavaScript (Mozilla, 2024), combined with the React library (Walke, 2024), was the chosen language for developing the front-end of the report writing tool. The combination of JavaScript and React enabled the ability to develop an interactive and customised user interface while being easy to use and accessible to all modern browsers that support JavaScript. Updates for React are still releasing to this date which shows that this library has plans to be supported into the foreseeable future. As this tool is open source, this provides organisations long-term support to change and modify the code to match their requirements.

The package manager Yarn (Yarn, 2024) was used to install and setup all the required files for the project, as well as run the development server to see the user interface update as changes were made in real-time. Package managers for React have a built-in setup command for setting up the development environment with the latest features in JavaScript. Finally, the package manager can automatically build the finished front-end into a production version which has all the debugging code and functions removed to optimise the entire front-end application.

Before Coding the user interface, a wireframe was created to develop a rough version of what the tool would look like before attempting to produce it through code. The landing page, the report overview page, and three separate editor pages for section headings, vulnerabilities, and notes were required. In Figures 11 and 12, the landing page and report page wireframes showcase a sidebar of button groups, a toolbar. The report page also contains a tabs section to access the three different areas of the tool which needed to be edited.

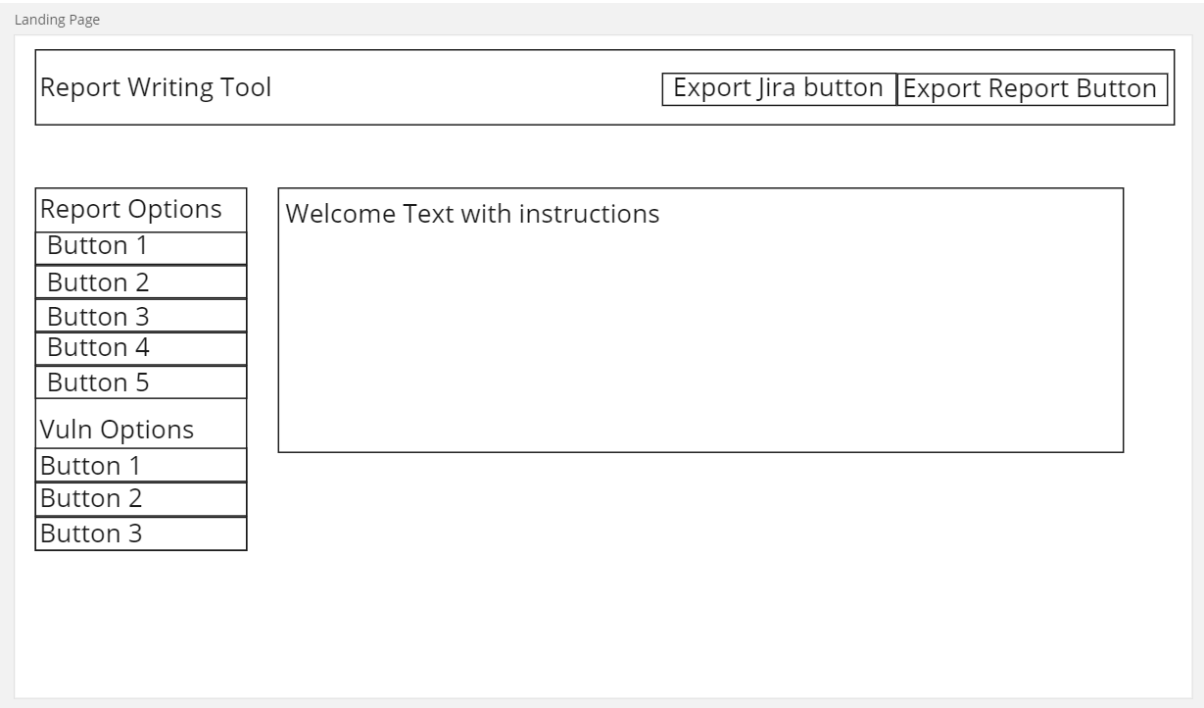


Figure 11 Landing Page of Report Writing Tool

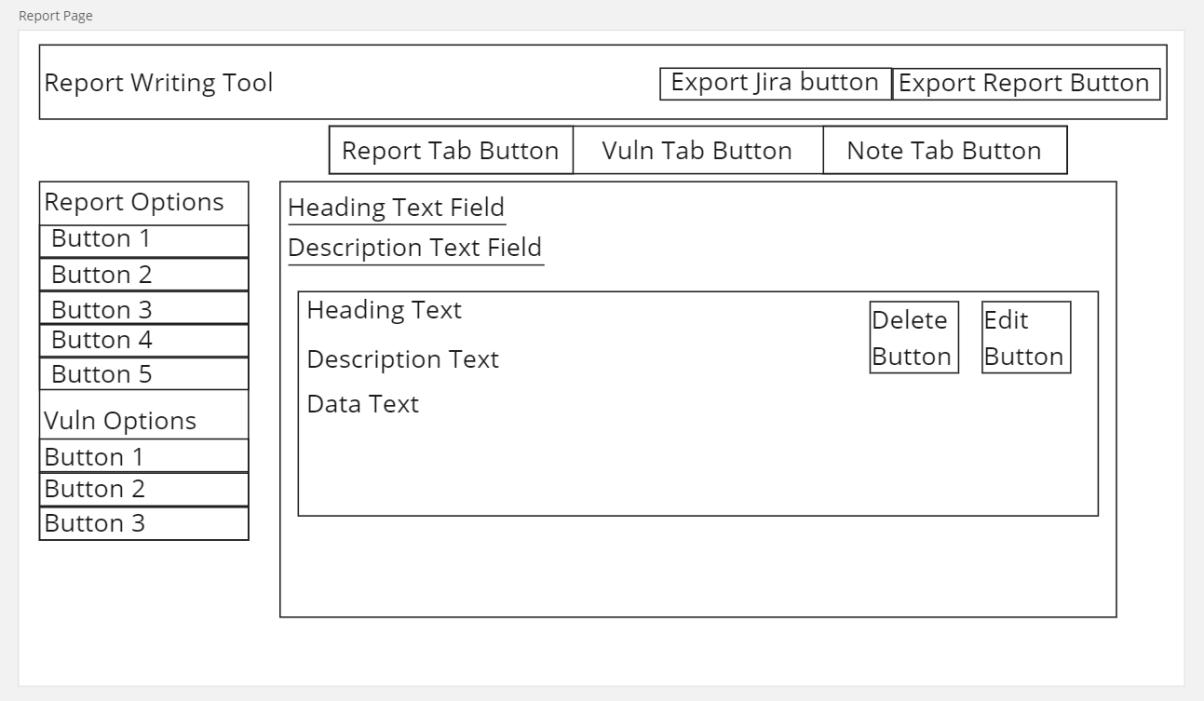


Figure 12 Report Page of Report Writing Tool

The editor pages for the notes and headings were identical while the vulnerabilities editor page had an extra part which included the severity of the vulnerability and a built-in CVSS version 3.1 calculator that is based on the NIST website's calculator (NIST, 2023). The two different editor page wireframes can be seen in Figures 13 and 14.

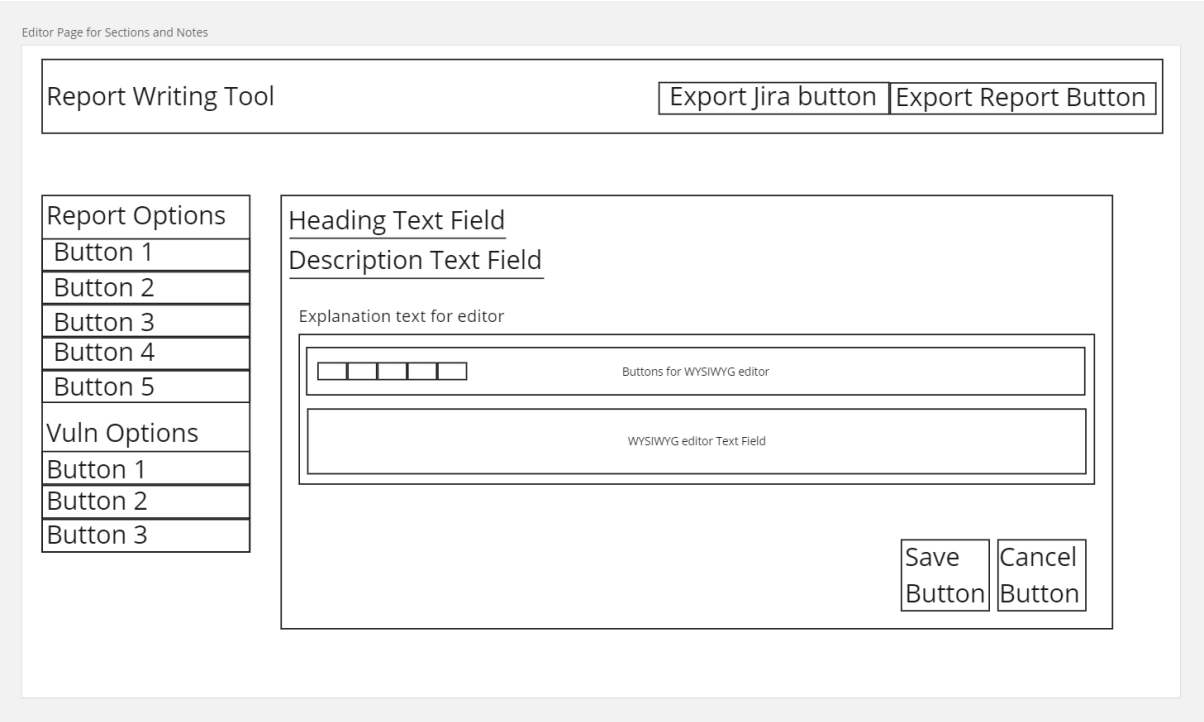


Figure 13 Editor Page for Sections and Notes

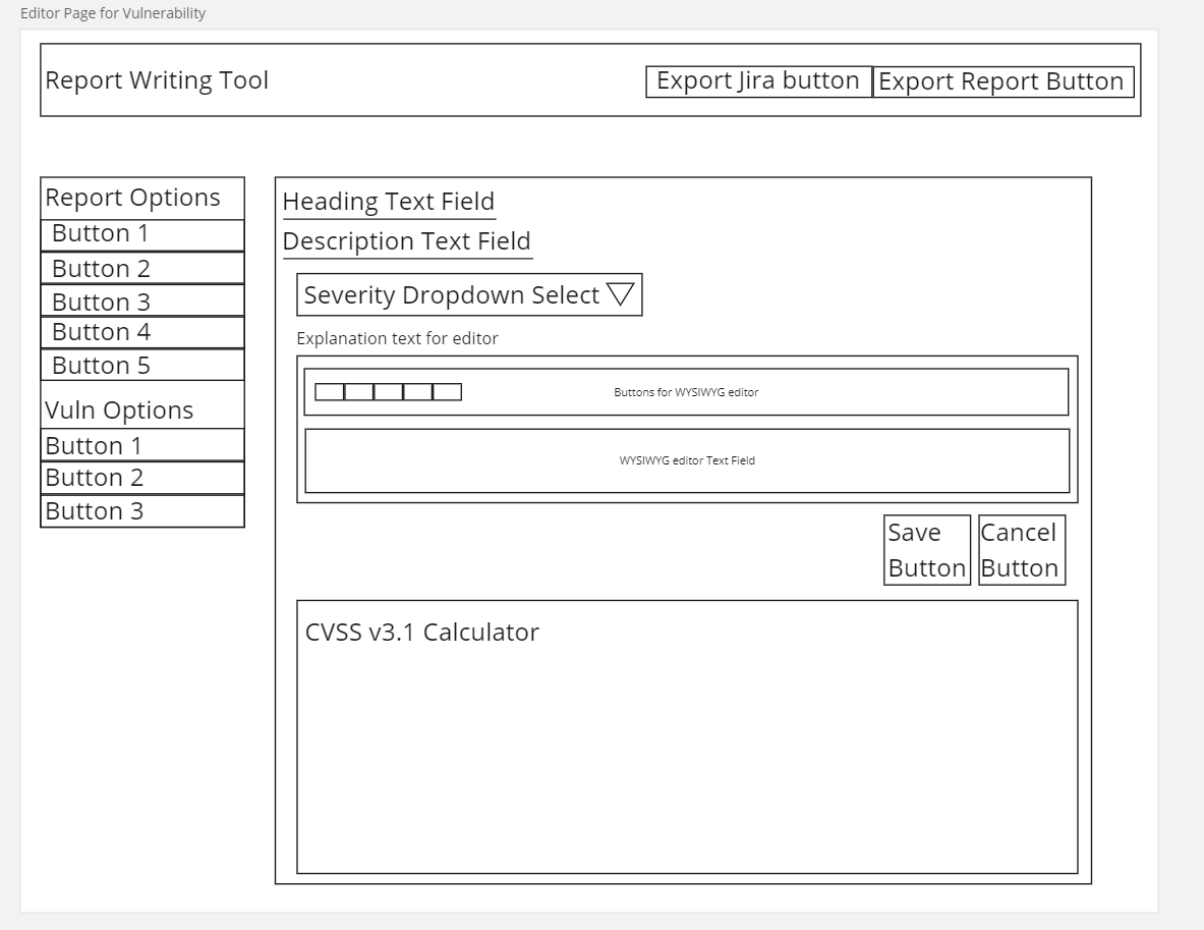


Figure 14 Editor Page for Vulnerabilities

A core library to the entire project was Material User Interface (MUI) (Material UI, 2024). MUI is an open-source library of premade user interface components that can be used and customised for specific uses with other libraries. Since most components used in development are from MUI, the design for the user interface is consistent with colour and styling (Salonen, 2023).

The front-end code for the application was initially developed as one file with a singular large component containing multiple smaller components for the toolbar, sidebar, and main section for displaying the list of headings, notes, or vulnerabilities. This would cause issues when including the editor pages required to write up the sections or vulnerabilities as the page would have to hide and unhide components through a button click which increases the code complexity, making the code harder to read and understand. The code file was originally 400 lines, after refactoring the file became under 100 lines of code. A snippet of the large component can be seen in Figure 14.

```
function App() {

  const [value, setValue] = React.useState('');

  const handleChange = (event, newValue) => {
    setValue(newValue);
  };

  return (
    <React.Fragment>
      <CustToolBar>
      <CustToolBarButtons arrowPlacement='bottom-end' tooltip={ExportVulnToJira}>Export Vulnerabilities To Jira</CustToolBarButtons>
      <CustToolBarButtons arrowPlacement='bottom-start' tooltip={ExportReport}>Export Report (Word) </CustToolBarButtons>
      </CustToolBar>
      <CustTabs onChange={handleChange} value={value}></CustTabs>


<Grid container>


<Box sx={{display: 'inline-flex', flexDirection: 'row', alignContent: 'stretch' , justifyContent: 'flex-start', flexWrap: 'nowrap', width: '100%'}}>

{/* This section of code handles the buttons displayed on the left sidebar */}

  <Grid item>
  <CustButtonGroup Num={0} value={value} group={ButtonsReport}>Report Options</CustButtonGroup>
  <CustButtonGroup Num={1} value={value} group={ButtonsVulnDatabank}>Vulnerability Options</CustButtonGroup>
  <CustButtonGroup Num={2} value={value} group={AuditLog}>Audit Options</CustButtonGroup>
</Grid>



    <Box sx={{bgcolor: '#e6e6e6', border:1, width: '84%', height: '100%', marginLeft: 1}}>

    <CustTabPanel value={value} index={0}>

    <Grid container
        margin={1}
        direction={'column'}
        marginLeft={"40%"}>


   <Grid item>
    <Box
       component='form'
       sx={{  '& > :not(style)': { m: 1, width: '50' },}}
       noValidate
       autoComplete="off"
       alignContent="center">

       <TextField id="Heading" label="Heading" variant='outlined' />

       <Fab color="primary" aria-label="add" marginLeft={"40%"} Type="submit">
       <AddCircleOutlineIcon />
     </Fab>
```

Figure 15 Snippet of large code component

Custom components were produced in separate files to mitigate the issue of a monolithic codebase contained in a singular component. The custom components were organised into folders based on which part of the user interface they were associated with, for example, the sidebar. A sample of the file list with the folders can be seen in Figure 15. Separating the components into custom components improved the readability and simplicity of the code. Utilising this modular programming method meant that new features could be implemented without causing issues to existing code in the tool.

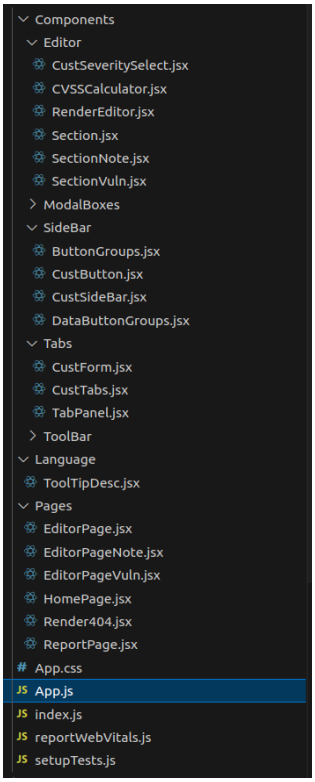


Figure 16 List of custom component files

To produce a layout similar to the wireframe, the components were contained using MUI's Grid component as it allows components to be dynamically resized based on space available on the screen. The Grid component also groups up multiple components, each as an individual Item Grid in a larger container Grid, which can help with positioning the custom components to match the wireframe. Overall, the Grid component supports accessibility for different devices with varying screen sizes. As seen in Figure 17, bright colours were temporarily added to identify the size of each component and how it would position itself using the Grid container component.

