**DESIGN AND DEVELOPMENT OF AN**

**AUTOMATED COMMAND-LINE VULNERABILITY SCANNER TOOL**

**BY**

**KANSIIME YOWERI**

**2020-08-01885**

**MUHOOZI OWEN**

**2020-08-01678**

**ARUHO ISAAC**

**2020-08-01316**

**SUPERVISED**

**By**

**Ms. NAKAJUBI SHAMUSI**

**A PROPOSAL SUBMITTED TO THE SCHOOL OF MATHEMATICS AND COMPUTING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS**

**FOR THE AWARD OF A BACHELOR'S DEGREE IN INFORMATION TECHNOLOGY OF KAMPALA INTERNATIONAL UNIVERSITY UGANDA.**

**MARCH 2023**

# **DECLARATION**

We declare that this project proposal is our original work and has never been submitted to any University or higher institution of higher learning for any academic award where other people’s work has been used, due acknowledgement has been made.

Signature.............................................................. Date........../................/.................

**KANSIIME YOWERI**

**2020-08-01885**

Department of Information Technology

School of Mathematic and Computing Kampala international University.

Signature.............................................................. Date........../................/.................

**MUHOOZI OWEN**

**2020-08-01678**

Department of Information Technology

School of Mathematic and Computing Kampala international University.

Signature.............................................................. Date........../................/.................

**ARUHO ISAAC**

**2020-08-01316**

Department of Information Technology

School of Mathematic and Computing Kampala international University.

**APPROVAL**

“I confirm that the work reported in this proposal was carried out by the candidates under my supervision.”

NAME OF THE SUPERVISOR………………………………………………….

SIGNATURE…………………………………………………….

DATE…………………………………………………………………

**TABLE OF CONTENT**

Contents

[**DECLARATION** 0](#_Toc132821742)

[**CHAPTER ONE** 5](#_Toc132821743)

[1.0 Background 5](#_Toc132821744)

[1.1 Statement of the problem 6](#_Toc132821745)

[1.2 Objectives. 7](#_Toc132821746)

[1.2.1 General Objective 7](#_Toc132821747)

[1.2.2 Specific Objectives 7](#_Toc132821748)

[1.2.3 Research Question 7](#_Toc132821749)

[1.3. Scope of the Study 7](#_Toc132821750)

[1.3.0 Content scope 7](#_Toc132821751)

[1.3.1 Time scope 7](#_Toc132821752)

[1.4 Significance of the Study 8](#_Toc132821753)

[**CHAPTER TWO** 9](#_Toc132821754)

[**LITERATURE REVIEW** 9](#_Toc132821755)

[2.0 Introduction 9](#_Toc132821756)

[2.1 Related studies 9](#_Toc132821757)

[2.2.1 Nessus (Columbia, Maryland) 9](#_Toc132821758)

[2.2.2 Acunetix (Malta) 10](#_Toc132821759)

[2.2.3 Rapid7 scanner (USA) 12](#_Toc132821760)

[2.2.4 Nikto (USA) 13](#_Toc132821761)

[2.2.5 NMAP (USA) 14](#_Toc132821762)

[2.4 Gaps identified 18](#_Toc132821763)

[**CHAPTER THREE** 20](#_Toc132821764)

[**RESEARCH METHODOLOGY** 20](#_Toc132821765)

[3.0 Introduction 20](#_Toc132821766)

[3.1 Research Approach 20](#_Toc132821767)

[3.2 Target Population 20](#_Toc132821768)

[3.3 Sampling technique 21](#_Toc132821769)

[3.3.1 Sample size. 21](#_Toc132821770)

[3.4 System Development Methodologies 21](#_Toc132821771)

[3.5 Functional requirements 22](#_Toc132821772)

[3.5.1 Other Requirements 22](#_Toc132821773)

[3.6 Data collection methods. 23](#_Toc132821774)

[3.6.1 Observation 23](#_Toc132821775)

[3.6.2 Interview 23](#_Toc132821776)

[3.6.3 Questionnaires. 24](#_Toc132821777)

[3.6.4 Documentary review 24](#_Toc132821778)

[3.7 System Design and Interface Techniques 25](#_Toc132821779)

[3.8 Testing 25](#_Toc132821780)

[3.8.1 Tools for Implementation 25](#_Toc132821781)

[3.8.2 Deliverables 26](#_Toc132821782)

[3.8.3 Stakeholders 26](#_Toc132821783)

[3.9 Conclusion 27](#_Toc132821784)

[**CHAPTER FOUR** 28](#_Toc132821785)

[**SYSTEMS ANALYSIS AND DESIGN** 28](#_Toc132821786)

[4.0 Introduction. 28](#_Toc132821787)

[4.1 System study. 28](#_Toc132821788)

[4.2 weaknesses of existing scanners. 28](#_Toc132821789)

[4.3 Strengths of existing scanners 29](#_Toc132821790)

[4.4 Requirement specification. 29](#_Toc132821791)

[4.4.1 Functional requirements 30](#_Toc132821792)

[4.4.2 Nonfunctional requirements. 30](#_Toc132821793)

[4.5 System requirements 31](#_Toc132821794)

[4.5.1 Hard ware requirements 31](#_Toc132821795)

[4.5.2 Software requirements 31](#_Toc132821796)

[4.6 Structured Requirement Analysis. 32](#_Toc132821797)

[4.6.1 Flow Chart 32](#_Toc132821798)

[4.6.2 Data Flow Diagram 33](#_Toc132821799)

[**CHAPTER FIVE** 34](#_Toc132821800)

[**SYSTEM IMPLEMENTATION, TESTING AND EVALUATION** 34](#_Toc132821801)

[5.0 Introduction 34](#_Toc132821802)

[5.1 Implementation 34](#_Toc132821803)

[5.1.1 Implementation strategy 34](#_Toc132821804)

[5.1.2 User Guide 35](#_Toc132821805)

[5.1.3 Program Deployment 35](#_Toc132821806)

[5.1.4 Functional requirements output 36](#_Toc132821807)

[5.1.5 Vulnerability detection 37](#_Toc132821808)

[5.1.6 Report Generation 38](#_Toc132821809)

[5.2 Testing 40](#_Toc132821810)

[5.2.1 Performance Testing 40](#_Toc132821811)

[5.2.2 Functionality Testing 41](#_Toc132821812)

[5.2.3 Challenges Faced 42](#_Toc132821813)

[**CHAPTER SIX** 44](#_Toc132821814)

[**CONCLUSION AND RECOMMENDATION CONCLUSION** 44](#_Toc132821815)

[6.0 Introduction 44](#_Toc132821816)

[6.1 CONCLUSION 44](#_Toc132821817)

[6.2 Recommendations 44](#_Toc132821818)

[6.3 Future study 45](#_Toc132821819)

[**REFERENCES** 46](#_Toc132821820)

# **CHAPTER ONE**

**INTRODUCTION**

## 1.0 Background

A security vulnerability is a weakness that may be exploited to cause damage, but its presence does not cause harm by itself (Jeeva, Raveena, Sangeetha, & Vinothini, 2016). Grabber, Vega, Acunetix, Wapiti (InfoSec Institute, 2014) are few examples of web vulnerability scanners. The Cloud Security Alliance (2016) has recently identified twelve major types of security concerns and threats. Many of these are relevant to areas where web vulnerability scanners may be helpful in reducing risks. For example, insecure APIs and insufficient due diligence, c, weak passwords and misconfigurations may be overlooked though they are of a great risk.

Software vulnerabilities are found in commonly used software products every day. More than 45

000 software vulnerabilities have been published by the National Vulnerability Database and over 4500 of these software vulnerabilities were published during 2015 alone (NVD, 2016). Efficient management of vulnerabilities is thus an important activity in modern enterprises security efforts. To manually keep track of all vulnerabilities, present in systems and remediate them appropriately is a daunting task (Sommestad, Ekstedt, Holm, & Afzal, 2015). Fortunately, there are tools aimed to provide automated support for this process available (Al-Ayed et al. 2017). One commonly applied solution involves the use of network vulnerability scanners (Werlinger et al. 2018). A network vulnerability scanner is an appliance or software which is used to scan the architecture of a network and report any identified vulnerabilities. The normal procedure of scanning a network with a vulnerability assessment tool generally involves three parts: network scanning, vulnerability scanning and vulnerability analysis (Manzuik et al. 2017). Network scanning involves identifying which hosts that are alive in the computer network, which operating systems that they use, and what services they run. During the vulnerability scan a database of vulnerability signatures is compared to the information obtained from a network scan to produce a list of vulnerabilities that are presumably present in the network. Most tools thereafter attempt to verify the presence of these vulnerabilities through signatures – carefully constructed queries which aim to verify the vulnerability’s presence without disrupting the service. Some tools also provide the possibility of actually exploiting vulnerabilities to fully verify their presence. Vulnerability analysis concern evaluating the severity of identified vulnerabilities. Organizations typically have a large number of vulnerabilities in their operational environment and some vulnerabilities lead to a higher security risk than others. For instance, some software vulnerabilities are easy to utilize, and some have dire consequences if they are exploited. It is thus important to assess the most significant problems and remediate these first. Powerful vulnerability analysis would certainly provide great value to the vulnerability management process. However, most vulnerability management tools are fairly immature in this respect. In academia there are several projects which have tried to solve this problem. A few examples include (MulVAL, 2015; Homer & Ou, 2016; NetSPA, 2016; TVATool (Noel, 2014); Jajodia & Noel 2015), incident response intelligence systems (Patsos, 2018) and work by Sommestad (T. Sommestad, 2016). All these analysis tools require detailed data about the vulnerabilities present in a network. The quality of the information produced by network and vulnerability scanning is thus of importance, regardless if an evaluation tool is used for the analysis or if it is done manually. Unfortunately, these scans are not always correct. Network and vulnerability scanning base their assessments on signatures of operating systems used, services running, and their corresponding vulnerabilities. These signatures do not always provide the correct result, which causes issues for the security management staff. Sometimes these issues result in failure to identify existing vulnerabilities (i.e. false negatives); sometimes they result in erroneously reporting inexistent vulnerabilities to be present (i.e. false positives). If scans produce vulnerability lists containing these errors it will impede efficient mitigation – false positives will result in efforts to manage nonexistent problems and false negatives may lead to unexpected security problems. While vulnerability scanners depend on their accuracy there have been no thorough evaluations to assess their performance. Nor has it been tested if there is a difference in accuracy between scans of different operating systems.

Moreso, With the advancements in mobile computing, web services, and browser based applications, most business rely on conducting their business communications and transactions online. However, these websites and web applications are not completely secure. Around 30,000 websites are being attacked every day (Lyne, 2017), and one out of every three websites is vulnerable to hacking (Schupak, 2016). Moreover, ninety percent of passwords are vulnerable to being stolen (Warman, 2017). The increasing number of websites and online applications increases the urgency of securing these websites. Web security scanners are automated tools that check out websites or web applications for security vulnerabilities, without accessing the application’s source code (Saeed, 2014). Web vulnerability scanners help to find vulnerabilities of web applications and websites.

## 1.1 Statement of the problem

Companies are increasingly digitizing their business operations and processes, leading to increase security risks. One of the major risks is a hacker exploiting a vulnerability that exists within an organization's IT infrastructure. To mitigate the risk of a security incident, companies need to be able to prevent, detect, respond and recover from such attacks. This has created space for developing an automated vulnerability scanner that can detect all the possible vulnerabilities and misconfigurations that can be exploited by unauthorized people and search for possible paths to compromise systems.

## 1.2 Objectives.

### 1.2.1 General Objective

To design and develop an automated vulnerability scanner to detect and report the vulnerabilities in both web based and network systems

### 1.2.2 Specific Objectives

1. To study the already existing automated vulnerability scanners
2. To design and develop an automated vulnerability scanner
3. To implement an automated vulnerability scanner.
4. To evaluate an automated vulnerability scanner according to the results of the scanner .

### 1.2.3 Research Question

1. What are the examples of already existing automated vulnerability scanners?
2. How can an automated vulnerability scanner be designed and developed?
3. How can an automated vulnerability scanner be implemented?
4. How can an automated vulnerability scanner be evaluated?

## 1.3. Scope of the Study

### 1.3.0 Content scope

The software will scan the full web application and its focus will be on finding vulnerabilities on parameter-based URLs, IP addresses and local systems and henceforth show information about vulnerability and its severity giving the solution to fix it.

### 1.3.1 Time scope

This project is estimated to take a period of 4 months of which they will include requirement gathering, system development, user interface design, testing and deployment.

## 1.4 Significance of the Study

1. **Business owners/Managers:** The project will help business owners and managers identify security flaws in their systems and take necessary actions to mitigate them. This can prevent potential data breaches, system downtime, and financial losses.

1. **IT administrators:** IT administrators can use vulnerability scanners to identify security risks and patch vulnerabilities. This can help them proactively manage the security of their systems and reduce the likelihood of a security incident.

1. **Security teams:** The project can be an essential tool for security teams. They can use the information gathered from the scanner to prioritize their efforts and respond to vulnerabilities that pose the most significant risk to the organization.

1. **Compliance auditors:** The project will help compliance auditors assess the security posture of an organization. This can be useful in ensuring that the organization meets the required security standards and regulations.

1. **Customers:** Customers of an organization can benefit from the use of vulnerability scanners as they can be assured that their data is being protected. The use of vulnerability scanners can enhance the trust and reputation of the organization among its customers. Overall, the use of vulnerability scanners can benefit stakeholders by reducing the risk of security incidents, improving the security posture of the organization, and enhancing the reputation and trust of the organization among its customers.

# **CHAPTER TWO**

# **LITERATURE REVIEW**

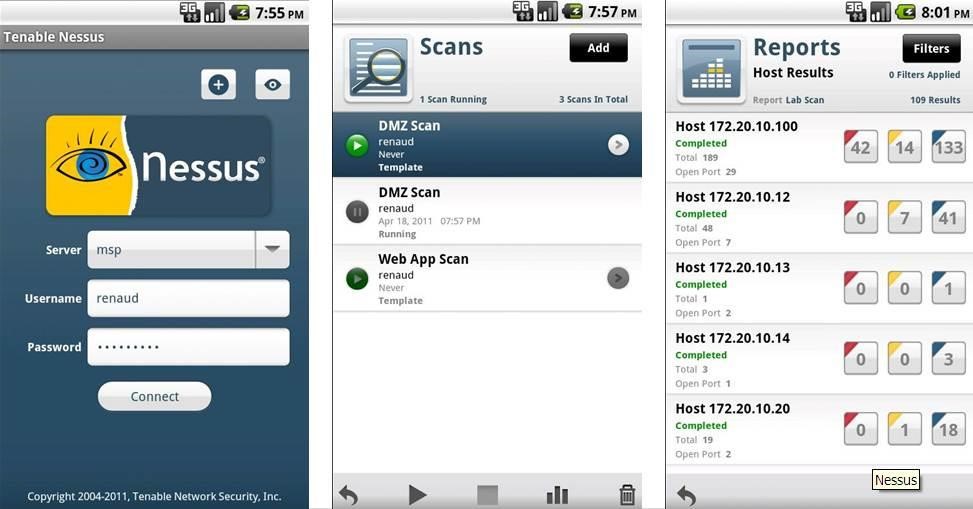
## 2.0 Introduction

Chapter two of this proposal covers the related studies which is the earlier research by different researchers in the area of designing and developing automated vulnerability scanners and the gapd identified in their research.

## 2.1 Related studies

### 2.2.1 Nessus (Columbia, Maryland)

This Nessus scanner is one of the popular vulnerability scanning tools that is created and patented by Tenable Network security.



Nessus uses a database of known vulnerabilities and checks for their presence in the scanned systems. It also provides detailed reports and remediation recommendations. Nessus is available as both a commercial and free open-source version. (Tenable, n.d.)

Nessus' vulnerability scanner can detect various types of vulnerabilities, including software flaws, configuration issues, and compliance violations. It uses various techniques, such as remote and local vulnerability scanning, to identify vulnerabilities in networks and systems. The tool also provides detailed reports and remediation recommendations. (Tenable, n.d.)

In a study by Li and Li (2020), Nessus was evaluated for its performance and effectiveness in detecting vulnerabilities. The study found that Nessus was effective in detecting vulnerabilities in networks and systems, especially for software flaws and configuration issues. However, the study also found that Nessus had limitations in detecting certain types of vulnerabilities, such as zeroday vulnerabilities and advanced persistent threats. (Li & Li, 2020)

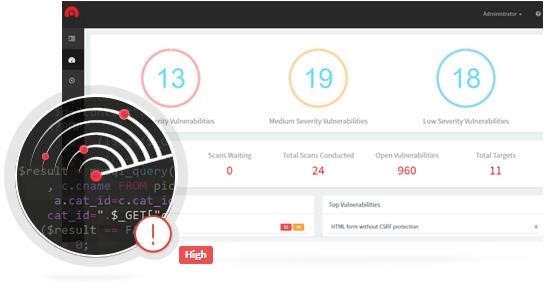
In a comparative study by Rai et al. (2020), Nessus was compared with other popular vulnerability scanners, such as OpenVAS and Qualys. The study found that Nessus had better performance and effectiveness in detecting vulnerabilities, especially for software flaws and configuration issues. The study also found that Nessus provided more detailed reports and remediation recommendations than other vulnerability scanners. (Rai et al., 2020)

In a case study by Santos et al. (2019), Nessus was used to identify vulnerabilities in a network for a real-world client. The study found that Nessus was effective in identifying vulnerabilities, such as software flaws and configuration issues, and provided detailed reports and remediation recommendations. The study also found that Nessus' ability to perform continuous monitoring and automate remediation was useful in maintaining the security of the network. (Santos et al., 2019)

### 2.2.2 Acunetix (Malta)

Acunetix is a fully automated, paid web application vulnerability assessment tool. It is capable of detecting and reporting vulnerabilities on over 4500 web applications.

Acunetix is a web application vulnerability scanner that can scan various web applications for vulnerabilities. It uses a combination of black-box and gray-box testing to identify vulnerabilities. Acunetix provides detailed reports and remediation recommendations, and it can be customized to meet specific needs. Acunetix is available as both a cloud-based and on-premises solution. (Acunetix, n.d.)