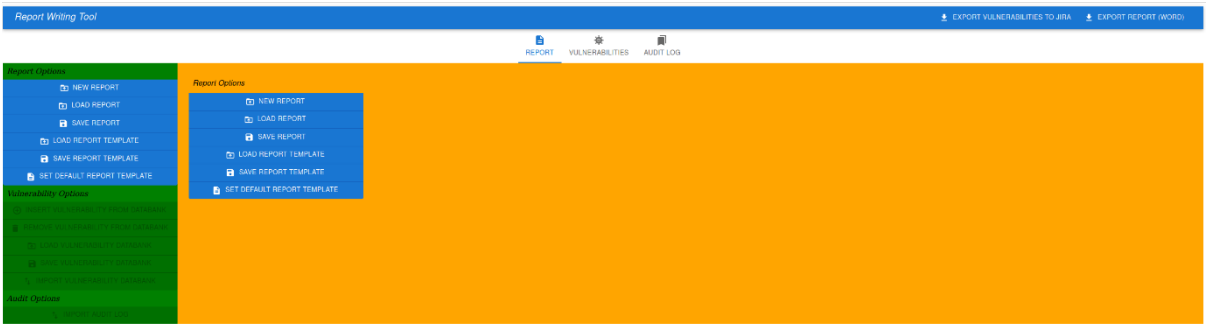


Figure 17 Testing Grid component with Bright Colours

React Router was integrated into the code to provide seamless transitions between the different pages without the browser refreshing to access the new page (Remix, 2024). Using this method meant components could be swapped out to stop code duplication when developing a separate page with identical components. This also ensures that the tool is a single-page application which improves performance since only the information that is required is being loaded rather than the entire page. Additionally, path parameters can be introduced using React Router to fetch data from the back-end

and display it on the front-end. Implementation of React Router being used to access the different components used for each page can be seen in Figure 18.

```
32
33  {/* This section of code handles switching out the components and passing along the id of both Reports and Editors */}
34        <Routes>
35          <Route path="/" element={<CustSideBar value={value}> <HomePage/> </CustSideBar>} />
36          <Route path="Reports/:ReportID" element={<CustSideBar value={value}> <ReportPage value={value}/> </CustSideBar>} />
37          <Route path="/Heading/:SectionID" element={<CustSideBar value={value}><EditorPage /></CustSideBar>} />
38          <Route path="/Vuln/:VulnID"  element={<CustSideBar value={value}><EditorPageVuln /></CustSideBar>}/>
39          <Route path="/Note/:NoteID" element={<CustSideBar value={value}><EditorPageNote /></CustSideBar>} />
40          <Route path="*" element={<Render404 />} />
41        </Routes>
42      </BrowserRouter>
43    </React.Fragment>
44    );}
45  export default App
```

Figure 18 React Router component group

The variable name after the ":" in the path is the path parameter that allows React Router to grab the unique report id passed through the URL using the UseParams() function. An example of this can be seen in Figure 19.

```
const {ReportID} = useParams();

const [refreshCustTabPanel, setRefreshCustTabPanel] = useState(false);

const handleRefreshCustTabPanel = () => {
    setRefreshCustTabPanel(!refreshCustTabPanel); // Toggle the state to trigger a refresh
};

    return(

        <Grid item xs={10}>
        <CustTabPanel
            value={value}
            index={0}
            endpoint={`/api/report/${ReportID}/sections`}
            topLevelEl="sections"
            ComponentPass={CustSection}
```

Figure 19 Example of UseParams() function used to grab ReportID

All the pages contain the same toolbar and sidebar to stop components being reloaded multiple times, which improves the performance of the tool. The toolbar contains the report writing tool name which links back to the landing page, and two buttons. The first button on the toolbar exports the report to word for final review and format checks before exporting to PDF to send to a client. The second button exports the vulnerabilities as individual Jira tickets tagged with their severity rating, which a client can gain access to by either exporting them to their own Jira board or accessing a Jira board created specifically for that client.

The sidebar contains 2 button groups, one for vulnerabilities and one for the report and report templates. The vulnerabilities button group has 3 buttons which are used for adding new vulnerabilities into the databank, removing a vulnerability from the databank, and putting vulnerabilities into the report from the databank. Report options button group has 5 buttons which

allow a user to start a new report based on a template or a blank report, load or remove a report, and save or remove a report template.

All buttons in both the toolbar and sidebar have tooltips appear when they are hovered over to explain what the button does in detail. This ensures a user can use the tool without requiring additional training as each button contains a self-explanatory tooltip. The descriptions for each tooltip were contained inside of a separate file and were imported into the files that required access to improve readability of code.

To provide flexibility and more freedom to users when using the editor page, a What You See Is What You Get (WYSIWYG) Editor was implemented. Draft-JS, which is developed and maintained by Meta, was used to allow control over font formatting and styling (Meta, 2024). The WYSIWYG editor also allows the user to embed links into the sections to provide further information to their client within the report. An advantage of integrating a WYSIWYG editor included ease of use for users as it does not require a user to have a deep knowledge of markup language to add new headers or format a paragraph of text to the exact specifications a user would want. The WYSIWYG editor and it's code can be seen in Figures 20 and 21.

Figure 20 WYSIWYG editor

```
export default function CustRenderEditor({ onDataChange, initialContent }) {
  const [editorState, setEditorState] = useState(() => {
    if (initialContent) {
      try {
        const contentState = convertFromRaw(JSON.parse(initialContent));
        return EditorState.createWithContent(contentState);
      } catch (error) {
        console.error('Error parsing initial content:', error);
      }
    }
    return EditorState.createEmpty();
  });

  useEffect(() => {
    if (initialContent) {
      try {
        const contentState = convertFromRaw(JSON.parse(initialContent));
        setEditorState(EditorState.createWithContent(contentState));
      } catch (error) {
        console.error('Error parsing initial content:', error);
      }
    }
  }, [initialContent]);

  const handleEditorStateChange = (newEditorState) => {
    setEditorState(newEditorState);
    onDataChange(JSON.stringify(convertToRaw(newEditorState.getCurrentContent())));
  };

  return (
    <Grid
      item
      sx={{ bgcolor: 'whitesmoke', border: 1, flexGrow: 1, padding: 2, marginTop: 1, marginBottom: 2 }}
    >
      <Editor
        spellCheck={true}
        placeholder="Please Enter Text Here!"
        editorState={editorState}
        onEditorStateChange={handleEditorStateChange}
        toolbarClassName="toolbarClassName"
        wrapperClassName="wrapperClassName"
        editorClassName="editorClassName"
      />
    </Grid>
  );
}
```

Figure 21 Code for WYSIWYG editor

The text and formatting provided by the WYSIWYG editor is stored as a JSON file which cannot be displayed properly to the user interface. To display the text and formatting, the data had to be converted to raw, as seen in Figure 21, and passed to a read-only version of the WYSIWYG editor used on the sections of the report page. The read-only editor can be seen in Figure 22.

```
return (
    <Grid container
        direction={'row'}
        justifyContent={'space-between'}
        sx={{ bgcolor: 'whitesmoke', border: 1, flexGrow: 1, padding: 2, marginTop: 1, marginBottom: 2 }}>

        <Grid item md={6}>

            {/* Text from adding a new heading will go here */}
            <Typography variant="h5" sx={{ textDecoration: 'underline' }}>{Heading}</Typography>
            <Typography variant="h7">{Description}</Typography>

            {/* Render the content with Draft.js Editor */}
            <Box margin={5}>
                <Editor
                    editorState={editorState}
                    toolbarHidden={true} // Hide toolbar to make it read-only
                    readOnly={true} // Make it read-only
                />
            </Box>

        </Grid>

        <Grid item sx={8} md={0}>
            <IconButton color='error' onClick={handleDeleteSection}><DeleteIcon /> </IconButton>
            <IconButton color='info' onClick={handleEditSection}><EditIcon /></IconButton>
        </Grid>
    </Grid>
)
```

Figure 22 Read-only version of editor used in Section component

The editor page for vulnerabilities included three additional components, a multi-choice dropdown to select severity of the vulnerability and a CVSS version 3.1 calculator imported from cvss-v3.1-react (habilelabs, 2020). The calculator as an additional feature is useful for client debriefings which are held at the end of a penetration test to explain how the severity of the vulnerability is determined. The calculator in its current state only uses the base metrics and can be seen in Figure 23.

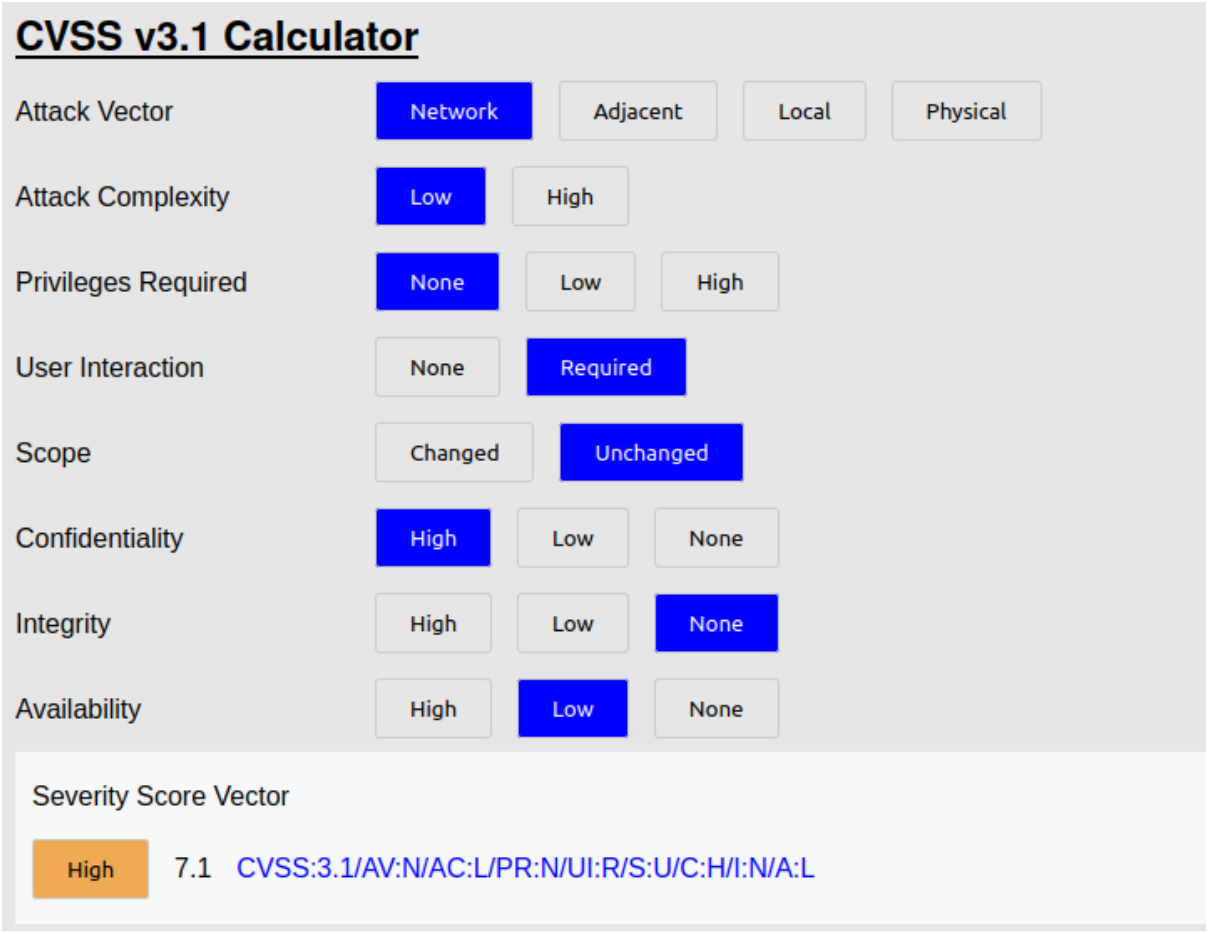


Figure 23 CVSS calculator v3.1

The main section of the report page was split into three separate tabs which were the headings, vulnerabilities, and notes tabs. To create a new entry, the user would input a heading and description into the form at the top of the tabs. On each user entry in the landing page, there was a delete button to remove the entry and an edit button which would take the user to the editor page for that entry. The sections can be seen in figure 24.

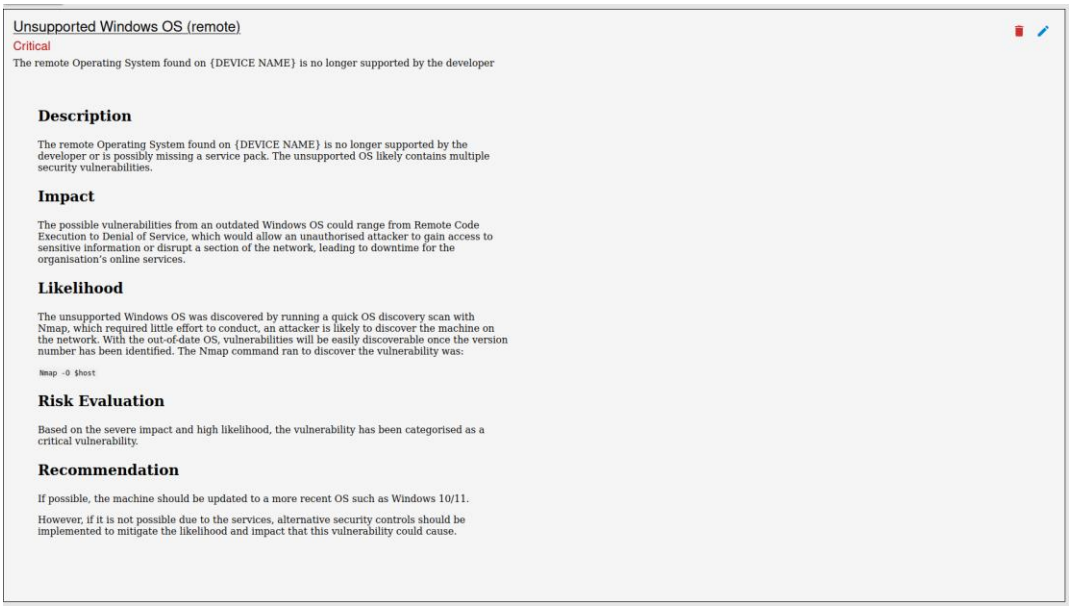


Figure 24 Section for vulnerability entries in report writing tool

The front-end of the tool was required to be attached to the back-end of the tool to continue development, because the event handlers require immediate connection to a storage solution to store the user input. SWR library for data fetching was implemented to retrieve data from the back-end storage and display it onto the page using a custom section or modal component (vercel, 2024). The implementation and output for this component after fetching the data can be seen in Figures 25 and 26.

```
import React, { useState } from 'react';
import { FormControl, FormControlLabel, Radio, RadioGroup, Button } from '@mui/material';
import useSWR from 'swr';
import CircularProgress from '@mui/material/CircularProgress';
const fetcher = (url) => fetch(url).then((res) => res.json());



export default function ReportSelectionJira({handleModalClose}) {
  const [selectedReport, setSelectedReport] = useState();

  const { data, isLoading } = useSWR('/api/report', fetcher)


  const handleReportChange = (event) => {
    setSelectedReport(event.target.value);
  };

  const handleExportReport = () => {
    const body = {}
    body.ReportId = selectedReport
  fetch(
    `/api/report/exportjira`,
    {
      method: "POST",
      headers: {
        "Content-Type": "application/json"
      },
      body: JSON.stringify(body)
    }).then(res => res.json()).then(res =>{
      handleModalClose()
    })
};


  if (isLoading) return <CircularProgress />

  return (
    <div>
      <FormControl component="fieldset">
        <RadioGroup  value={selectedReport} onChange={handleReportChange}>
          {data.reports.map((Report) => (
            <FormControlLabel key={Report.id} value={Report.id} control={<Radio />} label={Report.Report} />
          ))}
        </RadioGroup>
      </FormControl>
      <Button
        onClick={handleExportReport}
```

Figure 25 UseSWR fetching all reports from database

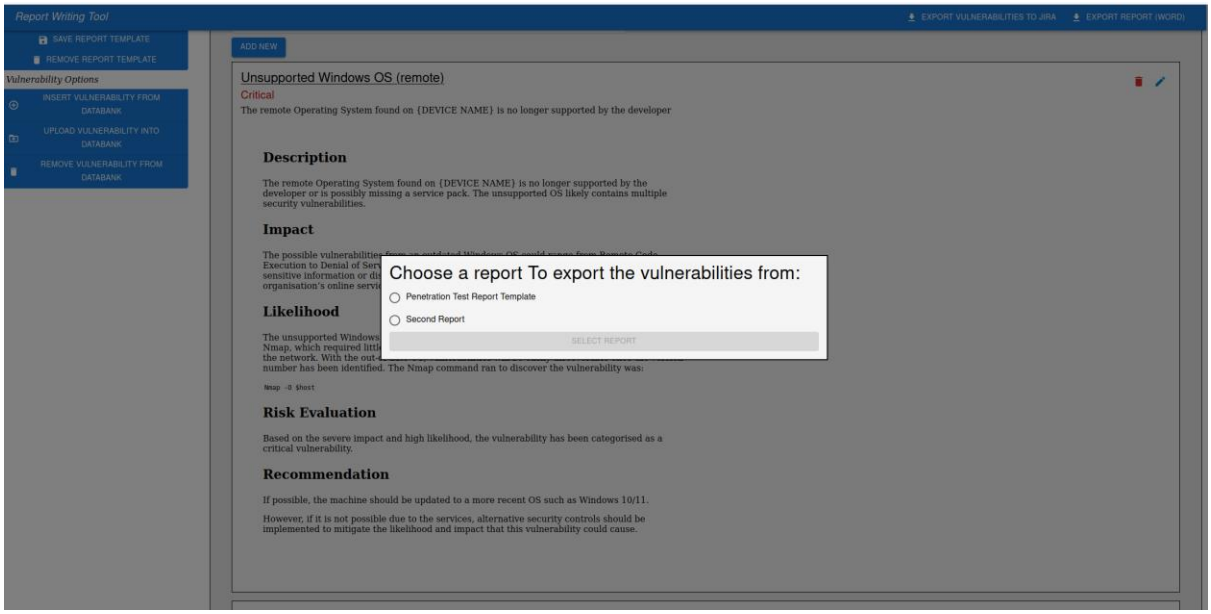


Figure 26 Modal component using UseSWR to fetch all reports in database

The penetration test report template that was developed was integrated as the default report template which would load with the tool upon initial startup.