Acunetix web application vulnerability scanner has various features and capabilities, such as crawl and attack engine, which help in identifying vulnerabilities in web applications. The tool can detect various types of vulnerabilities, including XSS, SQL injection, and file inclusion vulnerabilities. It also provides detailed reports and remediation recommendations. (Acunetix, n.d.)

In a study by Li and Li (2020), Acunetix was evaluated for its performance and effectiveness in detecting vulnerabilities in web applications. The study found that Acunetix was effective in detecting vulnerabilities in web applications, especially for SQL injection and XSS. The study also found that Acunetix had limitations in detecting certain types of vulnerabilities, such as XML external entity (XXE) and server-side request forgery (SSRF). (Li & Li, 2020)

In a comparative study by Chakrabarty et al. (2018), Acunetix was compared with other popular web application vulnerability scanners, such as Burp Suite and AppSpider. The study found that Acunetix had better performance and effectiveness in detecting vulnerabilities in web applications. The study also found that Acunetix provided more detailed reports and remediation recommendations than other web application vulnerability scanners. (Chakrabarty et al., 2018)

In a study by Kaur and Singh (2018), Acunetix was evaluated for its performance and effectiveness in detecting vulnerabilities in web applications. The study found that Acunetix was effective in detecting vulnerabilities in web applications, especially for SQL injection and XSS vulnerabilities. However, the study also found that Acunetix had limitations in detecting certain types of vulnerabilities, such as server-side request forgery (SSRF) and remote code execution (RCE) vulnerabilities. (Kaur & Singh, 2018)

In a case study by Almehmadi et al. (2018), Acunetix was used to identify vulnerabilities in a web application for a real-world client. The study found that Acunetix was effective in identifying vulnerabilities, such as SQL injection and XSS vulnerabilities, and provided detailed reports and remediation recommendations. The study also found that Acunetix's ability to perform continuous monitoring and automate remediation was useful in maintaining the security of the web application. (Almehmadi et al., 2018)

### 2.2.3 Rapid7 scanner (USA)

Rapid7 is a popular vulnerability scanner that is used to identify vulnerabilities in web applications, networks, and cloud environments. The tool provides various features and capabilities, such as vulnerability scanning, malware detection, and threat intelligence



Rapid7's vulnerability scanner can detect various types of vulnerabilities in web applications, networks, and cloud environments, including SQL injection, XSS, and cross-site request forgery (CSRF) vulnerabilities. It uses various techniques, such as scanning and credentialed testing, to identify vulnerabilities. The tool also provides detailed reports and remediation recommendations, as well as integrations with other security tools. (Rapid7, n.d.)

In a study by Karim et al. (2020), Rapid7 was evaluated for its performance and effectiveness in detecting vulnerabilities in web applications. The study found that Rapid7 was effective in detecting vulnerabilities in web applications, especially for SQL injection and XSS vulnerabilities. The study also found that Rapid7 had a low rate of false positives and provided detailed reports and remediation recommendations. However, the study also found that Rapid7 had limitations in detecting certain types of vulnerabilities, such as server-side template injection (SSTI) and business logic vulnerabilities. (Karim et al., 2020)

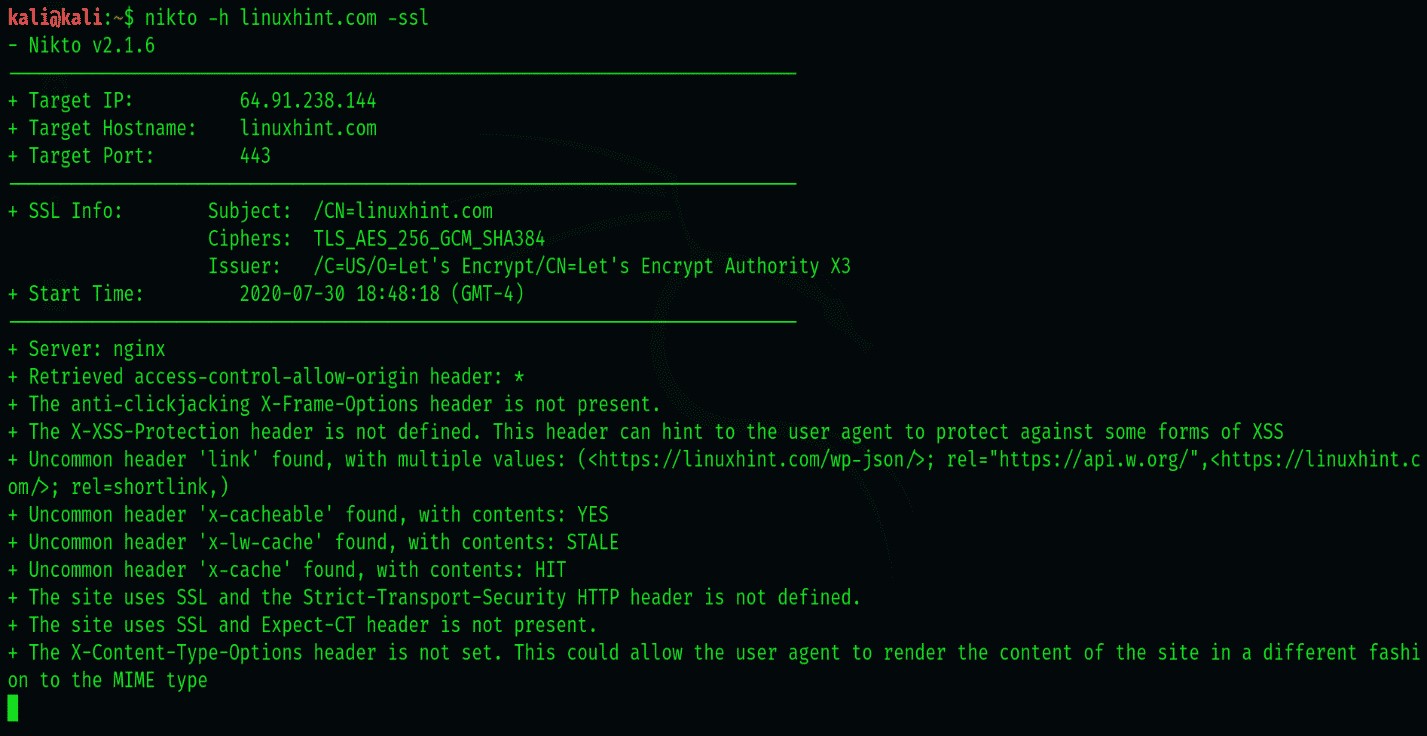
In a comparative study by Sheikh et al. (2021), Rapid7 was compared with other popular vulnerability scanners, such as Burp Suite and OWASP ZAP. The study found that Rapid7 had better performance and effectiveness in detecting vulnerabilities in web applications, networks, and cloud environments. The study also found that Rapid7 provided more detailed reports and remediation recommendations than other vulnerability scanners. (Sheikh et al., 2021)

In a case study by Khawaja et al. (2019), Rapid7 was used to identify vulnerabilities in a cloud environment for a real-world client. The study found that Rapid7 was effective in identifying vulnerabilities, such as misconfigured security groups and outdated software, and provided detailed reports and remediation recommendations. The study also found that Rapid7's ability to integrate with other security tools, such as AWS Security Hub and Splunk, was useful in managing and monitoring the security of the cloud environment. (Khawaja et al., 2019)

### 2.2.4 Nikto (USA)

Nikto is an open-source web server scanner that is designed to identify vulnerabilities in web servers and web applications. The tool provides various features and capabilities, such as web server fingerprinting, vulnerability scanning, and SSL testing.

Nikto can detect various types of vulnerabilities in web servers and web applications, including outdated software, configuration errors, and known vulnerabilities in web server software. It uses various techniques, such as HTTP requests and banner grabbing, to identify vulnerabilities in web servers and web applications. The tool also provides detailed reports and remediation recommendations. (CIRT, n.d.)



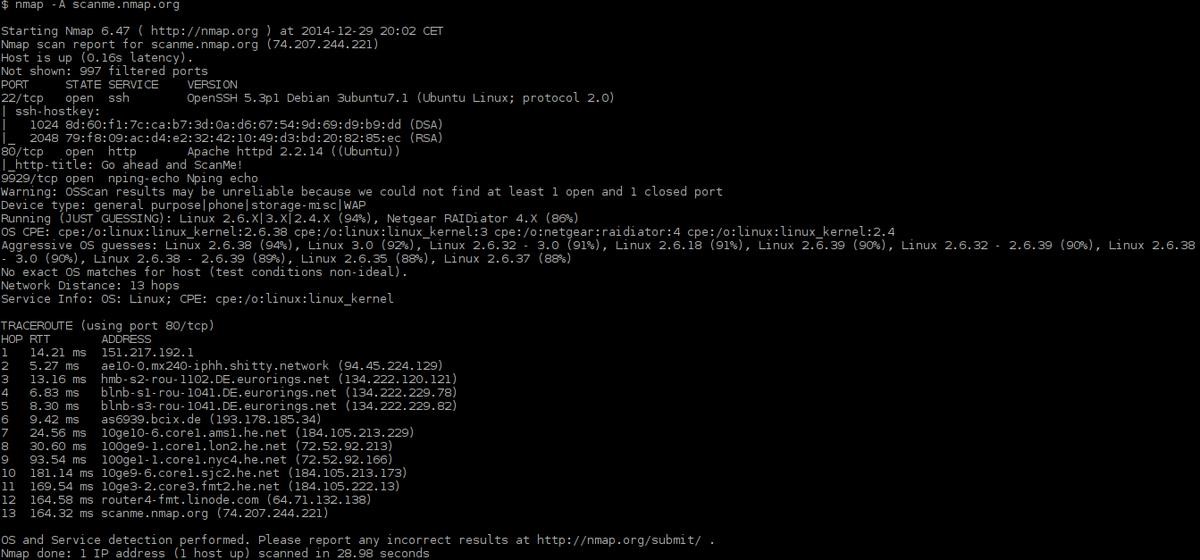
In a study by Zaman et al. (2017), Nikto was evaluated for its performance and effectiveness in detecting vulnerabilities in web servers and web applications. The study found that Nikto was effective in detecting vulnerabilities in web servers and web applications, especially for outdated software and configuration errors. The study also found that Nikto had limitations in detecting certain types of vulnerabilities, such as XSS and SQL injection vulnerabilities. (Zaman et al., 2017)

In a comparative study by Ahmad et al. (2018), Nikto was compared with other popular vulnerability scanners, such as Nmap and OpenVAS. The study found that Nikto had similar performance and effectiveness in detecting vulnerabilities in web servers and web applications, especially for outdated software and configuration errors. The study also found that Nikto provided more detailed reports and remediation recommendations than other vulnerability scanners. (Ahmad et al., 2018)

In a case study by Hossain et al. (2019), Nikto was used to identify vulnerabilities in a web application for a real-world client. The study found that Nikto was effective in identifying vulnerabilities, such as outdated software and configuration errors, and provided detailed reports and remediation recommendations. The study also found that Nikto's ability to perform quick and accurate scans was useful in maintaining the security of the web application. (Hossain et al., 2019)

### 2.2.5 NMAP (USA)

Nmap is a popular open-source command-line network exploration and security auditing tool that can be used to identify vulnerabilities in systems and networks. The tool provides various features and capabilities, such as host discovery, port scanning, and vulnerability scanning.



Nmap can detect various types of vulnerabilities in systems and networks, including open ports, service versions, and known vulnerabilities in software. It uses various techniques, such as TCP/IP fingerprinting and OS detection, to identify vulnerabilities in systems and networks. The tool also provides detailed reports and remediation recommendations. (Nmap, n.d.)

In a study by Yadav et al. (2020), Nmap was evaluated for its performance and effectiveness in detecting vulnerabilities in systems and networks. The study found that Nmap was effective in detecting vulnerabilities in systems and networks, especially for open ports and service versions.

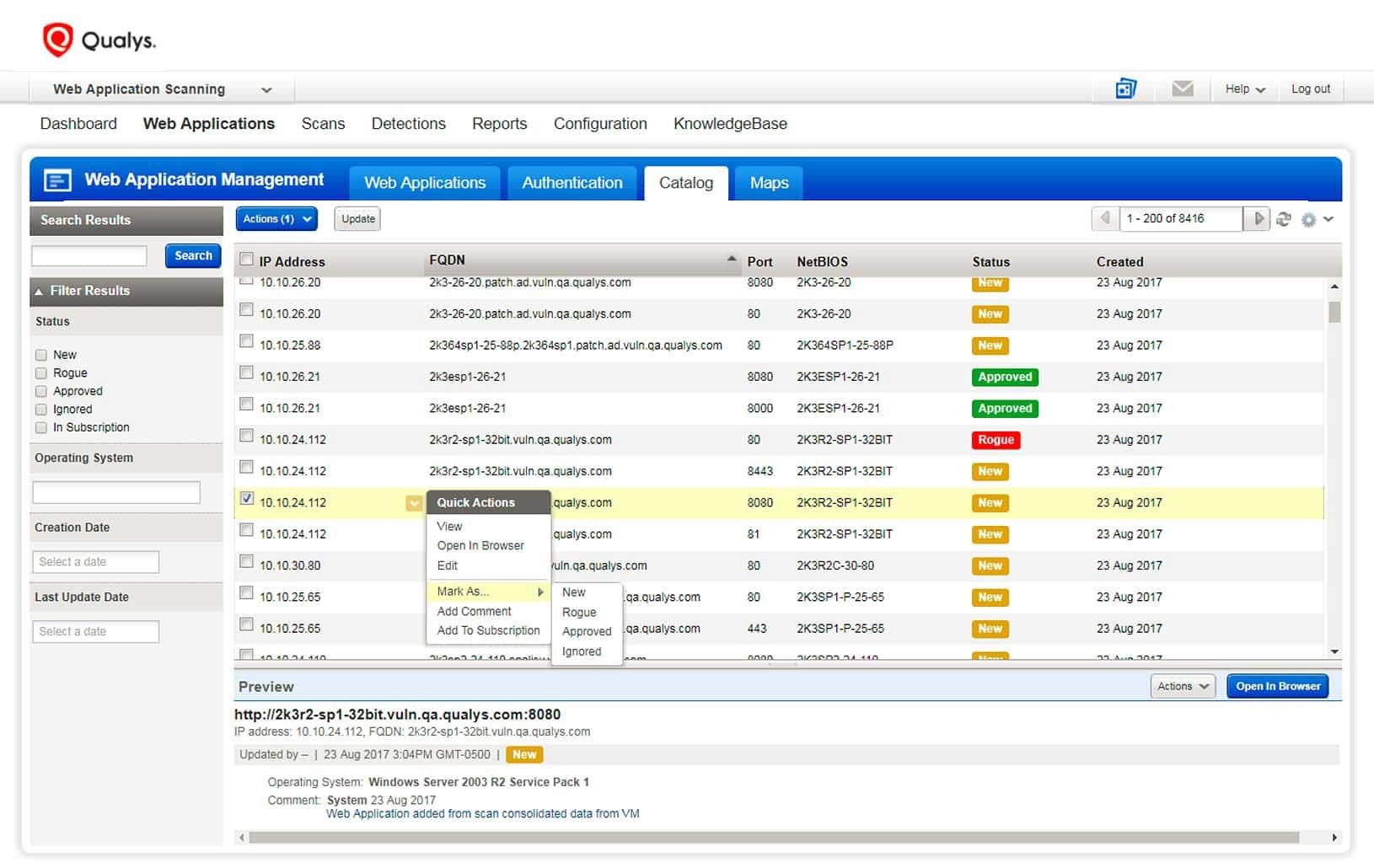
The study also found that Nmap had limitations in detecting certain types of vulnerabilities, such as zero-day vulnerabilities and sophisticated attacks. (Yadav et al., 2020)

In a comparative study by Ahmad et al. (2018), Nmap was compared with other popular vulnerability scanners, such as Nikto and OpenVAS. The study found that Nmap had similar performance and effectiveness in detecting vulnerabilities in systems and networks, especially for open ports and service versions. The study also found that Nmap provided more detailed reports and remediation recommendations than other vulnerability scanners. (Ahmad et al., 2018)

In a case study by Uzair et al. (2020), Nmap was used to identify vulnerabilities in a network for a real-world client. The study found that Nmap was effective in identifying vulnerabilities, such as open ports and service versions, and provided detailed reports and remediation recommendations. The study also found that Nmap's ability to perform quick and accurate scans was useful in maintaining the security of the network. (Uzair et al., 2020)

**Qualys (USA)**

Qualys is a cloud-based vulnerability scanner that can scan web applications, cloud infrastructure, and network devices. It provides real-time alerts and remediation recommendations. Qualys also integrates with other security tools, such as SIEM and GRC, to provide a holistic view of an organization's security posture. (Qualys, n.d.)



Qualys can detect various types of vulnerabilities in network and web-based applications, including SQL injection, XSS, and file inclusion vulnerabilities. It uses various techniques, such as scanning and penetration testing, to identify vulnerabilities in network and web-based applications. The tool also provides detailed reports and remediation recommendations. (Qualys, n.d.)

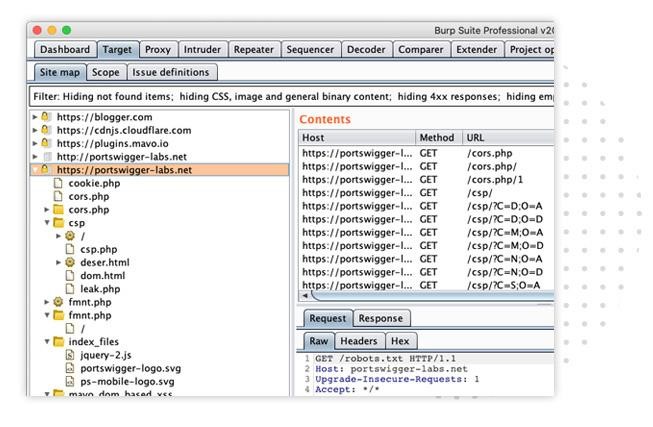
In a study by Wang et al. (2018), Qualys was evaluated for its performance and effectiveness in detecting vulnerabilities in network and web-based applications. The study found that Qualys was effective in detecting vulnerabilities in network and web-based applications, especially for SQL injection and XSS vulnerabilities. The study also found that Qualys had limitations in detecting certain types of vulnerabilities, such as server-side request forgery (SSRF) and remote code execution (RCE) vulnerabilities. (Wang et al., 2018)

In a comparative study by Shen et al. (2018), Qualys was compared with other popular vulnerability scanners, such as Nessus and OpenVAS. The study found that Qualys had similar performance and effectiveness in detecting vulnerabilities in network and web-based applications, especially for SQL injection and XSS vulnerabilities. The study also found that Qualys provided more detailed reports and remediation recommendations than other vulnerability scanners. (Shen et al., 2018)

In a case study by Sharma et al. (2020), Qualys was used to identify vulnerabilities in a network for a real-world client. The study found that Qualys was effective in identifying vulnerabilities, such as SQL injection and XSS vulnerabilities, and provided detailed reports and remediation recommendations. The study also found that Qualys' ability to perform continuous monitoring and automate remediation was useful in maintaining the security of the network. (Sharma et al., 2020)

**Burp Suite (UK)**

Burp Suite is a web application security testing tool that includes an automated vulnerability scanner. It can scan web applications for vulnerabilities, including SQL injection and cross-site scripting. Burp Suite also includes other security testing tools, such as a web proxy and a vulnerability scanner. Burp Suite is available as both a free and paid version. (PortSwigger, n.d.)