Figure 27 Snippet of report template used in report writing tool

As seen in Figure 27, the descriptions explaining how to write-up each section was included as placeholder text to ensure that reporter had guidelines to follow while writing. Additionally, eight vulnerabilities were written up as sample data to go into the vulnerabilities section, ranging from critical to low.

3.4.2. Back-End Development

Python (The Python Software Foundation, 2024) in combination with the Flask library (Ronacher, 2024) was the preferred language for developing the report writing tool's back-end. Using the Flask library provides a scalable web development environment which is useful for this type of development as requirements could change in the future or additional features could be added without refactoring all the code within the tool. Flask also uses a development server to run while coding to see bugs happen in real-time, which can aid in narrowing down where the issue lies in the code. Flask also has a vast amount of documentation to help reduce the learning curve when starting development.

MongoDB (MongoDB, 2024), with the Python library PyMongo (MongoDB, 2024) was included in this tool to store all the reports, vulnerabilities, and templates within the back-end of the tool. MongoDB uses NoSQL which is a non-relational system that stores information in a collection as a series of binary JSON-like files named BSON documents. This replaces the alternative MySQL which uses a relational information system of tables with rows and columns. The advantage of using BSON is that the data can vary between each document and does not rely on a schema like MySQL does, which has improved performance when scaling up and increasing the amount of the documents within the database.

To understand what collections were required and the fields required inside of the documents, the tool Moon Modeler was used (Datensen, 2023). Based on the requirements of the front-end, seven collections were designed. The reports collection, the sections collection, the vulnerabilities collection, the notes collection, the sections template collection, the vulnerability templates collection, and the report template collection. Each document in the vulnerability databank has a unique identifier which is used to identify vulnerabilities in a document in the reports collection. The diagram of the three collections and their relations can be seen in Figure 28.

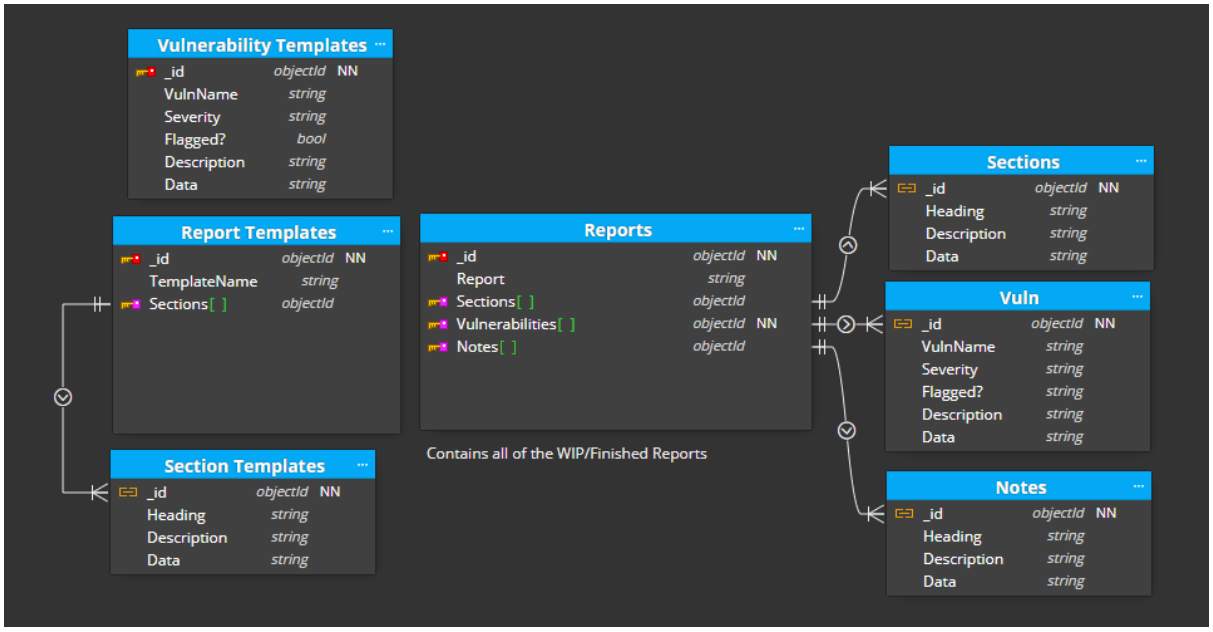


Figure 28 MongoDB Diagram Planning

The diagram of the collections was then imported into the back-end Python file to create the seven collections and the fields required for each document. To connect these to the front-end, the database needs to be accessible through a web connection, which can be done with Flask acting as an API endpoint.

To allow the front-end to call upon the back-end to read and write the information into the database, API calls were developed. The API calls use Flask's blueprints to separate API calls into files categorised by topic such as sections, notes, and vulnerabilities. Each of these blueprints were called into the main Python file named server.py as seen in Figure 29.

```python
from flask import Flask, render_template, request, url_for, redirect
from Blueprints.AllReports import allReportsBlueprint
from Blueprints.currentReport import currentReportBlueprint
from Blueprints.exportJira import exportJiraBlueprint
from Blueprints.exportWord import exportWordBlueprint
from Blueprints.note import noteBlueprint
from Blueprints.section import sectionBlueprint
from Blueprints.template import templateBlueprint
from Blueprints.vuln import vulnBlueprint

from Blueprints.JsonEncoder import CustomJSONProvider

app = Flask(__name__)
app.json = CustomJSONProvider(app)
# ------------------------------------------------------------------------ #
#              Related to grabbing all of a document/collection            #
# ------------------------------------------------------------------------ #
app.register_blueprint(allReportsBlueprint)
app.register_blueprint(exportJiraBlueprint)
app.register_blueprint(exportWordBlueprint)
# ------------------------------------------------------------------------ #
#                          Below is all Report API Calls                   #
# ------------------------------------------------------------------------ #
app.register_blueprint(currentReportBlueprint)
# ------------------------------------------------------------------------ #
#                        Below is all Section API calls                    #
# ------------------------------------------------------------------------ #
app.register_blueprint(sectionBlueprint)
# ------------------------------------------------------------------------ #
#                         Below is all Vuln API calls                      #
# ------------------------------------------------------------------------ #
app.register_blueprint(vulnBlueprint)
# ------------------------------------------------------------------------ #
#                          Below is Notes API Calls                        #
# ------------------------------------------------------------------------ #
app.register_blueprint(noteBlueprint)
# ------------------------------------------------------------------------ #
#                  Below this is all the Template API Calls                #
# ------------------------------------------------------------------------ #
app.register_blueprint(templateBlueprint)


if __name__ == "__main__":
    app.run(debug=True)
```

Figure 29 All Blueprints being called into main Python file

A blueprint dedicated to exporting the data and formatting to a word document was included as a back-end feature to reduce the number of fetch requests between the front-end and the back-end. The function used the libraries, Python-docx (Canny, 2024), Python-docx-template (Lapouyade, 2024), and Beautifulsoup (Richardson, 2024) as they can be used to create, manipulate, and template a Word document while converting html to the correct format for docx documents. The JSON provided from the WYSIWYG editor had to be translated to HTML and then rich text before it could be displayed onto the word document. Additionally, to place the information at the appropriate spots on the word document, Jinja2 delimiters were used to inform the doc.render() function where to place information that was fetched from the report. The code and a part of the report template for exporting can be seen in Figure 30 and 31.

```
            elif vulnerability['Severity'] == 'Low':
                LowVulns.append(vulnerability)

        print(MediumVulns)

        # Prepare context for rendering the template
        context = {'ExecutiveSumHead': execSummarySection.get("Heading", ""),
                   'ExecutiveSumBody': execSummarySection.get("Data", ""),
                   'Sections': sectionsData,
                   'CriticalVulns': CriticalVulns,
                   'HighVulns': HighVulns,
                   'MediumVulns': MediumVulns,
                   'LowVulns': LowVulns,
                   'AppendixHead': appendixSection.get("Heading", ""),
                   'AppendixBody': appendixSection.get("Data", ""),
                   'GlossaryHead': glossarySection.get("Heading", ""),
                   'GlossaryBody': glossarySection.get("Data", "")}

        # Render the template
        doc.render(context)

        # Save the generated Word document
        doc.save(report["Report"] + '.docx')

        return {"message": "Word document generated successfully."}, 200
    else:
        return {"error": "Report not found."}, 404

```

Figure 30 Delimiters for doc.render() function

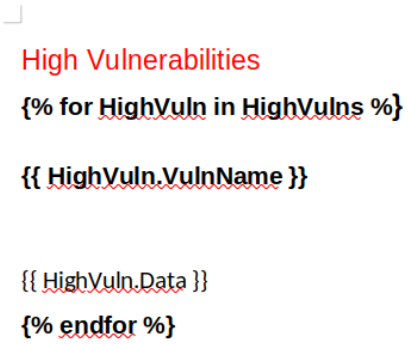


Figure 31 Example of delimiters for importing text

Interacting with Jira to export the vulnerabilities as tickets required the library Python Jira (Speakmon, 2024). The blueprint connected to an example Jira board using an API and project key to test the exporting of tickets. For organisations that wish to use this, they can connect their own Jira board by changing the address, keys, and login for the Jira connection in the Python file. The code and output to the Jira board can be seen in Figures 32 and 33.

```python
# Jira API credentials
JIRA_URL = 'https://report-writing-project.atlassian.net'
JIRA_USERNAME = '2005335@uad.ac.uk'
JIRA_API_TOKEN = 'ATATT3xFfGF0t46feRg1S73dJJz05woPnqHXxSN1Av3bh8uVBwRgXZtNAXuufHFLBMRoQnctHRaaPMIpQxGwyG-AddRuj9
JIRA_PROJECT_KEY = 'REP'


jira_client = JIRA(JIRA_URL, basic_auth=(JIRA_USERNAME, JIRA_API_TOKEN))

# Function to create a Jira issue
def create_jira_issue(summary, description):
    issue_dict = {
        'project': {'key': JIRA_PROJECT_KEY},
        'summary': summary,
        'description': description,
        'issuetype':  {"id": "10001"},
    }

    return jira_client.create_issue(fields=issue_dict)

#Grabs all Vulnerabilities for the report to export as Jira Tickets
@exportJiraBlueprint.route("/api/report/exportjira", methods=["POST"])
def ExportJira():
    data = request.json
    reportId = data.get("ReportId")


    # Initialize HTML exporter
    html_exporter = HTML()

    # Retrieve report data from MongoDB
    report = reports.find_one({"_id": ObjectId(reportId)})

    if report:
        # Extract vulns IDs from the report document
        vulnIds = report.get("Vulnerabilities", [])

        # Fetch vuln documents from the vulnerabilities collection using the vulns IDs
        vulnsData = []
        for vulnId in vulnIds:
            vuln = vulns.find_one({"_id": vulnId})
            if vuln:
                vulnsData.append(vuln)
```

Figure 32 Code to export vulnerabilities as Jira tickets

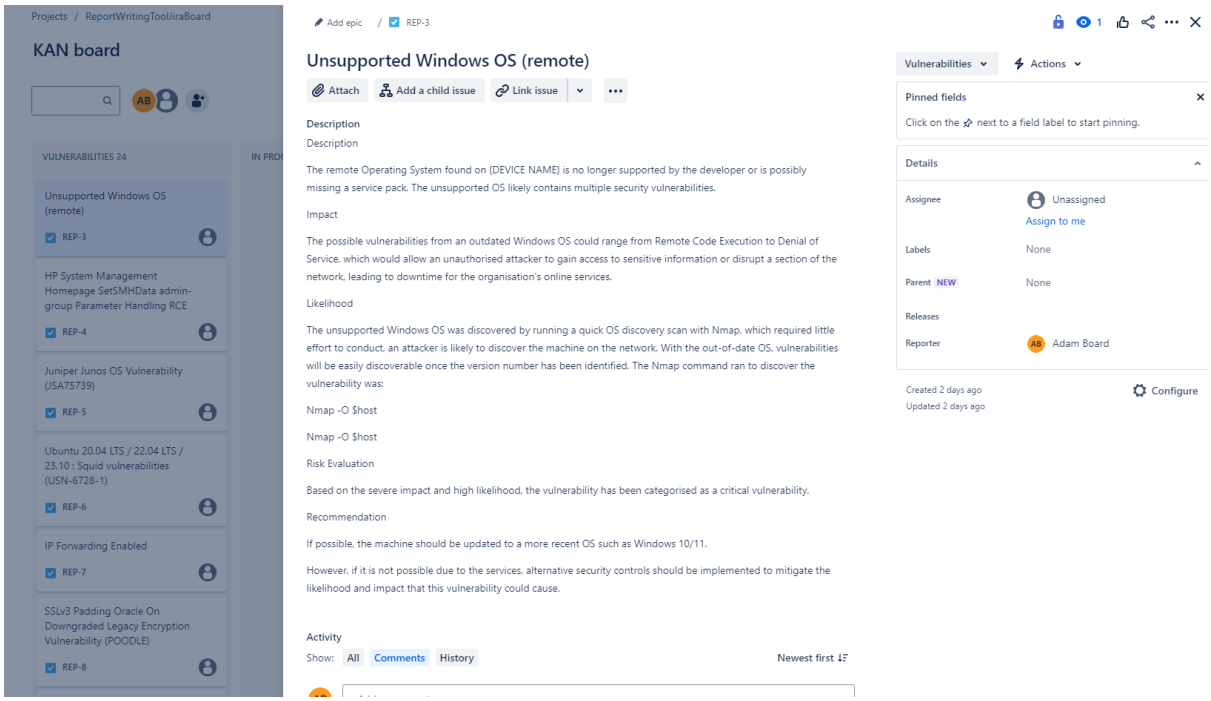


Figure 33 Ticket output to Jira board

To test each API call before integrating the functionality with the front-end, a web request tool named Bruno was utilised (AngelaSYuan, 2024). As seen in Figure 34, each API had to be tested with multiple requests to ensure that any bugs were identified and could be mitigated before integrating into the front-end. JSON was sent in the body of the requests to match the data format that MongoDB accepted, which was Binary JSON (BSON).

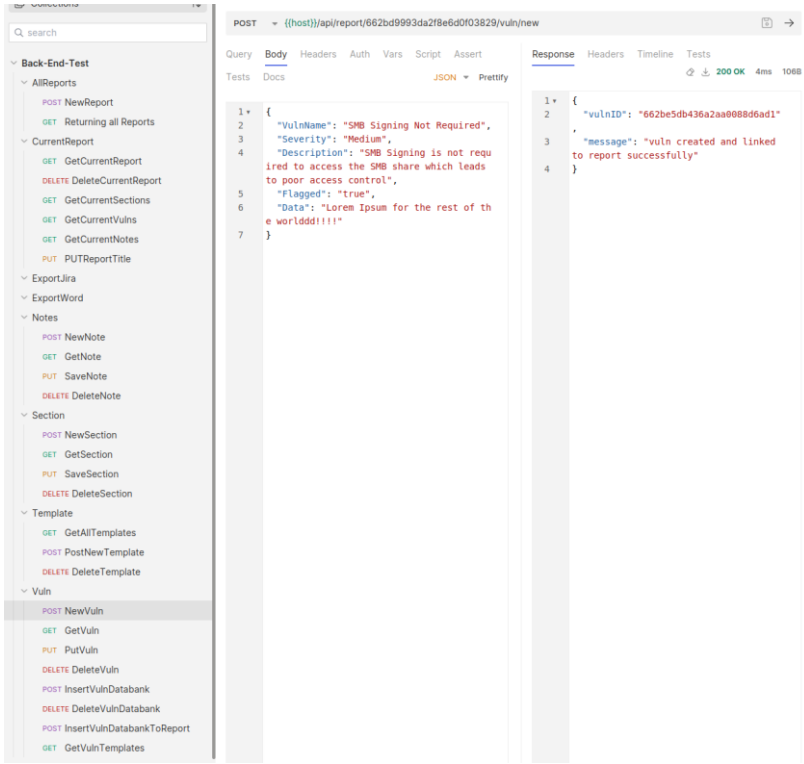


Figure 34 List and Example of testing API calls

4. Results

The results section presents the findings from analysing the different penetration testing reports and includes examples of how this shaped the final report template. The completed report writing tool with input and output examples has also been presented in this section.

4.1. Penetration Test Report Template

From reviewing the three research paper's report templates and identifying deliverables from the ISO 29147 (ISO, 2018) guidelines during the literature review, the following report template was produced as seen in Figure 35.

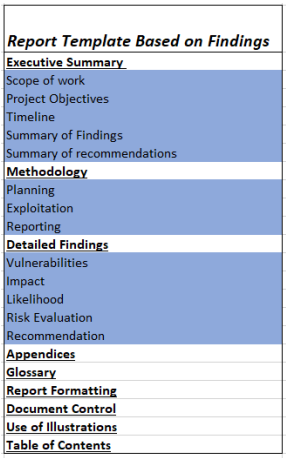


Figure 35 Results of report template research

As seen in Figure 35, there is three main sections each with subsections, as well as a list of sections that are additional to the report which is found below the recommendation subsection. These were determined the sections that are necessary in an effective penetration test report template.

As shown in Figure 36, the results of the analysis conclude that TrailOfBits's report had the highest rating of 6.94, and SecureLayer7 had the lowest rating of 1.26.

Penetration Test Report Analysis					
Organisation	**Overall Rating */10**	**Top-Level Overview**	**Methodology**	**Detail Findings**	**Additional Importance**
GlitchSecure-Example-Pentest-Report	5.03	5.50	5.00	5.20	4.43
GlitchSecure-Example-Vulnerability-Assessment-Report	4.52	5.50	5.00	5.00	2.57
Bishop Fox Assessment Report	4.10	5.67	1.00	4.60	5.14
chess-cybersecurity-penetration-testing-sample-report	5.05	6.50	6.67	3.60	3.43
EXAMPLE-Penetration Testing Report v.1.0	3.90	4.83	1.67	3.40	5.71
TrailOfBits - Kubernetes Final Report	6.94	6.50	8.00	6.40	6.86
UnderDefense - Anonymised-BlackBox-Penetration-Testing-Report	5.38	7.50	5.00	5.00	4.00
NCC_Group_-_phpMyAdmin	4.11	5.17	0.00	6.40	4.86
NCC_Group_-_Ricochet	4.05	4.67	0.00	6.40	5.14
iSEC_Wikimedia	5.79	6.67	3.67	6.40	6.43
NCC Group Zcash Crypto Report 2016	5.35	6.00	3.67	6.60	5.14
Atredis+Partners Mullvad+VPN+Platform+Security Assessment Report	5.33	6.33	7.00	3.00	5.00
BHP_Mobile_Wallet_Penetration_Test_Report	6.10	6.50	9.00	4.60	4.29
RedSiege-SampleReport	3.14	3.33	0.00	3.80	5.43
OffSec penetration-testing-sample-report	3.68	3.67	2.00	5.20	3.86
VoidSec-Ghost	3.05	2.00	0.00	6.20	4.00
SecureLayer7-Pentest-report	1.26	2.17	0.00	2.00	0.86
Report URI - 2020 Penetration Test Report	6.63	8.17	3.00	7.20	8.14
Cure53 - pentest-report bitwarden	2.00	3.00	0.67	2.60	1.71
Cynergi-Solutions - eclipse-bank-security-assessment-report-example	6.27	5.833333333	6.00	7.40	5.86
Description On Each Section	Overall Rating of the Report as a whole	Every section that is related to the top-level overview of the report which would be given to an executive/non-technical user	This section provides the needed information about how the penetration testing was conducted. What steps have been followed to collect the information, analyze them, the risk rating methodology used	This section provides detailed information for each finding. Present the findings in the simplest way as possible. There are a number of ways in which results can be presented. Here are a few: • Tables • Graphs • Pie or Bar charts • Diagrams	Anything that is important to a good report but is not a finding in itself for vulnerabilities. Instead, the additional importance makes it nicer for a user to read.

Figure 36 Results of report analysis

The notes included in the analysis for TrailOfBits stated that the key takeaway for this report is that the executive summary was far too long, taking three full pages of the report; The appendices provided information that was additional but not necessary to understand the full report, which is the purpose of the appendices; The Descriptions for the methodology section were in-depth and would allow another penetration tester to replicate the testing procedure; The detailed findings did not include any form of graphs but did include screenshots to help explain a vulnerability. TrailOfBits report scored above five in each criterion.

The notes included in the analysis for SecureLayer7 explained that this is a perfect example of a poor report. The headings were included as links rather than a properly formatted table of contents; no appendices or references to provide further information; the vulnerability mitigations mentioned to look at a different vulnerability mitigation in the same report to get the answer; One of the vulnerability headings were missing the severity rating which meant detail was missing in the report and did not disclose complete information; Finally, the writing style of the report was including first-person writing with the use of "I". It scored a zero on methodology as it did not include one and overall had scores that never exceeded above three. Overall, the reports that were analysed did not reach higher than seven out of ten. The final report template that includes the descriptions and sections can be found in Appendix B.

4.2. Performance Review Report

The quality of a report directly correlates with the skillset of a penetration tester as shown in numerous research papers (Bertoglio, et al., 2022) (Kousik Barik, 2021). The performance review report, found in Appendix C, provides the information and tools required to improve a penetration tester's writing capabilities. The descriptions provided for each section to teach the reviewer include the suggested approach for giving effective feedback (Baker, 2010). An example of this can be seen in Figure 37.

Strengths of the report

The reporter writing this section should identify the strengths of the report and highlight them with a small list of bullet points to describe what made that section a success (Example Below). The reporter should also provide a summary of the strengths at the top of this section in either bullet points or a brief description to allow the employee to understand everything they performed well on. Additionally, if this is not the first report, the reporter should highlight the improvements made by the employee based on the "Areas of Improvement" section with a description on how they improved in that area. Finally, ensure the feedback is written in a clear and concise manner that incorporates the five main components: Relevance, Accuracy, Timely, Specific, and Easy to Understand. REMOVE THIS TEXT ONCE REPORT IS COMPLETE.

Remote Code Execution (RCE) vulnerabilities allow an unauthorised attacker to potentially steal sensitive information or cause downtime for the organisation's critical services. Which would damage the reputation of the organization.

- The explanation on Remote Code Execution is concise and describes the potential damage that could affect the client.
- The language used is not fear mongering to an extreme level and gives a realistic example of what could happen should an RCE of high or critical level be exploited.
- The language is simple and not too technical, which makes it easier for a client to read without having to decipher the information.
- No spelling mistakes or grammatical errors.

Figure 37 Description of how to provide feedback with example

4.3. Standardised Report Writing Tool

The report writing tool presents all the necessary features to produce a complete penetration test report. The built-in editor grants the ability to format and stylise the user's writing. The placeholder headings, vulnerabilities, and text that is set for the built-in report template provides the user guidelines and examples of how to write an effective report. The user can also determine the CVSS score by using the calculator which is included inside of the vulnerability editor page. Finally, the information written inside of the report writing tool can be exported as a complete report, or the vulnerabilities can be exported as Jira tickets separately. The example sections and vulnerability can be seen in Figures 38 and 39. A snippet of an example exported report can be seen in Figure 40, and the example Jira tickets can be seen in Figure 41.

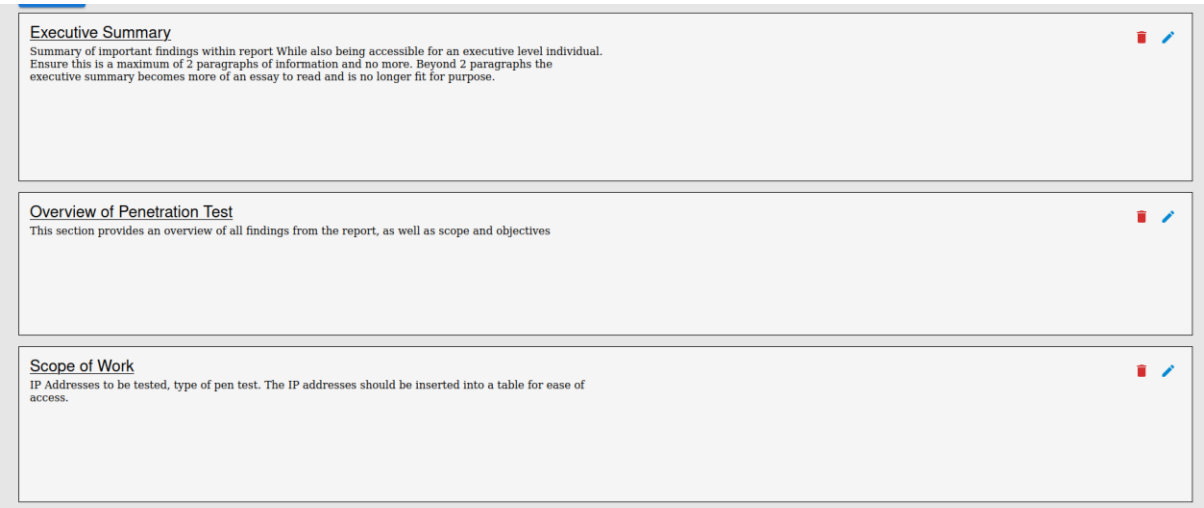


Figure 38 Example Sections From report template

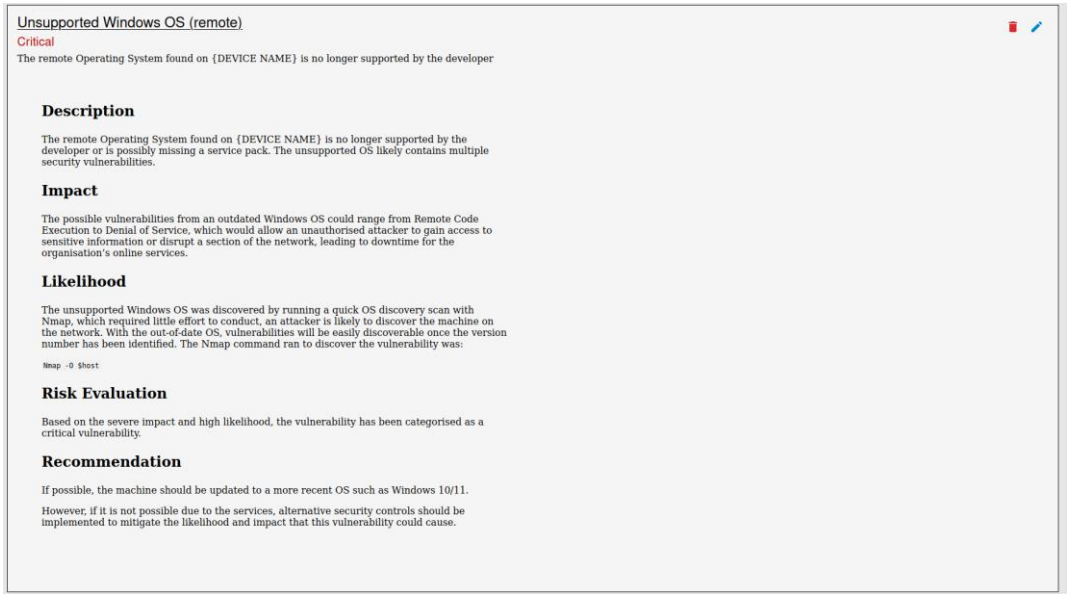


Figure 39 Example vulnerability from list of vulnerabilities

Detailed findings

Critical Vulnerabilities

Unsupported Windows OS (remote)

Description

The remote Operating System found on {DEVICE NAME} is no longer supported by the developer or is possibly missing a service pack. The unsupported OS likely contains multiple security vulnerabilities.

Impact

The possible vulnerabilities from an outdated Windows OS could range from Remote Code Execution to Denial of Service, which would allow an unauthorised attacker to gain access to sensitive information or disrupt a section of the network, leading to downtime for the organisation's online services.

Likelihood

The unsupported Windows OS was discovered by running a quick OS discovery scan with Nmap, which required little effort to conduct, an attacker is likely to discover the machine on the network. With the out-of-date OS, vulnerabilities will be easily discoverable once the version number has been identified.

Figure 40 Example report output using Python-docx-template

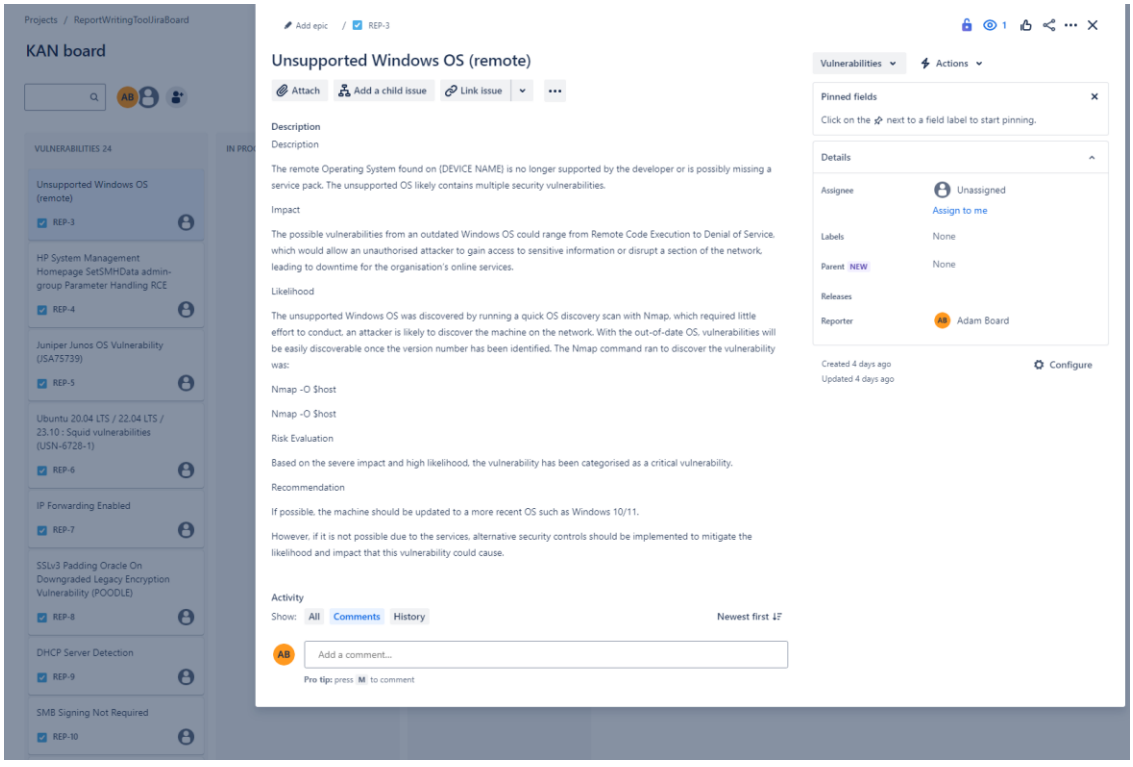


Figure 41 Exported vulnerabilities as Jira tickets

The full report with the exported information can be seen in Appendix E. The report writing tool is entirely open source which grants organisations the ability to alter the application to meet their requirements. Additionally, the vulnerability databank and custom templating reduces duplicated effort writing common vulnerabilities, or report sections, multiple times.

5. Discussion

The aim of this project was to implement a standardised report writing tool and evaluate its effectiveness based on previous literature and alternative report writing tools. Three research questions were proposed as stepping stones to accomplish the aim and reach a conclusion on whether the report writing tool was effective.

5.1. Increasing the Consistency and Efficiency of Report Writing

The first research question sought to determine how current tools increase the consistency and efficiency of penetration test report writing. As mentioned in the literature review, PlexTrac allows organisations to import their own report template into the tool and break down the template's headings into individual sections that can be moved around and edited freely. This can stop information overload for penetration testers when they open the report for the first time because they can focus on an individual section rather than the whole report. Additionally, PlexTrac utilises a vulnerability databank to reduce duplicated work being done for multiple reports (PlexTrac, 2022).