In a study by Chen et al. (2019), Burp Suite's vulnerability scanner was evaluated for its performance and effectiveness in detecting vulnerabilities. The study found that Burp Suite was effective in detecting vulnerabilities in web applications, especially for SQL injection and XSS. However, the study also found that the tool had limitations in detecting certain types of vulnerabilities, such as DOM-based XSS and CSRF. (Chen et al., 2019)

In a comparative study by Li and Li (2020), Burp Suite's vulnerability scanner was compared with other popular web application vulnerability scanners, such as Acunetix and Nessus. The study found that Burp Suite had better performance in detecting vulnerabilities, especially for SQL injection and XSS. The study also found that Burp Suite provided more detailed reports and remediation recommendations than other vulnerability scanners. (Li & Li, 2020)

In a case study by Torrecillas et al. (2020), Burp Suite's vulnerability scanner was used to identify vulnerabilities in a web application for a real-world client. The study found that Burp Suite was effective in identifying vulnerabilities, such as SQL injection and XSS, and provided detailed reports and remediation recommendations. The study also found that Burp Suite's ability to customize scans and perform manual testing was useful in identifying vulnerabilities that were not detected by automated scanning. (Torrecillas et al., 2020)

## 2.4 Gaps identified

There exist a large number of vulnerability detection and security assessment tools. Most of these tools (like Nikto or Nessus) rely on a repository of known vulnerabilities that are tested. This is in contrast to our automated vulnerability scanner, which will be focused on the identification of a broad range of general application-level vulnerabilities. In addition to application level vulnerability scanners, there are also tools that audit hosts on the network level. For example, tools such as NMap or Xprobe can determine the availability of hosts and accessible services. However, they are not concerned with higher-level vulnerability analysis.

**Integration with other tools:** Available vulnerability scanners may not be able to integrate with other security tools, which can be limiting for security teams who need to use multiple tools to manage their security posture. Our intended automated vulnerability scanner tool should be designed to integrate with other security tools to provide a more comprehensive security solution.

**Cost:** Some vulnerability scanners may have a high licensing cost or require a significant investment in hardware, which can be limiting for small businesses or organizations with limited resources. Our automated vulnerability scanner will be designed to be cost-effective and accessible to a wide range of users.

There are commercial web application vulnerability scanner available on the market that claim to provide functionality similar to our automated vulnerability scanner (like Acunetix Web Vulnerability Scanner). Unfortunately, due to the closed-source nature of these systems, many of the claims cannot be verified, and an in-depth comparison with our tool will be difficult. For example, it appears that the cross-site scripting analysis performed by Acunetix is much simpler than the complete attack we intend to develop with this automated tool. Also, no working proof of-concept exploits are generated.

Customization: Most available vulnerability scanners are not customizable, which can limit their usefulness for specific environments or use cases. Our automated vulnerability scanner will be designed to be customizable and flexible, allowing users to tailor the tool to their specific needs and requirements since we intend to make it open source.

Report Generation. Overall, report generation based on severity is a key aspect of vulnerability scanning we intend to incorporate into our scanner as it helps organizations to effectively manage their security risks which many available tools have not incorporated into their available tools. By using severity ratings to prioritize and guide remediation efforts, organizations can improve their overall security posture and reduce the likelihood of a successful cyber-attack. (Source: National Institute of Standards and Technology. "Guide to Vulnerability Management." https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-40r3.pdf)

Huang et al. presented a vulnerability detection tool that automatically executed SQL injection attacks. As far as SQL injection is concerned, our project is similar to theirs. However, their scanner is not as comprehensive as our tool because it lacks any detection mechanisms for XSS vulnerabilities where script code is injected into applications. The focus of their work, rather, is the detection of application level vulnerabilities that may allow the attacker to invoke operating level system calls (such as opening a file) for malicious purposes (Huang, 2014).

Limited scope. The earlier vulnerability scanners had limited scope and only identified vulnerabilities that are known to the scanner’s database, leaving unknown vulnerabilities un identified. According to a report by Forrester, “vulnerability scanners may not find every vulnerability, especially those that are not known or publicized.” (source: Forrester, “ the Forrester Wave: vulnerability risk management, Q1 2020”).

# **CHAPTER THREE**

# **RESEARCH METHODOLOGY**

## 3.0 Introduction

The methodology section provides a comprehensive overview of the research approach, target population, sampling techniques, system design and interface, testing, Deliverables and Stakeholders of an automated vulnerability scanner tool project.

## 3.1 Research Approach

The research approach adopted for this study is a case study approach. The case study approach was chosen, as it is suitable for mitigating the risk of a security incident and avoid the cost of cyber-attacks so that we can be able to prevent, detect, respond and recover from such attacks. From this approach we shall prevent many attacks by making sure we remediate all known software vulnerabilities and performing regular security assessments to identify possible unknown vulnerabilities. This approach will allow for the collection of both qualitative and quantitative data that will provide a complete picture of the situation.

## 3.2 Target Population

The target population of an automated vulnerability scanner project is estimated to reach up to 45,400 people which will typically include organizations that develop or manage web applications, such as businesses, government agencies, or educational institutions. These organizations are vulnerable to cyber-attacks that exploit vulnerabilities in their web applications, and an automated vulnerability scanner can help to identify and remediate these vulnerabilities before they are exploited by attackers.

The specific target population may vary based on the type of web applications being developed or managed. For example, an automated vulnerability scanner designed for small and medium-sized businesses may target organizations that use common web application platforms like WordPress1, while a scanner designed for large enterprises may target organizations with more complex web applications that require more customization and flexibility.

The target population may also include information security professionals, such as penetration testers or security analysts, who are responsible for managing the security of web applications. These professionals can use the vulnerability scanner to identify and prioritize vulnerabilities, and to provide recommendations for remediation.

Overall, the target population of an automated vulnerability scanner project is any organization or individual who is responsible for managing the security of web applications and who needs an effective tool for identifying and remediating vulnerabilities.

## 3.3 Sampling technique

**Random sampling:** This technique involves scanning a random selection of hosts, ports, services, or applications on a network. The goal is to provide a representative sample of the overall system without biasing the results towards any particular part of the system.

### 3.3.1 Sample size.

The sample size will be estimated using the formula (Slovene's formula).

n = N where n is the sample size; N is the sample population; e is the marginal error which is constantly 0.05

The sample size used in a vulnerability scanner refers to the number of targets or assets that are being scanned for vulnerabilities within a given population or network. The sample size can vary depending on the size and complexity of the network, the type of vulnerability being scanned for, and the resources available for the scan.

## 3.4 System Development Methodologies

Designing a vulnerability scanner involves several key components and considerations, including the scanning engine, the vulnerability database, the user interface, and the reporting functionality.

Here are some system design and interface techniques for a vulnerability scanner:

**Requirements gathering and analysis**: we intend to gather detailed information about the scope and requirements of the project, including the types of systems and networks that will be scanned, the types of vulnerabilities that need to be detected, and any specific features or functionality that are required.

**Research and development:** we shall conduct research on the latest vulnerabilities, attack vectors, and security best practices, and develop the necessary algorithms and techniques for detecting them.

**Design and architecture:** Design the overall architecture of the scanner, including the algorithms, data structures, and command line interface that will be used.

**Implementation**: Develop the scanner using a suitable programming language and framework, and test it on a variety of systems and networks to ensure it is working as expected.

**Integration and testing**: Integrate the scanner with existing security tools, such as firewalls, intrusion detection systems, and endpoint protection software, and conduct thorough testing to ensure it is working as intended.

**Deployment and maintenance:** Develop a user-friendly interface for the scanner, and deploy it in a production environment. Regularly maintain and update the scanner to ensure it stays current with the latest vulnerabilities and security threats.

The Agile methodology is a suitable approach for this project as it allows for flexibility and adaptability as the project progresses. Furthermore, regular testing and feedback will ensure that the final product meets the customer requirements.

## 3.5 Functional requirements

Functional requirements are those requirements that are used to illustrate the internal working nature of the system, the description of the system, and explanation of each subsystem. It consists of what task the system should perform, the processes involved, which data should the system holds and the interfaces with the user. The functional requirements identified in the vulnerability scanner must have, such as;

**Discovery:** The vulnerability scanner should be able to identify and locate all the devices, systems, and software applications within the network that require security assessment.