Based on the features available in PlexTrac, the developed report writing tool has a standard report template that is integrated into the tool, with the option to introduce custom templates. The tool also contains an integrated vulnerability databank to allow users to save vulnerability descriptions after they have been written in the tool. An additional feature of this tool is the capability to include descriptions explaining how to write each section in the template. This ensures that the user writing the report has guidelines to follow throughout the reporting stage. The ability to produce custom templates provides flexibility for larger organisations that already have a template with guidelines but not a way to apply it due to only using Microsoft Word as their sole report writing application. The report writing template included in the tool acts as a guideline rather than a hard standard as it provides a method of staying consistent while also allowing flexibility, which organisations have stated is incredibly important to them, as using a hard standard would reduce their ability to make the report their unique selling point (Knowles, et al., 2016). Having a unique selling point is especially important to stay relevant in a competitive and evolving market, as it makes an organisation stand out to clients when compared to their competitors.

Similarly to PlexTrac, the report writing tool's integrated vulnerability databank improves the efficiency of report writing for a penetration test. This is because common vulnerabilities that are found across multiple penetration test reports can be saved into the vulnerability databank with a detailed description to be reused again in a different report. This reduces the time wasted duplicating the same vulnerability description across multiple reports manually.

The mentioned features of the report writing tool help reduce the tediousness, complexity, and time-consumption that penetration testers have found to be feeling during the report writing stage (Alghamdi, 2021) (Sharma, et al., 2023). PlexTrac and another framework tool named Dradis, allow penetration testers to combine their findings in a centralised repository which each penetration tester has access to and allowed them to easily share information (Young, 2020) (PlexTrac, 2022). This can help keep consistency when a report is being written by multiple penetration testers. The report writing tool is split into a back-end and front-end for this exact purpose. The back-end acts as an API endpoint to interact with the database while multiple users can download and use the front-end open to work on the same report. Additionally, the tool has a dedicated section for adding notes internally which are related to the penetration test but does not get exported when the written sections and

vulnerabilities are being exported onto a word document. The notes section can be used to inform other penetration testers of what is left to be completed or what should be kept in mind when writing up the penetration test report. Along with the notes section, tooltips are provided on each button of the tool, detailing the exact functionality to reduce the time taken learning how to use the tool to allow penetration testers to easily adapt to using the report writing tool.

Unlike PlexTrac, Dradis is open source which allows organisations to personalise the features and functionality of the framework to meet their requirements. However, it is built using the web framework Ruby on Rails which is only used by 5.49% of developers worldwide. Meanwhile, the report writing tool in this project is also open source to provide the same level of customisability as Dradis, and instead uses Javascript React as its framework, which is used by 40.58% of developers globally (Vailshery, 2024). A major advantage is that it is more likely that an organisation would have a React developer who could alter the tool to fit their needs and improve the tool to boost the efficiency of report writing with new features compared to Dradis as it uses a less popular web framework.

5.2. Establishing the Standards and Components Used for Penetration Test Reports

Another question aimed to establish what current standards and components should be taken into consideration to produce consistent and effective penetration test reports. The objective of conducting an extensive review into penetration test reports aided in identifying the standards and components that are used in a penetration test. The analysis of numerous penetration test reports from different organisations indicated that there was a lack of standards being taken into consideration, especially for design, formatting, detailed vulnerabilities findings, and the executive summary. However, while the overall vulnerability findings were lacking, a consistent factor in all the reports analysed was the use of the CVSS scoring system for classification of vulnerability severity. This aligns with the findings from the literature review as clients often mandated that an organisation providing a penetration test report must use the CVSS scoring system to ensure they can compare with previous reports that had been done for them (Knowles, et al., 2016). The report writing tool uses the CVSS scoring system due to its common usage across cyber security as shown from the analysis of penetration test reports, which can be found in Appendix A, and literature review. Additionally, the CVSS scoring system is simple to use and understand due to its qualitative severity ratings which are typically colour coded based on its severity. Although, as described in the literature review, the CVSS scoring system could potentially cause wasted time remediating vulnerabilities as organisations typically use the base score and ignore the environmental and temporal scores. (Kumar, 2014). The environmental group score factors in collateral damage potential and percentage of systems affected by the vulnerability. The temporal group score factors in the current exploitability of the vulnerability, the remediations available for the vulnerability, and how credible the report details are regarding the vulnerability (Mell, et al., 2006). Ignoring these group scores reduces the accuracy of the overall vulnerability rating.

A solution to improve the detailed findings section of the reports from the report analysis would be to introduce the ISO 29147 standard for disclosure of potential vulnerabilities in products and online services (ISO, 2018). The penetration test report template designed for this project takes characteristics of the standard into account, such as including the reporter's contact information, including sample code to discover or demonstrate the vulnerability, risk assessment details of identified threats, and a technical description of what actions were performed with the results of those actions included in the report.

Design and formatting across twelve of the twenty reports analysed scored less than 6 which shows that organisations do not realise how much the format and design impact the effectiveness of conveying vital information such as vulnerabilities with their mitigations. The report writing tool follows the guidelines for design and formatting discussed in the literature review, such as a simple colour palette with a dyslexic friendly font (Murchie & Diomede, 2020). The dyslexic friendly fonts utilised were Calibri and Arial, which both belong to the font family Sans-Serif. They were chosen as recommended fonts from the British Dyslexia Association (British Dyslexia Association, 2023). Additionally, the penetration test report template informs the reporter to use illustrations and graphics. 74% of respondents to a study on business insight agreed that data visualisation improved their understanding of the information being conveyed to them (Kyrölä, 2022).

The executive summary is the first component of a penetration test report that a non-technical executive will read to obtain a brief overview of their network's security. Research papers discussing what makes an effective report state that an executive summary should be no longer than 2 paragraphs and summarises the most critical information using language that is focused towards being accessible to non-technical executives (Alharbi, 2010). However, from the report analysis, thirteen of the twenty reports scored 5 or less on the executive summary rating. The common reason for the low scores was identified to be the executive summary not effectively conveying the information.

The final component that largely affects the consistency and effectiveness of a penetration test report is the penetration testers writing the reports. Numerous research papers found that the skillset of a penetration tester directly correlates to the quality of a penetration test report (Bertoglio, et al., 2022) (Kousik Barik, 2021). To resolve the issue involving the human-aspect of report writing, the report writing tool is bundled with a performance review report which is written by a line-manager after each penetration test report is submitted that highlights the strengths and weaknesses of the report with suggestions and resources on how to improve their writing. As mentioned previously, the report writing tool also has guidelines in the attached report template that can aid the penetration tester in writing the penetration test report.

5.3. Implementing a Report Writing Tool to Improve Effectiveness in Communicating with Clients

The third question aims to analyse how implementing a standardised penetration test report writing tool can ensure consistency and increase the effectiveness in communicating findings to clients. A previous study in the pharmaceutical field proved that implementing a tool for aiding in report writing led to more complete reports (Barnes, et al., 2015). Another research paper about developing a red-teaming service-learning course showcased the effectiveness of Dradis, a penetration testing report writing framework, at ensuring that reports were being completed in the time frame available while maintaining a high standard of quality while writing the report (Young, 2020). Dradis was able to maintain the same quality while improving efficiency by importing the Penetration Testing Execution Standard package guidelines for a penetration tester to follow while writing up the penetration test report. Similarly, the report writing tool can contain descriptions on how to write various sections of a report and can be saved as a part of a template, which can be used for multiple reports to ensure they maintain a consistency in quality.

PlexTrac allows a user to export the vulnerabilities as Jira tickets which can be reviewed on an organisation's Jira board. This provides an efficient method of informing the IT or cyber security team of security concerns which can be assigned to specific members quickly through a Jira board. The

report writing tool can export all the vulnerabilities of a report to the Jira board that is set in the back-end code. Based on the literature review, the exported tickets are categorised by their severity to ensure that the critical and high vulnerabilities are communicated as an urgent priority to a client (McGready, 2023).

PlexTrac also has a basic variation of a CVSS calculator built into the tool to be used to determine the severity of a vulnerability (PlexTrac, 2022). To ensure the report writing tool had the same capabilities, a CVSS calculator was included in the editor page for vulnerabilities. The button options to determine the severity each had a description explaining how that option affects the overall vulnerability severity. However, both the report writing tool and PlexTrac only used the base score calculation for the CVSS score, which does not factor in environmental and temporal score. Therefore, the severity rating of the vulnerabilities may have some inaccuracies as it does not consider all factors that could potentially affect the severity, as mentioned previously in Research question 2.

Finally, PlexTrac provides the option to import Nessus descriptions of vulnerabilities directly into the report to save time writing up the vulnerability descriptions. However, as stated in the literature review, automated report generation using tools such as Nessus can be difficult for a non-technical user to read and understand (Salim, et al., 2022). This is because the automated reports provide an overload of information without structuring it out in a format that is easy for a non-technical user to access, which could potentially lead to vulnerabilities being left unmitigated as the information is not accessible to top-level executives. The report writing tool does not include a method of importing descriptions from tools such as Nessus, and instead provides a report template with guidelines that can be followed to ensure the important information is conveyed as a main priority.

6. Conclusion & Future Work

This project was undertaken to design and implement a standardised penetration test report writing tool and evaluate its effectiveness. The background for this project highlighted the importance of providing an effective penetration test report for a client to ensure critical vulnerabilities are not left unmitigated (Mohd Nizam Zakaria, 2019). To produce effective reports, organisations began to implement standardised report templates and report writing tools. However, no project had been conducted to evaluate the effectiveness of implementing a standardised penetration test report template.

In cyber security, automation tools have been used previously to improve the efficiency of each stage of a penetration test, including the report writing stage. Tools such as Nessus are used to automatically generate reports with descriptions explaining each vulnerability found but were typically ineffective at this task due to the overload of information that came from automated report generation (Salim, et al., 2022). To improve report writing, standards such as ISO 29147 (ISO, 2018) and CHECK (National Cyber Security Centre, 2023) were introduced to provide guidance on the information that should be included when disclosing vulnerabilities. Additionally, the Common Vulnerability Scoring System was used to provide qualitative severity ratings for vulnerabilities based on the base, environmental, and temporal group scores (Mell, et al., 2006). However, even with the guidelines introduced, penetration test reports could not satisfy both top level-executives and security personnel effectively (Mohd Nizam Zakaria, 2019).

To understand the requirements to produce an effective report template, research into penetration test report templates was considered (Alharbi, 2010) (Kousik Barik, 2021) (Mohd Nizam Zakaria, 2019). The research determined that an effective report template contained an executive summary, which provides a summary of the key findings in two short paragraphs and uses language that is accessible for executives, as they are the ones acting upon any recommendations and mitigations for the discovered vulnerabilities. The next section within a report template would be the objectives and scope that aid the client in understanding what type of penetration test was conducted and what devices were included as potential targets for the scope. Afterwards, the detailed finding section describes the vulnerabilities regarding their impact against the organisation should it be exploited, the likelihood of the vulnerability being exploited, and the recommendations for mitigating the vulnerability. An important factor of producing an effective penetration test report template is using the appropriate format, colour, and styling (Murchie & Diomede, 2020). Using hard-to-read fonts and clashing colours can cause a report to be less accessible to a target audience. Using recommended dyslexic-friendly font families such as Sans-Serif (British Dyslexia Association, 2023), with simple colours can improve the accessibility of the information contained in the report. Additionally, including labelled graphics and illustrations can emphasise important points that the report is attempting to convey to the target audience.

However, the largest factor in producing an effective penetration test report is the penetration tester writing the report. This project highlighted that the quality of a penetration test report is directly correlated to the skillset of a penetration tester (Bertoglio, et al., 2022) (Kousik Barik, 2021). Therefore, improving the skills of penetration testers will directly affect the quality of the penetration test report, which can be achieved by providing feedback to ensure penetration testers are improving after each report they complete. To provide feedback that does not demotivate penetration testers, the feedback for improving should consider these five factors: Relevance, Accuracy, Timeliness, Specificity, and Ease

of Understanding (Baker, 2010). If penetration testers can meet, or improve, on these five factors, they will make significant improvements to the quality of the penetration test reports.

While improving the penetration testers skillset will increase the quality of the penetration test reports, there is additional methods that can be incorporated to improve the quality of penetration test reports further. An analysis of the pharmaceutical sector has showcased that implementing a report writing aid can improve the completeness of information within a report (Barnes, et al., 2015). Similarly, in cyber security report writing tools such as Dradis and Plextrac, has shown to increase the effectiveness of penetration test reports and improve the efficiency for writing the reports. Dradis does this by acting as a central repository that penetration testers can share their findings and notes from the entire penetration test (Young, 2020). PlexTrac allows users to create and save custom report templates and custom vulnerability descriptions, which removes duplicate effort in writing. However, both tools suffer from various issues (PlexTrac, 2022). Dradis uses Nessus' automatic report generation which as mentioned previously, is prone to causing information overload for the target audience. PlexTrac is closed source which limits the tool potential to meet an organisation's requirements. To provide a solution to the issues currently present in existing penetration test report writing tools, the project aimed to design and implement a report writing tool and evaluate its effectiveness against existing solutions and research.

From the results of the project, the aim was met successfully, because the methodology showcases an open-source report writing tool. The tool acts as a centralised repository for compiling findings by using a separate front-end and back-end to allow multiple front-end users to connect to a singular back-end server. The front-end includes a WYSIWYG editor for editing the sections, vulnerabilities, and notes, which enables the user to choose their own formatting and style to aid in conveying information effectively. Users can also create and save custom report templates and vulnerabilities to be reused for multiple projects to reduce duplication of effort. Additionally, all reports can be exported as Docx documents, or the vulnerabilities can be exported to a Jira board as tickets to be resolved. The report writing tool also includes a standardised report template built into the tool, which was designed and produced based on research papers discussing effective report templates and an analysis of various penetration test reports from different organisations. The template built into the tool contains guidelines for how to write each section of the report, with reminders to include graphics and keep the language targeted towards the target audience. To support users adapting to the report writing tool, each button has a tooltip attached that explains the functionality of the button to reduce confusion. Finally, to ensure that penetration testers are improving after each report they produce, a performance review report was bundled along with the report writing tool. The review report includes descriptions to explain how positive and negative feedback should be given based on research into employee feedback technologies (Baker, 2010).

The report writing tool combines the features from other report writing tools and builds upon them to produce an effective tool for improving the overall quality of penetration test reports. However, there are limitations to the report writing tool which would require more resources and a better knowledge of the programming languages used to produce. The report writing tool does not factor in environmental or temporal scores with the CVSS calculator that is built into the vulnerability editor page. The library (habilelabs, 2020) used for the CVSS calculator could be replaced with a custom-built calculator component that includes base, environmental, and temporal group scores to provide vulnerability severities with improved accuracy. Additionally, if reports become too large, a user would be required to scroll to find a specific section, which can waste time for a user. To combat this, adding a search function would allow users to locate the section they wish to edit immediately and improves

the user experience. Finally, due to the lack of knowledge in secure programming for React and Flask, the tool has not been designed with secure coding practice in mind, which means the report writing tool could potentially contain vulnerabilities. With further knowledge and resources, the code should be refactored to include security measures to mitigate potential vulnerabilities. If they are not identified and mitigated, the vulnerabilities become a risk for organisations using the report writing tool.

7. References

Alghamdi, A. A., 2021. *Effective Penetration Testing Report Writing,* s.l.: IEEE.

Alharbi, M., 2010. *Writing a Penetration Testing Report,* s.l.: SANS Institute.

AngelaSYuan, h., 2024. *Re-Inventing the API Client.* [Online] Available at: https://www.usebruno.com [Accessed 20 April 2024].

Baker, N., 2010. *Employee feedback technologies in the human performance system,* terre Haute: m, Human Resource Development International.

Barnes, C. et al., 2015. *Impact of an online writing aid tool for,* s.l.: BMC Medicine.

Bertoglio, D. D., Schüler, L. G. B., Zorzo, A. F. & Lunardi, R. C., 2022. *Understanding the Penetration Test Workflow: a security test with Tramonto in an e-Government application,* Wuhan: IEEE.

Bettenburg, N. et al., 2010. What makes a good bug report?. *IEEE Transactions on Software Engineering ,* 36(5), pp. 618-643.

British Dyslexia Association, 2023. *Creating a dyslexia friendly workplace.* [Online] Available at: https://www.bdadyslexia.org.uk/advice/employers/creating-a-dyslexia-friendly-workplace/dyslexia-friendly-style-guide#:~:text=Use%20sans%20serif%20fonts%2C%20such,may%20request%20a%20larger%20font. [Accessed 15 February 2024].

Canny, S., 2024. *Python Docx Documentation.* [Online] Available at: https://Python-docx.readthedocs.io/en/latest/ [Accessed 25 March 2024].

Datensen, 2023. *Moon Modeler.* [Online] Available at: https://www.datensen.com/data-modeling/moon-modeler-for-databases.html [Accessed 7 March 2024].

Farah Abu-Dabaseh, E. A., 2018. *Automated penetration testing: An overview,* Copenhagen: The 4th International Conference on Natural Language Computing.

GitHub, 2024. *GitHub Projects.* [Online] Available at: https://github.com/projects [Accessed 17 February 2024].

habilelabs, 2020. *cvss-v3.1-react.* [Online] Available at: https://github.com/habilelabs/cvss-v3.1-react [Accessed 27 February 2024].

Harrell, C. R., Patton, M., Chen, H. & Samtani, S., 2018. *Vulnerability Assessment, Remediation, and Automated Reporting: Case Studies of Higher Education Institutions,* Miami: IEEE.

ISO, 2018. *ISO/IEC 29147:2018,* Geneva: ISO.

juliocesarfort, 2023. *Public Pentesting Reports.* [Online] Available at: https://github.com/juliocesarfort/public-pentesting-reports [Accessed 25 October 2023].

Knowles, W., Baron, A. & McGarr, T., 2016. The simulated security assessment. *Elsevier,* Issue 52, pp. 296-316.

Kousik Barik, A. A. S. D. K. K. A. B., 2021. *Penetration Testing Analysis with Standardized Report Generation,* s.l.: Atlantis Press.

Kumar, H., 2014. Learning Nessus for Penetration Testing. In: A. Aranha, ed. *Learning Nessus for Penetration Testing.* Birmingham: Packt Publishing Ltd., pp. 12-15.

Kyrölä, A., 2022. *Reporting cyber security to management and board of directors,* Jyväskylä: University of Jyväskylä.

Lapouyade, E., 2024. *Welcome to Python-docx-template's documentation!.* [Online] Available at: https://docxtpl.readthedocs.io/en/latest/# [Accessed 26 March 2024].

Material UI, 2024. *Material UI - Overview.* [Online] Available at: https://mui.com [Accessed 24 January 2024].

McGready, A., 2023. *"You Missed A Period Don't Panic" Why Words Matter.* [Online] Available at: https://www.youtube.com/watch?v=NjER7pTWMDE [Accessed 27 October 2023].

Mell, P., Scarfone, K. & Romanosky, S., 2006. *Common Vulnerability Scoring System,* s.l.: IEEE.

Meta, 2024. *DraftJs Overview.* [Online] Available at: https://draftjs.org/docs/getting-started/ [Accessed 27 February 2024].

Microsoft, 2024. *VSCode.* [Online] Available at: https://code.visualstudio.com [Accessed 17 February 2024].

Mitev, N., 2023. *Public Pentest Reports.* [Online] Available at: https://pentestreports.com/reports/ [Accessed 25 October 2023].

Mohd Nizam Zakaria, P. A. P. N. M. S. A. I. M. N. K. O. Y., 2019. *A Review of Standardization for Penetration Testing Reports and Documents,* s.l.: IEEE.

MongoDB, 2024. *MongoDB.* [Online]
Available at: https://www.mongodb.com
[Accessed 4 March 2024].

MongoDB, 2024. *PyMongo Documentation.* [Online]
Available at: https://pymongo.readthedocs.io/en/stable/tutorial.html#
[Accessed 4 March 2024].

Mozilla, 2024. *What is JavaScript?.* [Online]
Available at: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript
[Accessed 21 January 2024].

Murchie, K. J. & Diomede, D., 2020. *Fundamentals of graphic design—essential tools for effective visual science communication,* Chicago: FACETS.

National Cyber Security Centre, 2023. *CHECK - Penetration Testing.* [Online]
Available at: https://www.ncsc.gov.uk/information/check-penetration-testing#:~:text=CHECK%20is%20the%20term%20for,qualifications%20and%20have%20suitable%20experience.
[Accessed 29 February 2024].

NIST, 2023. *Common Vulnerability Scoring System Calculator.* [Online]
Available at: https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator
[Accessed 12 November 2023].

PlexTrac, 2022. *An INSIDE LOOK at Penetration Testing Collaborative Platform (John Hammond).* [Online]
Available at: https://plextrac.com/video/an-inside-look-at-penetration-testing-collaborative-platform-john-hammond/?vPage=3
[Accessed 29 December 2023].

Remix, 2024. *React Router- Getting Started.* [Online]
Available at: https://reactrouter.com/en/main
[Accessed 20 February 2024].

Richardson, L., 2024. *Beautiful Soup Documentation.* [Online]
Available at: https://beautiful-soup-4.readthedocs.io/en/latest/
[Accessed 25 March 2024].

Ronacher, A., 2024. *Flask.* [Online]
Available at: https://flask.palletsprojects.com/en/3.0.x/
[Accessed 24 February 2024].

Salim, L. et al., 2022. *A Literature Review on the Impact of Effective Management in Cyber Security System Performance,* Jakarta: IEEE.

Salonen, S., 2023. *EVALUATION OF UI COMPONENT LIBRARIES IN REACT DEVELOPMENT,* Tampere: Tampere University.

Sharma, S., Pelletier, J. M. & Stackpole, B., 2023. *A Common Pentest Output Schema for Business Intelligence System Ingestion,* Opatija: IEEE.

Speakmon, B., 2024. *Jira 3.8.0.* [Online] Available at: https://pypi.org/project/jira/ [Accessed 20 March 2024].

The Python Software Foundation, 2024. *Python.* [Online] Available at: https://www.Python.org [Accessed 24 February 2024].

Vailshery, L. S., 2024. *Staistica, Most used web frameworks among developers worldwide, as of 2023.* [Online]
Available at: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/
[Accessed 10 April 2024].

vercel, 2024. *SWR: React Hooks for Data Fetching.* [Online] Available at: https://swr.vercel.app [Accessed 5 March 2024].

Walke, J., 2024. *React.* [Online] Available at: https://react.dev [Accessed 21 January 2024].

Yarn, 2024. *Yarn.* [Online] Available at: https://classic.yarnpkg.com/en/ [Accessed 22 March 2024].

Young, J. A., 2020. The Development of a Red Teaming Service-Learning. *Journal of,* 31(3), pp. 157-178.

8. Appendices

8.1. Appendix A – Penetration Test Report Analysis Criteria

Organisation	Overall Rating */10	Top-Level Overview	Executive Summary	Scope of work	Project Objectives	Timeline	Summary of Findings	Summary of recommendations	Methodology	Planning	Exploitation	Reporting
GlitchSecure-Example-Pentest-Report	5.03	5.50	5	8	7	4	4	5	5.00	7	8	0
GlitchSecure-Example-Vulnerability-Assessment-Report	4.52	5.50	5	8	7	4	4	5	5.00	7	8	0
Bishop Fox Assessment Report	4.10	5.67	2	5	4	7	8	8	1.00	0	3	0
chess-cybersecurity-penetration-testing-sample-report	5.05	6.50	5	9	8	2	6	9	6.67	7	7	6
EXAMPLE-Penetration Testing Report v.1.0	3.90	4.83	7	6	5	0	8	3	1.67	0	0	5
TrailOfBits - Kubernetes Final Report	6.94	6.50	3	7	9	6	7	7	8.00	8	7	9
UnderDefense - Anonymised-BlackBox-Penetration-Testing-Report	5.38	7.50	6	6	7	8	9	9	5.00	7	8	0
NCC_Group_-_phpMyAdmin	4.11	5.17	2	7	2	3	9	8	0.00	0	0	0
NCC_Group_-_Ricochet	4.05	4.67	4	5	2	3	7	7	0.00	0	0	0
ISEC_Wikimedia	5.79	6.67	1	8	8	8	6	9	3.67	5	6	0
NCC Group Zcash Crypto Report 2016	5.35	6.00	5	6	5	4	8	8	3.67	4	7	0
Atredis+Partners Mullvad+VPN+Platform+Security Assessment Report	5.33	6.33	6	5	9	6	7	5	7.00	6	8	7
BHP_Mobile_Wallet_Penetration_Test_Report	6.10	6.50	8	9	7	8	7	0	9.00	9	9	9
RedSiege-SampleReport	3.14	3.33	2	0	2	1	6	9	0.00	0	0	0
OffSec penetration-testing-sample-report	3.68	3.67	2	3	5	0	5	7	2.00	2	2	2
VoidSec-Ghost	3.05	2.00	2	0	3	0	7	0	0.00	0	0	0
SecureLayer7-Pentest-report	1.26	2.17	6	5	1	1	0	0	0.00	0	0	0
Report URI - 2020 Penetration Test Report	6.63	8.17	7	9	6	10	8	9	3.00	0	0	9
Cure53 - pentest-report bitwarden	2.00	3.00	2	7	5	4	0	0	0.67	0	0	2
Cynergi-Solutions - eclipse-bank-security-assessment-report-example	6.27	5.833333333	6	10	10	3	6	0	6.00	6	6	6
Description On Each Section	Overall Rating of the Report as a whole	Every section that is related to the top-level overview of the report which would be given to an executive/non-technical user	Summary of important findings within report While also being accessible for an executive	IP Addresses to be tested, type of pen test, and extra info	Provide the objectives that the organization will gain after knowing the risks related to the penetration of the target IP addresses/ system or application and what they will get after mitigating these risks by implementing the recommendations in the penetration testing report.	Testing duration The tested IP address's risks, from pen-testing point of view, during this period only. The pen-tester does not hold any responsibilities if some risk aroused after this period of time due to some changes in the target systems.	In a glance view show the number of discovered risks based on priorities. Avoid language that are overly frightening	Based on the analysis of risks and the high level finding, the high level recommendation for the target organization need to be described.	This section provides the needed information about how the penetration testing was conducted. What steps have been followed to collect the information, analyze them, the risk rating methodology used to calculate the risk for each piece of vulnerability and it may also contain the tools that the pen-tester used for each stage.			

Figure 42 Part 1 of the penetration Test report analysis criteria

Detail Findings	Vulnerabilities	Impact	Likelihood	Risk Evaluation	Recommendation
5.20	8	8	0	3	7
5.00	8	7	0	3	7
4.60	7	6	0	3	7
3.60	4	5	0	4	5
3.40	7	4	0	3	3
6.40	9	6	4	5	8
5.00	8	4	0	4	9
6.40	6	6	5	7	8
6.40	6	6	5	7	8
6.40	7	4	5	7	9
6.60	6	6	5	7	9
3.00	5	2	0	2	6
4.60	3	9	0	2	9
3.80	6	2	0	2	9
5.20	6	8	0	4	8
6.20	3	9	10	9	0
2.00	4	3	0	1	2
7.20	7	6	6	9	8
2.60	3	4	0	3	3
7.40	7	9	7	7	7
This section provides detailed information for each finding. Present the findings in the simplest way as possible. There are a number of ways in which results can be presented. Here are a few: • Tables • Graphs • Pie or Bar charts • Diagrams	For each piece of vulnerability, a clear description should be shown about the source of the vulnerability, its impact and the likelihoods of the vulnerability to be exploited. Report should explain the source of the vulnerability and the root cause of the problem not the symptom of it.	The report should explain the impact of the vulnerability's exploitation by the threat agent	The report should state the likelihood of a vulnerability being exploited by the threat source (e.g. a hacker). Practical penetration tester may think of the likelihood as a combination of ease of access, level of access gained, difficulty of discovering the vulnerability and exploiting it, and the value of the asset to the target organization.	"Process of comparing the estimated risk against given risk criteria to determine the significance of the risk	Based on the risk rating and the target asset, the penetration tester should provide an acceptable recommendation with alternatives. For example, for weak authentication protocols being used to validate accounts for accessing a customer database through the ASP Web application, pen tester may provide more than option for mitigating the risk such as: 1- Implement Public Key Infrastructure (PKI) by providing certificate to all users of the database and require certificate-based authentication on the front-end

Figure 43 Part 2 of the penetration Test report analysis criteria

Additional Importance	References	Appendices	Glossary	Report design & Formatting	Document Control	Use of Illustrations	Table of Contents
4.43	7	7	0	3	3	7	4
2.57	6	0	0	1	1	6	4
5.14	7	8	0	8	0	4	9
3.43	3	0	0	4	10	7	0
5.71	7	9	4	0	10	0	10
6.86	7	10	5	7	6	7	6
4.00	8	7	0	4	0	9	0
4.86	6	8	7	6	4	3	0
5.14	6	8	7	6	6	3	0
6.43	8	7	0	8	8	4	10
5.14	6	8	7	6	6	3	0
5.00	9	4	0	5	6	2	9
4.29	5	4	9	4	7	1	0
5.43	6	3	7	3	1	8	10
3.86	5	3	0	4	1	5	9
4.00	0	7	0	6	2	3	10
0.86	0	0	0	2	0	1	3
8.14	8	8	6	9	9	7	10
1.71	3	0	0	1	2	1	5
5.86	0	9	3	4	10	8	7
Anything that is important to a good report but is not a finding in itself for vulnerabilities. Instead, the additional importance makes it nicer for a user to read.	External links that relate to a piece of information within the report to give a wider context to the reader and allows for additional research from the sources they have used.	An appendix contains additional information related to the report but which is not essential to the main findings. This can be consulted if the reader wishes to but the report should not depend on this. Such information may include the scanning result, vulnerability assessment results, or other information, which may be useful for the reader.	Define the meaning of technical terms	In report planning, page design needs to be decided upon to develop the look and feel of the report. This includes but not limited to the header and footer content, fonts to be used and colors. This will be controlled based on how the service provider's document looks and feels.	Document control will be based on the service provider control of document procedure. Cover Page Document Properties Version Control	The appropriate use of illustrations such as charts, graphs, figures	his will list all sections of the report in a sequence with the page numbers. If the report includes some appendices, the titles of these should be listed but not page numbered.

Figure 44 Part 3 of the penetration Test report analysis criteria

8.2. Appendix B – Complete Penetration Test Report Template

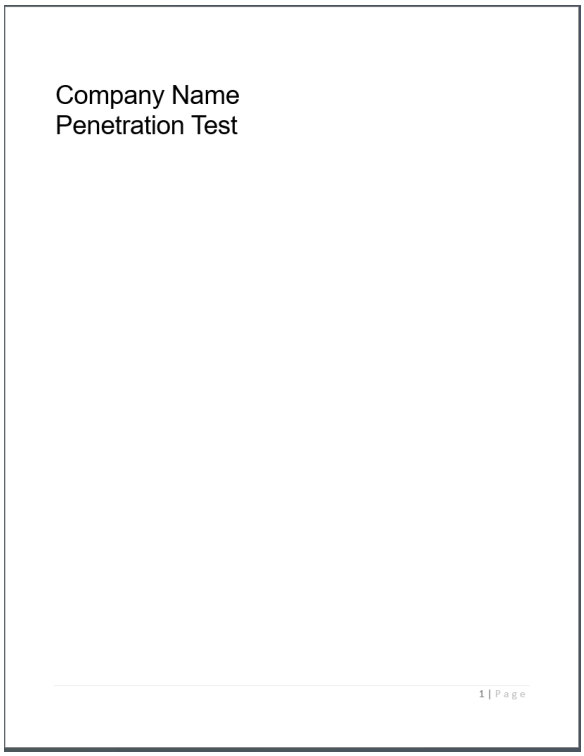


Figure 45 Page 1 of penetration test report template

Document Control

The first page of this document should be the cover page that contains the organisation's logo and the type of test.

In this section include the reporter's name, their role, and a method of contacting them (Email, work No., etc). Additionally, it is best to write the version control and document properties inside of tables to allow a user to quickly locate the information they require without having to read a paragraph of information.

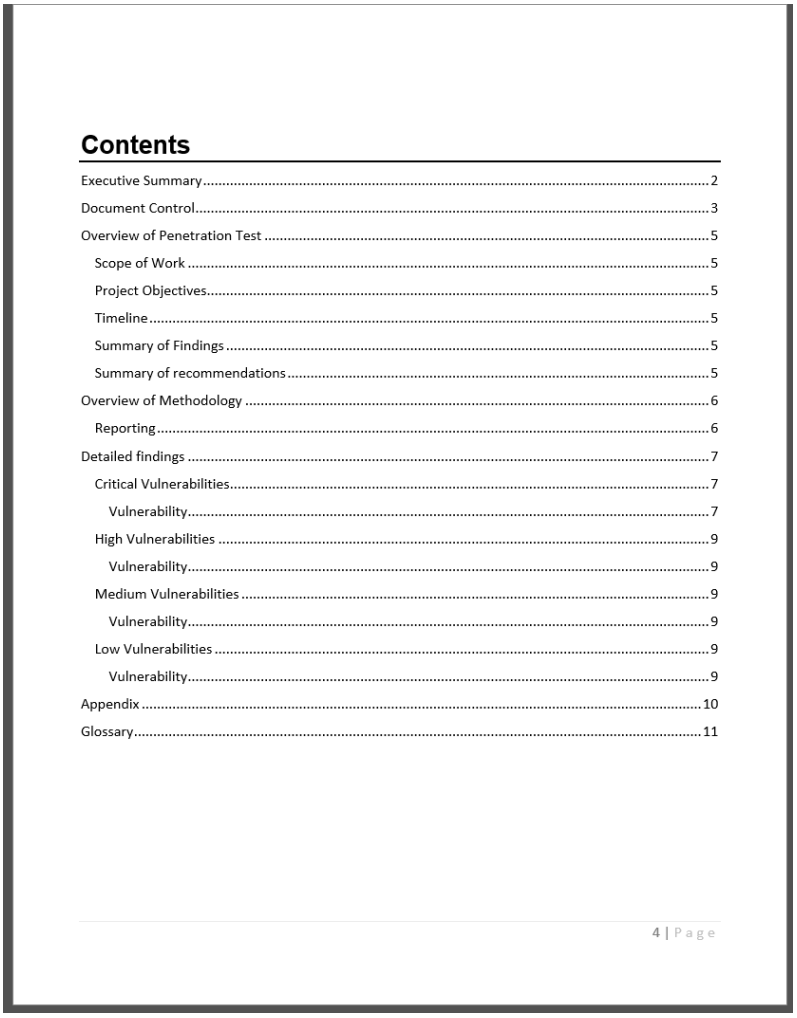
Finally, do not include an introduction section about your organisation since this can be distracting from the main content of the report and a brief introduction should be done in the executive summary.