**Vulnerability Assessment:** The scanner should be capable of analyzing the vulnerabilities present in the target systems, networks, and applications.

**Risk Analysis:** The vulnerability scanner should provide a comprehensive analysis of the risks associated with identified vulnerabilities, including the potential impact on system security and business operations.

**Reporting:** The scanner should generate comprehensive reports that highlight the vulnerabilities, their potential impact, and recommendations for remediation.

**Remediation Management:** The vulnerability scanner should provide guidance on how to remediate the identified vulnerabilities, including patches, upgrades, and configuration changes. **Integration:** The vulnerability scanner should be able to integrate with other security tools to provide a complete and holistic security solution.

**Compliance:** The scanner should comply with relevant industry standards, regulations, and guidelines, such as PCI DSS, HIPAA, and ISO 27001

### 3.5.1 Other Requirements

The requirements of this project can be broadly categorized into the following areas:

**3.5.2 Technical requirements:**

These are the specific technical requirements that the scanner must meet, such as the programming language and framework to be used, the algorithms and techniques to be used for vulnerability detection, and the ability to integrate with other security tools.

**3.5.3 User interface requirements.**

These are the specific requirements for the user interface of the scanner, such as ease of use, navigation, and design.

**3.5.4 Performance requirements.**

These are the specific performance requirements that the scanner must meet, such as the speed of scanning, the number of vulnerabilities that can be detected in a given time period, and the accuracy of the scanner.

**3.5.5 Security requirements.**

These are the specific security requirements that the scanner must meet, such as data encryption, user authentication, and access controls.

## 3.6 Data collection methods.

### 3.6.1 Observation

This technique will be used to gain awareness of the overall activities in the country as well as the identifying any potential issues and activity inputs to come up with an objective analysis.

### 3.6.2 Interview

Researchers will use interview because it helps them to get a better response rate than mailed questions. The researchers will be able to engage administrators of organizations who work in all departments in order to collect vast data which will be used develop the system.

Interviews will be used by researchers based on the feedback of the respondent and they are more personal than self-directed questionnaires. Researchers will use in-depth interview, which will be a dialogue between the interviewer and the respondents who included the staffs, trainees and different customers.

Open-ended questions and extensive investigation characterized the in-depth interviews. Researchers followed an interview guide that will include a list of questions to be discovered that will speed up the interview and made it systematic.

In-depth interviews will be useful where the interview will respond about a highly sensitive matter and the respondents will feel more comfortable with it than with a questionnaire as group discussion. It will also be useful when the subject matter will be complex and where the researchers needs detailed information.

### 3.6.3 Questionnaires.

Sekaran and Bougie, (2013) defined a questionnaire as a pre-formulated written set of questions to which respondents record their answers. In this study, the researcher will develop the questionnaire in a way that it is composed of both open and close-ended questions. For close-ended questions the researcher will be asking the respondents to make choices from the alternatives provided. Open-ended questions aimed at allowing the respondents to answer them in the way they choose.

Questionnaires for a vulnerability scanner:

1. Which kind of security bleaches have you ever faced?
2. Which kind of vulnerabilities should the scanner be able to discover?
3. How should the vulnerability report be presented?
4. Which other security tools should the scanner be integrated with?
5. Can you be able to decrypt encrypted data?

|  |
| --- |
|  |

Yes

|  |
| --- |
|  |

No

1. To what do you think the scanner will help your organization / business

### 3.6.4 Documentary review

Some of the information shall be collected by studying documents from within and out of the study area. Information from the Literature review shall bring out the need for a computerized system to serve as an expense tracker and how critical information circulation should be in such an environment as a market. However, most of the records could be printed on paper and making it hard to be retrieved as they could have been lost already. This indicates the need for a system tracking expenses and managing records that could keep and retrieve such information on demand. The documentary review helps the researchers to perceive the views of people elsewhere in the world towards Criminal Systems. The researcher’s findings shall clearly that there is a need for a system for crime management and reporting.

## 3.7 System Design and Interface Techniques

**Modular design:** A modular design approach can be used to break the scanner system down into smaller, more manageable components that can be developed and tested independently. This approach can make the scanner more reliable and easier to maintain.

**Scalability:** The scanner system should be designed to be scalable, so that it can handle larger and more complex networks as needed. This can be achieved through the use of distributed architectures, load balancing, and other techniques.

**User-friendly interfaces:** The scanner should have an intuitive and user-friendly interface that is easy for users to navigate and understand. This can include graphical user interfaces (GUIs), dashboards, and other visualization tools.

**Customization:** The scanner should allow users to customize the scanning parameters and settings to meet their specific needs. This can include the ability to configure scanning schedules, select specific targets or assets to scan, and adjust the severity thresholds for identified vulnerabilities. Integration with other tools: The scanner should be designed to integrate with other security tools and technologies, such as intrusion detection systems (IDS), security information and event management (SIEM) systems, and vulnerability management platforms.

**Reporting and analytics**: The scanner should be capable of generating detailed reports on the vulnerabilities it has identified, including the severity of the vulnerabilities and recommended remediation steps. The system should also provide analytics to help users identify trends and patterns in vulnerability data over time.

## 3.8 Testing

Testing a vulnerability scanner is an important step in ensuring that it is functioning properly and accurately identifying vulnerabilities in a system. Here are some common methods for testing a vulnerability scanner:

**Penetration testing:** A penetration testing (pen-testing) team can be engaged to simulate real-world attacks against the system and assess the effectiveness of the vulnerability scanner in identifying vulnerabilities. This can help identify any blind spots or weaknesses in the scanner's coverage.

**Test data sets:** Test data sets can be created to simulate different types of vulnerabilities in the system. These data sets can be used to verify that the scanner is identifying the vulnerabilities correctly.

### 3.8.1 Tools for Implementation

There are many different tools that can be used in the implementation of a vulnerability scanner project. Here are some common tools:

**Nmap:** Nmap is a popular open-source tool used for network mapping and port scanning. It can be used to discover hosts and services on a network, and to identify potential vulnerabilities associated with those services.

**Nessus:** Nessus is a commercial vulnerability scanner that can be used to identify vulnerabilities in both network and web applications. It has a large database of known vulnerabilities and can provide detailed reports on the vulnerabilities it identifies.

**OpenVAS:** OpenVAS is an open-source vulnerability scanner that is similar to Nessus. It can be used to identify vulnerabilities in both network and web applications, and can provide detailed reports on the vulnerabilities it identifies.

### 3.8.2 Deliverables

Functioning vulnerability scanner software

Technical documentation on the scanner's capabilities and limitations

User manual for the scanner

Training and support for the scanner's deployment and use

Regular updates to the scanner to ensure it stay current with the latest vulnerabilities and security threats.

### 3.8.3 Stakeholders

The stakeholders of the vulnerability scanner project can include:

**The project sponsor or client**: The organization or individual who is funding and/or requesting the project.

**Project manager:** The person responsible for overseeing the project and ensuring it is completed on time, within budget, and to the satisfaction of the client.

**Development team:** The team responsible for designing, developing, and testing the scanner. **Security professionals:** The individuals who will be using the scanner to identify and address vulnerabilities in their systems, such as IT security staff, network administrators, and security engineers.

**Regulators and Auditors:** the individuals or organizations that are responsible for ensuring that the systems and networks are compliant with industry regulations and standards.

**Management:** The individuals who are responsible for making strategic decisions and allocating resources for the organization, they will be affected by the results of the project.

## 3.9 Conclusion

The development of a comprehensive vulnerability scanner is a crucial step in maintaining the security of modern systems and networks. This

project aims to develop a scanner that can effectively identify a wide range of vulnerabilities, misconfigurations and weak passwords.

With the ability to integrate with other security tools, the scanner will provide a comprehensive security solution.

With a user-friendly interface, the scanner will be easy to deploy and use by security professionals, helping to keep

# **CHAPTER FOUR**

# **SYSTEMS ANALYSIS AND DESIGN**

## 4.0 Introduction.

In this chapter, we discussed the weakness and strength of the current system, analysis and detailed

design issues, design requirements and system functionalities. We further addressed the conditions

that were necessary for the effective functioning of the system and also the tools that were used in

the development and design of the new system.

## 4.1 System study.

We studied the existing scanners to identify their strength and weaknesses. The information that was acquired from this study by employing a number of techniques and tools such as questionnaire and interview guides, gave the basis for the design of our new automated vulnerability scanner. An interview guide with open ended questions was prepared and administered to the respondents who included IT administrators, security teams and business owners so as to enable them give their views freely. This technique was chosen because it helped us to obtain full range and depth, we realized that the available automated scanners were not be able to integrate with other security tools, a high licensing cost or require a significant investment in hardware. Due to these factors systems have remained prone to vulnerabilities and attacks. This has created need for a more reliable and efficient automated vulnerability scanner.

## 4.2 weaknesses of existing scanners.

Available vulnerability scanners may not be able to integrate with other security tools, which can be limiting for security teams who need to use multiple tools to manage their security posture. Our intended automated vulnerability scanner tool should be designed to integrate with other security tools to provide a more comprehensive security solution.