Figure 46 Page 2 of penetration test report template

Executive Summary

Summary of important findings within report While also being accessible for an executive level individual. Ensure this is a maximum of 2 paragraphs of information and no more. Beyond 2 paragraphs the executive summary becomes more of an essay to read and is no longer fit for purpose.

Figure 47 Page 3 of penetration test report template



Contents

Figure 48 Page 4 of penetration test report template

Overview of Penetration Test

Scope of Work

IP Addresses to be tested, type of pen test. The IP addresses should be inserted into a table for ease of access.

Project Objectives

Provide the objectives that the organization will gain after knowing the risks related to the penetration of the target IP addresses/ system or application and what they will get after mitigating these risks by implementing the recommendations in the penetration testing report.

Timeline

Testing duration

The tested IP address's risks, from pen-testing point of view, during the testing period only. The pen-tester does not hold any responsibilities if some risk aroused after this period due to some changes in the target systems.

Summary of Findings

In a glance view show the number of discovered risks based on priorities.

Avoid language that are overly frightening. The best method of including this information would be to write up a table of the vulnerabilities with their impact to the organisation and risk evaluation. Additionally including a bar chart or a pie chart of the vulnerabilities based on their risk rating would be recommended.

Summary of recommendations

Based on the analysis of risks and the high-level finding, the high-level recommendation for the target organization needs to be described. The best method of including this information is make a sub heading of each vulnerability in order of risk and give a concise summary of how to mitigate the vulnerability that is easy to read and simple to understand.

Figure 49 Page 5 of penetration test report template

Overview of Methodology

This section provides the needed information about how the penetration testing was conducted. What steps have been followed to collect the information, analyze it, the risk rating methodology used to calculate the risk for each piece of vulnerability, and it may also contain the tools that the pen-tester used for each stage.

Reporting

This section should provide the client the information required to understand how the vulnerabilities obtained their vulnerability classification and what factors were considered such as CVSS score using not just the base score but including the environmental and temporal scores as well.

Figure 50 Page 6 of penetration test report template

Detailed findings

This section provides detailed information for each finding. Present the findings in the simplest way as possible.

Critical Vulnerabilities

Vulnerability

Description

For each piece of vulnerability, a clear description should be given about the source of the vulnerability, its impact, and the likelihoods of the vulnerability to be exploited. Report should explain the source of the vulnerability and the root cause of the problem not the symptom of it. Additionally include images if it aids the client in understanding where the vulnerability was caused.

Impact

The report should explain the impact of the vulnerability's exploitation by the threat agent.

Likelihood

The report should state the likelihood of a vulnerability being exploited by the threat source (e.g. a hacker). Practical penetration tester may think of the likelihood as a combination of ease of access,

level of access gained, difficulty of discovering the vulnerability and exploiting it,

and the value of the asset to the target organization.

Risk Evaluation

Process of comparing the estimated risk against given risk criteria to determine the significance of the risk. There is many methods to doing this, a recommended method would be to use the CVSS scoring system.

Recommendation

Based on the risk rating and the target asset, the penetration tester should provide an acceptable recommendation with alternatives. For example, for weak authentication protocols being used to validate accounts for accessing a customer database through the ASP Web application, pen tester may provide more than option for mitigating the risk such as: 1-Implement Public Key Infrastructure (PKI) by providing certificate to all users of the database and require certificate-based authentication on the front-end. Additionally include screenshots or commands if it aids the client in understanding how to

Figure 51 Page 7 of penetration test report template

mitigate the vulnerability. When adding commands, use the "Commands" style which has been created specifically to make the information stick out from the rest of the recommendation.

Sample Command

References

Include external links that relate to a piece of information within the vulnerability to give a wider context to the reader and allow for additional research from the sources they have used. Give each reference a small sentence explaining what additional value the reference gives.

Figure 52 Page 8 of penetration test report template

High Vulnerabilities

Vulnerability

Description
Impact
Likelihood
Risk Evaluation
Recommendation
References

Medium Vulnerabilities

Vulnerability

Description
Impact
Likelihood
Risk Evaluation
Recommendation

Low Vulnerabilities

Vulnerability

Description
Impact
Likelihood
Risk Evaluation
Recommendation

Figure 53 Page 9 of penetration test report template

Appendix

An appendix contains additional information related to the report, but which is not essential to the main findings. This can be consulted if the reader wishes to, but the report should not depend on this. Such information may include the scanning result, vulnerability assessment results, or other information, which may be useful for the reader. It is recommended to separate the Appendix into multiple Appendix sections (Appendix A, Appendix B, Appendix C, etc).

Figure 54 Page 10 of penetration test report template

Glossary

Define the meaning of technical terms which may be confusing to an individual that is not in cyber-security specifically. It would be best to make a bullet point kind of list with the definition under it to allow for an individual to quickly find the definition they are looking for.

Figure 55 Page 11 of penetration test report template

8.3. Appendix C – Performance Review Report

Performance Review for <INSERT NAME>

Employee Name: <INSERT NAME>

Employee ID: <INSERT ID>

Employee Role: <INSERT ROLE>

Client: <INSERT CLIENT>

Report: <INSERT REPORT TYPE>

Figure 56 Page 1 of performance review report

Document Control

The first page of this document should be the cover page that contains the organisation's logo and the type of test.

In this section include the reporter's name, their role, and a method of contacting them (Email or work phone number). Additionally, it is best to write the version control and document properties inside of tables to allow an employee to quickly locate the information they require without having to read a paragraph of information.

Figure 57 Page 2 of performance review report

Strengths of the report

The reporter writing this section should identify the strengths of the report and highlight them with a small list of bullet points to describe what made that section a success (Example Below). The reporter should also provide a summary of the strengths at the top of this section in either bullet points or a brief description to allow the employee to understand everything they performed well on. Additionally, if this is not the first report, the reporter should highlight the improvements made by the employee based on the "Areas of Improvement" section with a description on how they improved in that area. Finally, ensure the feedback is written in a clear and concise manner that incorporates the five main components: Relevance, Accuracy, Timely, Specific, and Easy to Understand. REMOVE THIS TEXT ONCE REPORT IS COMPLETE.

Remote Code Execution (RCE) vulnerabilities allow an unauthorised attacker to potentially steal sensitive information or cause downtime for the organisation's critical services. Which would damage the reputation of the organization.

- The explanation on Remote Code Execution is concise and describes the potential damage that could affect the client.
- The language used is not fear mongering to an extreme level and gives a realistic example of what could happen should an RCE of high or critical level be exploited.
- The language is simple and not too technical, which makes it easier for a client to read without having to decipher the information.
- No spelling mistakes or grammatical errors.

Figure 58 Page 3 of performance review report

Areas of improvement for the report

The reporter writing this section should identify the Areas of Improvement in the report and highlight them with a small list of bullet points to describe what made that section a success (Example Below). The reporter should also provide a summary of improvements with goals and methods to reach these goals at the top of this section. The summary should be in bullet points with short descriptions for clarification if required. This allows the employee to understand the areas they need to improve on without confusion or ambiguity. Additionally, include resources that may be useful to the employee when trying to improve their report writing skills such as online learning resources, or in house training that is available when they are not working on billable hours of work with clients. Finally. Ensure the feedback is written in a clear and concise manner that incorporates the five main components: Relevance, Accuracy, Timely, Specific, and Easy to Understand. Using these five components aids in identifying where the employee needs to improve in their report writing skills and how to reach that level of improvement. REMOVE THIS TEXT ONCE REPORT IS COMPLETE.

Remote Code Execution (RCE) vulnerabilities allow an unauthorised attacker to potentially steal sensitive informationn or cause downtime for the organisation's critical services. Which would damage the reputation of the organisation.

- While the explanation is good, change the "would" to "could" as the vulnerability is not guaranteed to damage the reputation of the organisation.
- Spelling mistake "sensitive informationn" should be "sensitive information".
- For Remote Code Execution, include in the brackets to look at glossary as it provides further information about Remote Code Execution. For example, "Remote Code Execution (See Glossary for definition).

Figure 59 Page 4 of performance review report

Additional Information

The reporter should use this section to include any additional information that would be useful to the employee as well as resources with clarification to why the following resource is useful to the employee. This section can also include images to ensure the above two sections do not get cluttered with images.

Figure 60 Page 5 of performance review report

8.4. Appendix D – Vulnerability Descriptions

Critical

Unsupported Windows OS (remote)

Description

The remote Operating System found on {DEVICE NAME} is no longer supported by the developer or is possibly missing a service pack. The unsupported OS likely contains multiple security vulnerabilities.

Impact

The possible vulnerabilities from an outdated Windows OS could range from Remote Code Execution to Denial of Service, which would allow an unauthorised attacker to gain access to sensitive information or disrupt a section of the network, leading to downtime for the organisation's online services.

Likelihood

The unsupported Windows OS was discovered by running a quick OS discovery scan with Nmap, which required little effort to conduct, an attacker is likely to discover the machine on the network.

With the out-of-date OS, vulnerabilities will be easily discoverable once the version number has been identified.

The Nmap command ran to discover the vulnerability was:

```
Nmap -O $host
```

Risk Evaluation

Based on the severe impact and high likelihood, the vulnerability has been categorised as a critical vulnerability.

Recommendation

If possible, the machine should be updated to a more recent OS such as Windows 10/11.

However, if it is not possible due to the services, alternative security controls should be implemented to mitigate the likelihood and impact that this vulnerability could cause.

Figure 61 Unsupported Windows OS vulnerability

HP System Management Homepage SetSMHData admin-group Parameter Handling RCE

Description

The HP System Management Homepage (SMH) running on the machine {DEVICE NAME} is affected by a remote code execution vulnerability due to an overflow condition in the mod_smh_config.so library. This is caused by improper validation for supplying user input to the /Proxy/SetSMHData endpoint. This can be exploited by an unauthenticated attacker by crafting a request.

Impact

The exploitation would lead to a denial of service or remote code execution, which would allow an unauthorised attacker to bring the online service down causing downtime for the organisation or an attacker could gather sensitive information stored on the host.

Likelihood

As the vulnerability was found in an older version of HP SMH, it would be simple for an unauthorised attacker to discover the version of the HP SMH by scanning the network and using web developer tools against the webpage of the SMH to discover the outdated version.

Risk Evaluation

Based on the severe impact and high likelihood, the vulnerability has been categorised as a critical vulnerability.

Recommendation

If possible, the organisation should Upgrade to the latest version of HP SMH.

Otherwise, the organisation should implement security controls to mitigate the severity of the vulnerability.

Figure 62 HP System Management Homepage parameter handling RCE vulnerability

High

Juniper Junos OS Vulnerability (JSA75739)

Description

The Juniper Junos OS version on the machine {DEVICE NAME}, is affected by a vulnerability that allows an attacker to cause a denial of service for the host. The vulnerability is caused by an improper validation in the input in the Routing Protocol Daemon of Juniper.

Impact

Denial of service could potentially lead to downtime for the organisation's services which could halt service for clients as well as employees in the organisation.

Likelihood

The outdated OS was discovered by running a quick OS discovery scan with Nmap, which required little effort to conduct, an attacker is likely to discover the machine on the network.

With the out-of-date OS, vulnerabilities will be easily discoverable once the version number has been identified.

The Nmap command ran to discover the vulnerability was:

```
Nmap -O $host
```

Risk Evaluation

Based on the high impact and likelihood for being exploited, the vulnerability has been categorised as a high vulnerability.

Recommendation

If possible, Update the Juniper Junos OS to the latest version with all available software patches.

Otherwise mitigate the severity of the vulnerability through alternative methods such as additional security controls.

Figure 63 Juniper Junos OS vulnerability

Ubuntu 20.04 LTS / 22.04 LTS / 23.10 : Squid vulnerabilities (USN-6728-1)

Description

The Ubuntu machine {DEVICE NAME} has multiple packages installed all of which contain a denial-of-service vulnerability. The packages are all related to squid, which is a caching proxy for the web.

Impact

Denial of service could potentially lead to downtime for the organisation's services which could halt service for clients as well as employees in the organisation.

Likelihood

The affected packages were discovered by running a quick scan with Nmap, which required little effort to conduct.

With affected packages, vulnerabilities will be easily discoverable once the version number has been identified.

Risk Evaluation

Based on the high impact and likelihood for being exploited, the vulnerability has been categorised as a high vulnerability.

Recommendation

The affected packages should be removed if not required or updated if possible.

Figure 64 Ubuntu 20.04 LTS Squid vulnerabilities

Medium

SMB Signing Not Required

Description

The hosts {Device Names} do not require authentication on the remote SMB server.

Impact

Due to this vulnerability, a remote unauthorised attacker could conduct man-in-the-middle attacks against the server, intercepting SMB traffic. The traffic could potentially contain sensitive information for an attacker to steal.

Likelihood

SMB Signing not required is a common vulnerability which appears in many networks, which leads to attackers attempting to exploit it as apart of a standard procedure during an attack.

To determine if the hosts above are vulnerable to this type of attack, an Nmap scan was performed using the "smb-security-mode" script. The command used was:

```
nmap -p137,139,445 --script smb-security-mode $host
```

Risk Evaluation

Based on the medium impact and medium likelihood for being exploited, the vulnerability has been categorised as a medium vulnerability.

Recommendation

To remediate this vulnerability, if possible, enforce message signing in the host's configuration. To do this on Windows, find the policy setting 'Microsoft network server: Digitally sign communications (always)'. On Samba, the setting will be called 'server signing'.

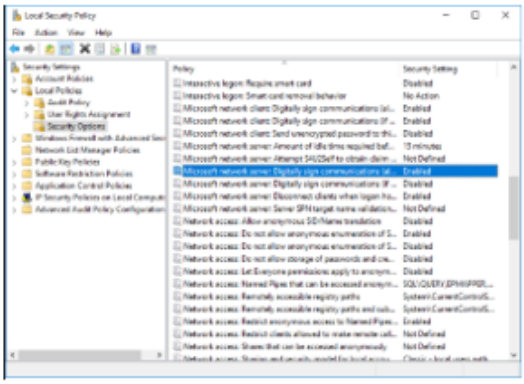


Figure 65 SMB signing not required vulnerability

IP Forwarding Enabled

Description

The hosts {DEVICE NAMES} have IP forwarding enabled.

Impact

IP forwarding could be exploited by an attacker to route packets through the hosts and bypass firewalls, routers, and/or NAC filtering.

Likelihood

The chance of an attacker using this for an attack is not likely compared to other methods discovered in this assessment however, the damage that can occur if this is used to bypass a firewall could be detrimental to the organisation.

Risk Evaluation

Based on the high impact and low likelihood for being exploited, the vulnerability has been categorised as a medium vulnerability.

Recommendation

Unless the hosts are routers, it is recommended that IP forwarding is disabled. On Linux, this can be done by inputting:

```
0 > /proc/sys/net/ipv4/ip_forward
```

On Windows, the key 'IPEnableRouter' should be set to 0 under:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters
```

Figure 66 IP forwarding enabled vulnerability

Low

SSLv3 Padding Oracle On Downgraded Legacy Encryption Vulnerability (POODLE)

Description

The host is affected by a Man-In-The-Middle attack known as POODLE. This is an information disclosure vulnerability. This vulnerability is caused by the way SSL 3.0 handles padding bytes when decrypting messages that are encrypted using block ciphers in cipher block chaining (CBC) mode.

Impact

Attackers could decrypt a selected byte of a cipher text in as few as 256 tries if they are able to force a victim application to repeatedly send the same data over newly created SSL 3.0 connections.

Likelihood

To verify the presence of this vulnerability the following command was used:

```
sudo nmap -sV -version-light -script ssl-poodle -p- $host
```

While the command is easy to identify, there is a requirement of forcing an application to send the same data over a request and exploit it as well. This makes the vulnerability less likely to be exploited.

Risk Evaluation

Based on the low impact and low likelihood for being exploited, the vulnerability has been categorised as a low vulnerability.

Recommendation

Disable SSLv3. Services that must support SSLv3 should enable the TLS Fallback SCSV mechanism until a point at which SSLv3 can be disabled.

Figure 67 SSLv3 Passing Oracle on downgraded legacy encryption vulnerability

DHCP Server Detection

Description

It is possible to query the DHCP server, so it exposes information about its associated network.

Impact

An attacker can use the information they gather from the DHCP server to familiarise themselves with the organisation's network.

Likelihood

A simple Nmap command was used to gather the information from the DHCP server:

```
nmap -sU -p 67 --script=dhcp-discover $host
```

This command can be simply run by unauthorised attackers to gather information about a network.

Risk Evaluation

Based on the minimal impact and high likelihood for being exploited, the vulnerability has been categorised as a low vulnerability.

Recommendation

Remove any options that are not in use and apply filtering to keep the information off the network.

Figure 68 DHCP server detection vulnerability

8.5. Appendix E – Exported Report from Report Writing Tool

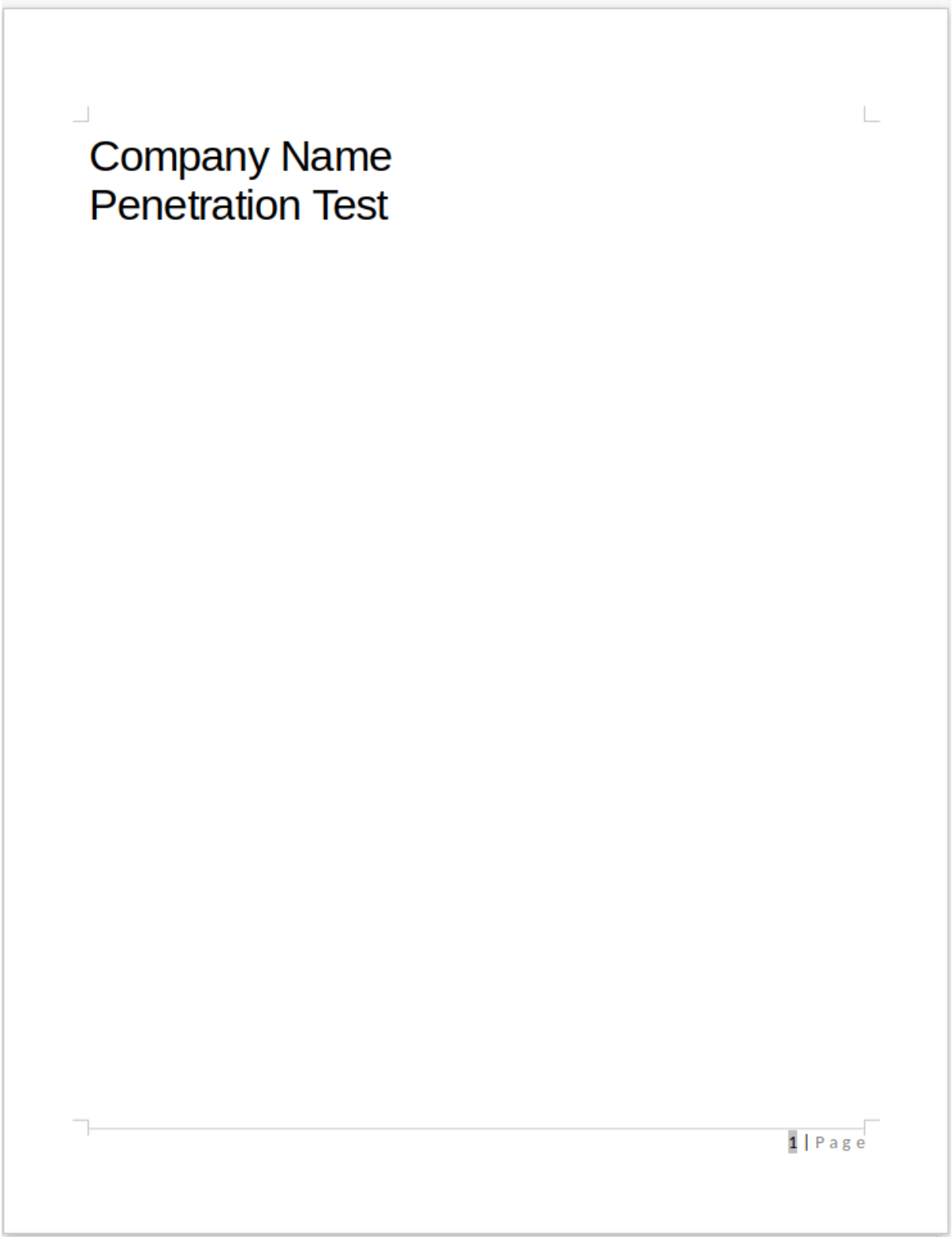


Figure 69 Page 1 of exported report from report writing tool

Document Control

The first page of this document should be the cover page that contains the organisation's logo and the type of test.

In this section include the reporter's name, their role, and a method of contacting them (Email, work No., etc). Additionally, it is best to write the version control and document properties inside of tables to allow a user to quickly locate the information they require without having to read a paragraph of information.

Finally, do not include an introduction section about your organisation since this can be distracting from the main content of the report and a brief introduction should be done in the executive summary.

Figure 70 Page 2 of exported report from report writing tool

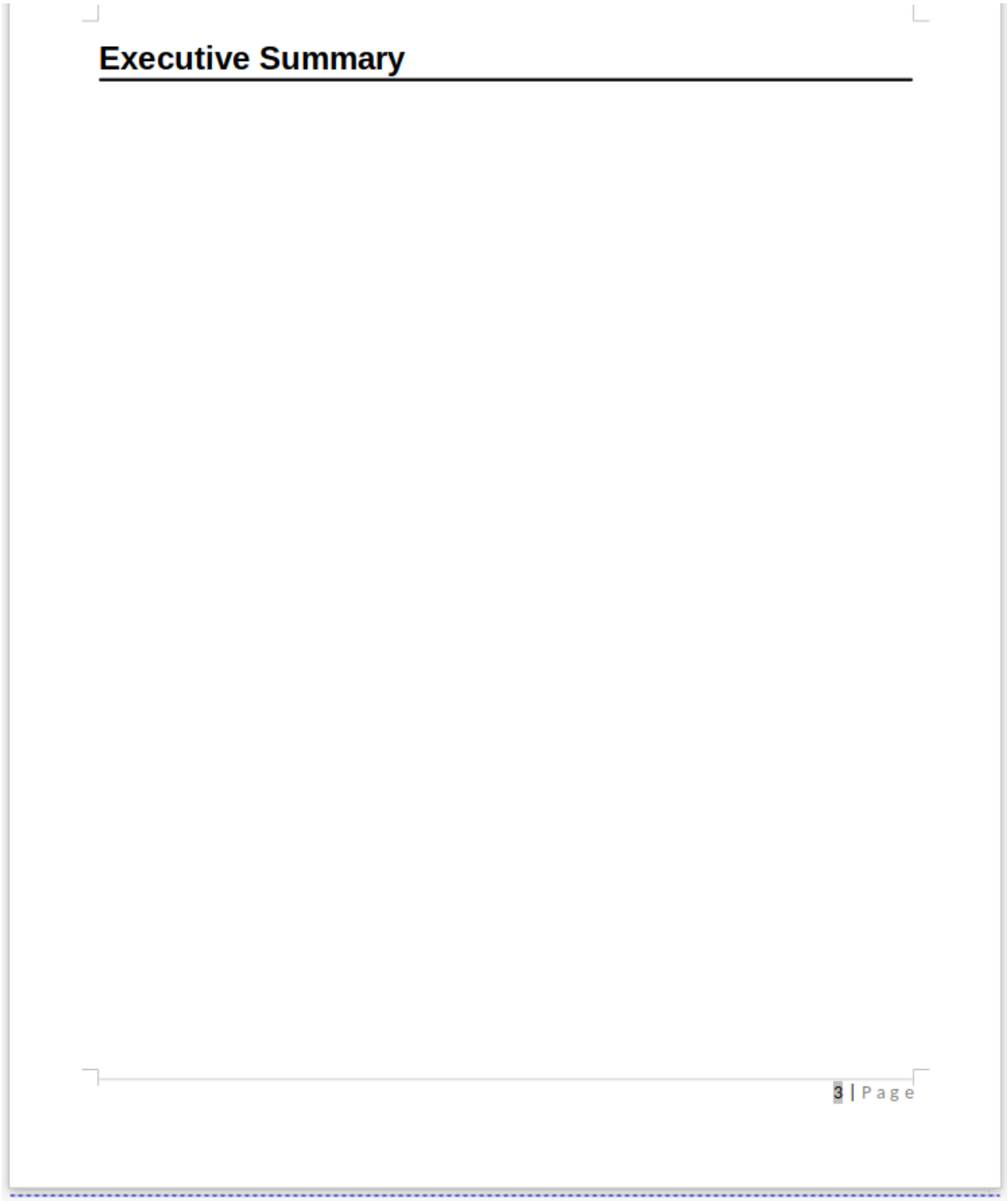


Figure 71 Page 3 of exported report from report writing tool

Contents

Figure 72 Page 4 of exported report from report writing tool

Overview of Penetration Test

Scope of Work

Project Objectives

Timeline

Summary of Findings

Figure 73 Page 5 of exported report from report writing tool

Summary of recommendations

Overview of Methodology

Reporting

Figure 74 Page 6 of exported report from report writing tool

Detailed findings

Critical Vulnerabilities

Unsupported Windows OS (remote)

Description

The remote Operating System found on {DEVICE NAME} is no longer supported by the developer or is possibly missing a service pack. The unsupported OS likely contains multiple security vulnerabilities.

Impact

The possible vulnerabilities from an outdated Windows OS could range from Remote Code Execution to Denial of Service, which would allow an unauthorised attacker to gain access to sensitive information or disrupt a section of the network, leading to downtime for the organisation's online services.

Likelihood

The unsupported Windows OS was discovered by running a quick OS discovery scan with Nmap, which required little effort to conduct, an attacker is likely to discover the machine on the network. With the out-of-date OS, vulnerabilities will be easily discoverable once the version number has been identified. The Nmap command ran to discover the vulnerability was:

Nmap -O $host

Nmap -O $host

Risk Evaluation

Figure 75 Page 7 of exported report from report writing tool

Based on the severe impact and high likelihood, the vulnerability has been categorised as a critical vulnerability.

Recommendation

If possible, the machine should be updated to a more recent OS such as Windows 10/11.

However, if it is not possible due to the services, alternative security controls should be implemented to mitigate the likelihood and impact that this vulnerability could cause.

HP System Management Homepage SetSMHData admin-group Parameter Handling RCE

Description

The HP System Management Homepage (SMH) running on the machine {DEVICE NAME} is affected by a remote code execution vulnerability due to an overflow condition in the mod_smh_config.so library. This is caused by improper validation for supplying user input to the /Proxy/SetSMHData endpoint. This can be exploited by an unauthenticated attacker by crafting a request.

Impact

The exploitation would lead to a denial of service or remote code execution, which would allow an unauthorised attacker to bring the online service down causing downtime for the organisation or an attacker could gather sensitive information stored on the host.

Likelihood

As the vulnerability was found in an older version of HP SMH, it would be simple for an unauthorised attacker to discover the version of the HP SMH by scanning the network and using web developer tools against the webpage of the SMH to discover the outdated version.

Figure 76 Page 8 of exported report from report writing tool

Based on the severe impact and high likelihood, the vulnerability has been categorised as a critical vulnerability.

Recommendation

If possible, the organisation should Upgrade to the latest version of HP SMH.

Otherwise, the organisation should implement security controls to mitigate the severity of the vulnerability.

Figure 77 Page 9 of exported report from report writing tool

High Vulnerabilities

Juniper Junos OS Vulnerability (JSA75739)

Description

The Juniper Junos OS version on the machine {DEVICE NAME}, is affected by a vulnerability that allows an attacker to cause a denial of service for the host. The vulnerability is caused by an improper validation in the input in the Routing Protocol Daemon of Juniper.

Impact

Denial of service could potentially lead to downtime for the organisation's services which could halt service for clients as well as employees in the organisation.

Likelihood

The outdated OS was discovered by running a quick OS discovery scan with Nmap, which required little effort to conduct, an attacker is likely to discover the machine on the network.

With the out-of-date OS, vulnerabilities will be easily discoverable once the version number has been identified.

The Nmap command ran to discover the vulnerability was:

Nmap -O $host

Figure 78 Page 10 of exported report from report writing tool

Nmap -O $host

Risk Evaluation

Based on the high impact and likelihood for being exploited, the vulnerability has been categorised as a high vulnerability.

Recommendation

If possible, Update the Juniper Junos OS to the latest version with all available software patches.

Otherwise mitigate the severity of the vulnerability through alternative methods such as additional security controls.

Ubuntu 20.04 LTS / 22.04 LTS / 23.10 : Squid vulnerabilities (USN-6728-1)

Description

The Ubuntu machine {DEVICE NAME} has multiple packages installed all of which contain a denial-of-service vulnerability. The packages are all related to squid, which is a caching proxy for the web.

Impact

Denial of service could potentially lead to downtime for the organisation's services which could halt service for clients as well as employees in the organisation.

Figure 79 Page 11 of exported report from report writing tool

Likelihood

The affected packages were discovered by running a quick scan with Nmap, which required little effort to conduct.

With affected packages, vulnerabilities will be easily discoverable once the version number has been identified.

Risk Evaluation

Based on the high impact and likelihood for being exploited, the vulnerability has been categorised as a high vulnerability.

Recommendation

The affected packages should be removed if not required or updated if possible.

Figure 80 Page 12 of exported report from report writing tool

Medium Vulnerabilities

IP Forwarding Enabled

Description

The hosts {DEVICE NAMES} have IP forwarding enabled.

Impact

IP forwarding could be exploited by an attacker to route packets through the hosts and bypass firewalls, routers, and/or NAC filtering.

Likelihood

The chance of an attacker using this for an attack is not likely compared to other methods discovered in this assessment however, the damage that can occur if this is used to bypass a firewall could be detrimental to the organisation.

Risk Evaluation

Based on the high impact and low likelihood for being exploited, the vulnerability has been categorised as a medium vulnerability.

Recommendation

Figure 81 Page 13 of exported report from report writing tool

Unless the hosts are routers, it is recommended that IP forwarding is disabled. On Linux, this can be done by inputting:

0 > /proc/sys/net/ipv4/ip_forward

0 > /proc/sys/net/ipv4/ip_forward

On Windows, the key 'IPEnableRouter' should be set to 0 under:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters

SMB Signing Not Required

Description

The hosts {Device Names} do not require authentication on the remote SMB server.

Impact

Due to this vulnerability, a remote unauthorised attacker could conduct man-in-the-middle attacks against the server, intercepting SMB traffic. The traffic could potentially contain sensitive information for an attacker to steal.

Likelihood

SMB Signing not required is a common vulnerability which appears in many networks, which leads to attackers attempting to exploit it as apart of a standard procedure during

Figure 82 Page 14 of exported report from report writing tool

an attack.

To determine if the hosts above are vulnerable to this type of attack, an Nmap scan was performed using the "smb-security-mode" script. The command used was:

nmap -p137,139,445 --script smb-security-mode $host

nmap -p137,139,445 --script smb-security-mode $host

Risk Evaluation

Based on the medium impact and medium likelihood for being exploited, the vulnerability has been categorised as a medium vulnerability.

Recommendation

To remediate this vulnerability, if possible, enforce message signing in the host's configuration. To do this on Windows, find the policy setting 'Microsoft network server: Digitally sign communications (always)'. On Samba, the setting will be called 'server signing'.

Figure 83 Page 15 of exported report from report writing tool

Low Vulnerabilities

SSLv3 Padding Oracle On Downgraded Legacy Encryption Vulnerability (POODLE)

Description

The host is affected by a Man-In-The-Middle attack known as POODLE. This is an information disclosure vulnerability. This vulnerability is caused by the way SSL 3.0 handles padding bytes when decrypting messages that are encrypted using block ciphers in cipher block chaining (CBC) mode.

Impact

Attackers could decrypt a selected byte of a cipher text in as few as 256 tries if they are able to force a victim application to repeatedly send the same data over newly created SSL 3.0 connections.

Likelihood

To verify the presence of this vulnerability the following command was used:

sudo nmap -sV –version-light –script ssl-poodle –p- $host

sudo nmap -sV –version-light –script ssl-poodle –p- $host

While the command is easy to identify, there is a requirement of forcing an application to send the same data over a request and exploit it as well. This makes the vulnerability

Figure 84 Page 16 of exported report from report writing tool

less likely to be exploited.

Risk Evaluation

Based on the low impact and low likelihood for being exploited, the vulnerability has been categorised as a low vulnerability.

Recommendation

Disable SSLv3. Services that must support SSLv3 should enable the TLS Fallback SCSV mechanism until a point at which SSLv3 can be disabled.

DHCP Server Detection

Description

It is possible to query the DHCP server, so it exposes information about its associated network.

Impact

An attacker can use the information they gather from the DHCP server to familiarise themselves with the organisation's network.

Likelihood

A simple Nmap command was used to gather the information from the DHCP server:

Figure 85 Page 17 of exported report from report writing tool

nmap -sU -p 67 --script=dhcp-discover $host

nmap -sU -p 67 --script=dhcp-discover $host

This command can be simply run by unauthorised attackers to gather information about a network.

Risk Evaluation

Based on the minimal impact and high likelihood for being exploited, the vulnerability has been categorised as a low vulnerability.

Recommendation

Remove any options that are not in use and apply filtering to keep the information off the network.

Figure 86 Page 18 of exported report from report writing tool

Appendix

Figure 87 Page 19 of exported report from report writing tool

Figure 88 Page 20 of exported report from report writing tool