Some vulnerability scanners may have a high licensing cost or require a significant investment in hardware, which can be limiting for small businesses or organizations with limited resources. Our automated vulnerability scanner will be designed to be cost-effective and accessible to a wide range of users.

Most available vulnerability scanners are not customizable, which can limit their usefulness for specific environments or use cases. Our automated vulnerability scanner will be designed to be customizable and flexible, allowing users to tailor the tool to their specific needs and requirements since we intend to make it open source.

Limited scope. The earlier vulnerability scanners had limited scope and only identified vulnerabilities that are known to the scanner’s database, leaving unknown vulnerabilities un identified.

## 4.3 Strengths of existing scanners

The existing scanners offer comprehensive testing for example Nikto is designed to scan wide range of web servers and applications and can identify over 6700 potential dangerous files and programs.

Existing scanners are regularly updated to keep pace with the latest webserver, system and application vulnerabilities ensuring that users are always testing against the most current threats.

The existing scanners are platform independent whereby they are cross platform tools which means that they can be used on a variety of operating systems like windows Linux and macOS.

Existing scanners have active community development whereby there is regular active community of developers and users who contribute to their ongoing development and support.

## 4.4 Requirement specification.

The requirement analysis stage of software engineering is the practice of obtaining, collecting and analyzing information that is used to identify the users' requirement of the new system. The term elicitation is used in research to raise the fact that good requirement cannot just be collected from users as would be indicated by the name requirement engineering. Requirement elicitation is not trivial because you can never be sure that you may get all requirements from users by asking them what the system would do and it includes the process like interview, observation, workshop, brain storming and questionnaires. Identifying the required functionality of the system is very important as a system with the incomplete functionality may lead to it being rejected. A description of the aim of the project is given here along with the details of the functional and non-functional requirements for the system.

### 4.4.1 Functional requirements

This scanner is a tool for information gathering and vulnerability assessment of web applications that can find loopholes in website code. It is an Open Source Intelligence (OSINT)-based tool for reconnaissance that is useful for information gathering about a target website or system IP address. It has modules for Geo lookup, banner grabbing, DNS lookup, and port scanning. This tool integrates multiple security scanning tools, including nmap, dnsrecon, wafw00f, uniscan, sslyze, fierce, wpscan, the harvester, xsser, amass, and nikto, under one entity. The tool is lightweight, not process intensive, and saves time by checking for the same vulnerabilities with multiple tools to identify false positives. It analyzes the severity of the vulnerabilities as critical, high, medium, low, informational and provides definitions, remediation guidance, and executive summaries. The tool generates a detailed, comprehensive report with complete details of the total number of vulnerability checks, vulnerability checks skipped, vulnerabilities detected and the total time taken to accomplish the scan.

### 4.4.2 Nonfunctional requirements.

This scanner that has several non-functional requirements that make it a desirable tool for security professionals. It able to perform efficiently and quickly, even when scanning large web applications or networks. This is important because users need to be able to get results quickly, especially when dealing with large amounts of data. Additionally, the tool should be user-friendly, with clear documentation and an easy-to-use interface. This helps to reduce the learning curve for new users, allowing them to quickly become proficient in the tool's use.

Another important non-functional requirement for this scanner is compatibility. The tool is able to be compatible with a wide range of operating systems and web servers. This is critical because different web servers have different configurations and vulnerabilities that need to be identified. Additionally, the tool should not introduce any new vulnerabilities into the web application being scanned. This is important to maintain the integrity of the application being scanned.

Our scanner is reliable and produce accurate results consistently. The tool is able to handle large web applications and networks without compromising performance or accuracy of the scanned website, Server or system.

Security: Our scanner does not introduce any security vulnerabilities into the network being scanned and is not vulnerable to attacks itself. Security is a critical non-functional requirement for our scanner. Our scanner does not compromise the security of the network being scanned or expose itself to attacks that could compromise its functionality or data. This is achieved security through various techniques such as sandboxing, privilege separation, and secure communication protocols. The tool also provides guidance on how to use it securely and avoid common security pitfalls.

Usability: Our scanner is easy to use and understand, with clear documentation and a user-friendly interface. Usability is an important non-functional requirement for the tool. The tool must be accessible to users of all skill levels and provide clear instructions and guidance on how to use the tool effectively.

## 4.5 System requirements

These are required by the system to be able to perform its expected functionalities efficiently and effectively. The following are required to run the system.

### 4.5.1 Hard ware requirements

A desktop computer with Intel Core i3 64-bit processor and Graphic card I GB RAM, and macOS Elcapitan operating system was used. However, the user of the system may be required to have a personal computer of at least 4GB of RAM with a 200GB hard disk; a system unit with at least IOOGB free hard disk space, IOOGB of RAM but 250GB recommended as well as Intel Pentium Xeon 2200 MHZ FC-PGA Processor.

### 4.5.2 Software requirements

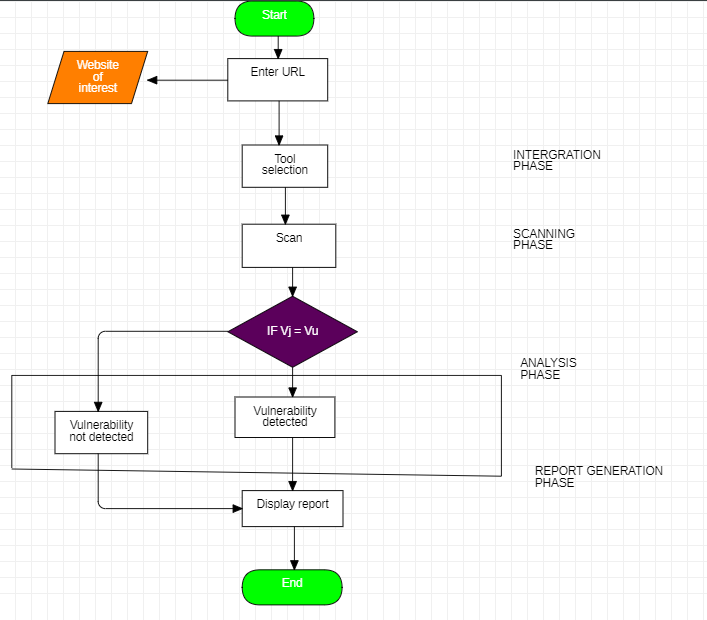
The scanner rans on the principles of a command line based console interpreting and executing given commands. Python was used to develop this scanner because of its robust libraries that help in automating tasks. Debian based Kali Linux installed on the virtual machine or on the host for offensive security because of its advantage of pre-installed tool likr nmap, wfuzz that come pre-installed in it is required since our tool needs to integrate with these other tools.

## 4.6 Structured Requirement Analysis.

### 4.6.1 Flow Chart

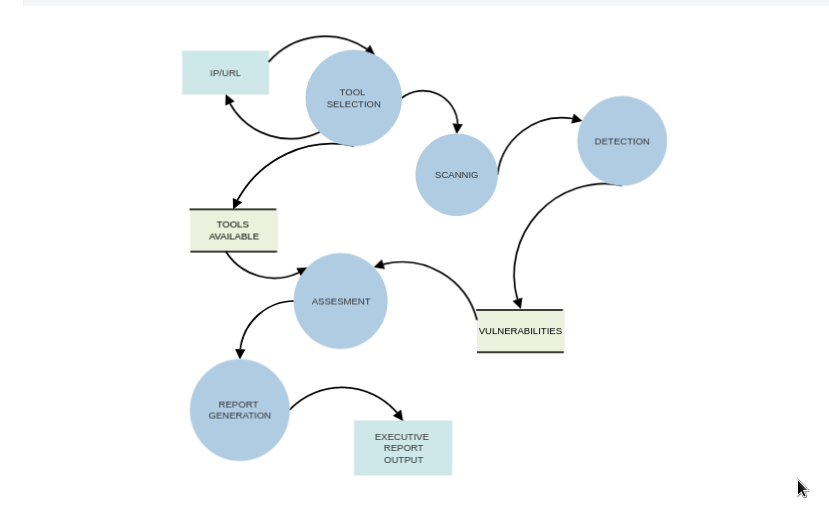
A flowchart is a graphical representation of a process or system that uses symbols, arrows, and text to show the sequence of steps involved in completing a task. It is a visual tool used to illustrate how information or materials move through a process from start to finish.

In a flowchart, different shapes are used to represent different elements of a process, such as a start or end point, decision points, actions, inputs or outputs, and connectors. The shapes are connected by arrows to show the flow and sequence of steps in the process.



### 4.6.2 Data Flow Diagram

A data flow diagram (DFD) or a bubble chart is a graphical tool for structured analysis. It was De Marco and Gane and Sarson who introduced DFD. DFD models a system by using external entities from which data flows to a process, which transforms the data and creates. output data-flows which go to other process or external entities or files. Data in files may also flow to processes as arrows. There are various symbols used in the DFD. Bubbles represent the processes. Named arrows indicate the data flow. External entities are represented by rectangles and are Outside the system such as vendors or customers with whom the system interacts. They either supply or consume data. Entities supplying data are named as sources and those that consume data are called sinks. Data are stored in a data store by a process in the system. Each component in a DFD is labeled with a descriptive name. Process names are further identified with a number. Data flow Diagrams are made up of a number of symbols, which represents system components.



# **CHAPTER FIVE**

# **SYSTEM IMPLEMENTATION, TESTING AND EVALUATION**

## 5.0 Introduction

System analysis and design process including requirement analysis, business solution options, feasibility study, architectural design was discussed in previous chapter. Generally, software bugs cannot be avoided as these always exist in any software module. But it is not because of the  carelessness or irresponsibility of programmer but because of the complexity. Humans have only limited ability to manage complexity. This chapter discusses about the testing of the solution and implementation methodologies.

This chapter basically focused on converting the design procedures and diagrams to an executable code that constructs the necessary interfaces that enhanced user interaction with the system.

## 5.1 Implementation

System implementation projects are long difficult. Journeys by which organizations move from an Old set of technology/methods/procedures to a new one, a software implementation method is a systematic structured approach to effectively integrate software-based service or component into workflow of an organizational structure or an individual end-user. The complexity of implementing product software differs on several issues. Examples are: the number of end users that will use the product software, the effects that the implementation has on changes of tasks and responsibilities for the end user, the culture and the integrity of the organization where the software is going to be used and the budget available. It is vital to select the right strategy for implementing the application to assure successful results.

### 5.1.1 Implementation strategy

The implementation strategy for an automated vulnerability scanner involves defining the scope, selecting the right tool, configuring and testing the scanner, deploying it to the target systems, monitoring and analyzing results, remediating vulnerabilities, generating reports, and continuously improving the process. It is important to prioritize vulnerabilities based on their severity and potential impact, and to repeat the scanning process regularly to ensure new vulnerabilities are identified and addressed. The ultimate goal is to improve the organization's overall security posture.

### 5.1.2 User Guide

User guides have written in plain English rather than technical language. The guide covers how to run, update and install the scanner, how and what commands to enter on the terminal, how to customize the tools to be integrated and how to set parameters and how to use utilities

Further, a list of error messages and advice on what to do if something went wrong are also included in the user guide.

### 5.1.3 Program Deployment

The deployment of this scanner is organized in very simple smarter manner. As standard practice, a bash script setup was created including all the files and requirements including other necessary supporting components. Using this setup script, system can be installed in any client computer very easily. In order to implement modifications, upgrades etc. easily, the scanner allows an update command that is run to get the latest release of the tool or for the latest updates especially vulnerability databases. After installing the setup in the client machine preferably Linux, user has to just give the tool executable permissions so as it can execute with super user privileges.

To effectively use this tool on a beginner level one has to first run help option on the tool so that the guide or manual options are displayed to the user.

The user then runs the tool to check for possible available tools to be integrated with the tool.

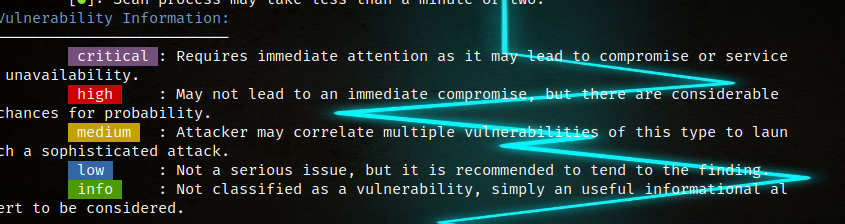
After that an IP address or URL is passed to the scanner and then the scanner is able to scan, detect, assess, mitigate and generate a report that is well detailed in the same folder the user ran the tool.

The report is named in the time and date the scan was completed so as to keep track of the day the scan took place

### 5.1.4 Functional requirements output

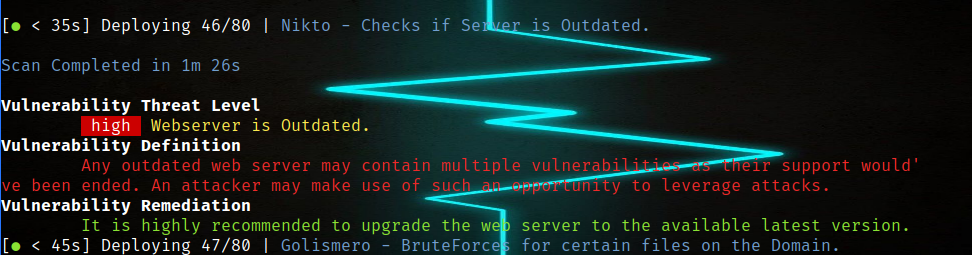
#### 5.1.4.1 Discovery

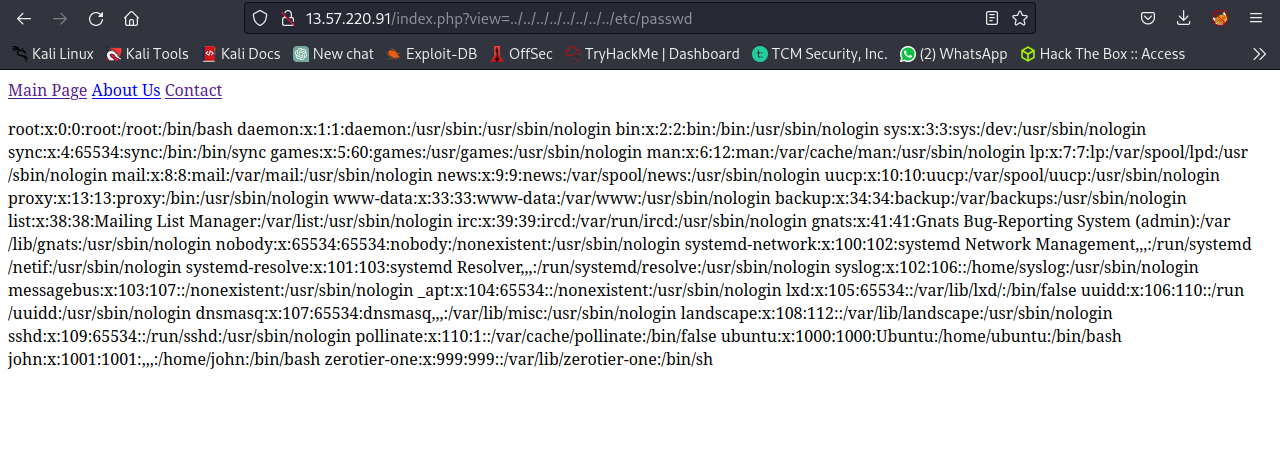
Our scanner was able to discover known vulnerabilities like cross-site scripting, sql injection, open port, remote code execution, server side attacks and so on.



As seen above the tool was able to discover a critical vulnerability which was cross-site scripting attack famously known as **XXS** attack hence our discovery functionality of Discovery was archived.

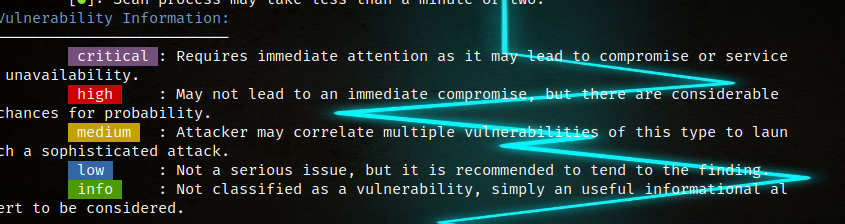
Our scanner used its integration capacity to discover outdated web server as the algorithm compared the scanned system with the current version of the server hence discovering an outdated server which may be prone to attacks.

This vulnerability could lead to exposure of important file like /etc/passwd file that hold all password in on the server including the admin password if an attack exploit a local file inclusion vulnerability as seen in the figure below how we were able to traverse the out dates server to view the password file using local file inclusion discovered by our tool as part of the outdated servers.

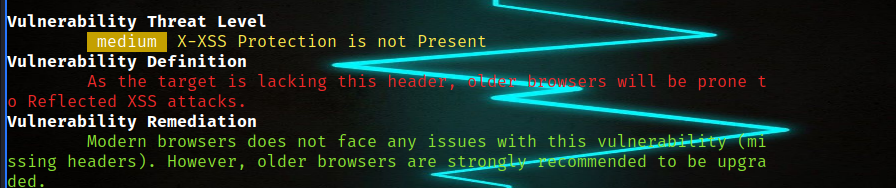


#### **5.1.4.2 Risk Analysis:**

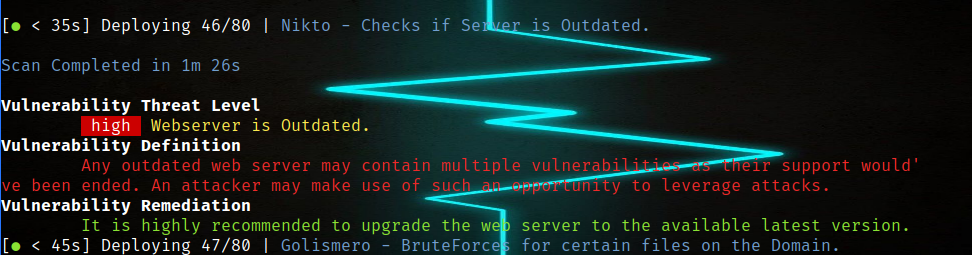
The vulnerability scanner should provide a comprehensive analysis of the risks associated with identified vulnerabilities, including the potential impact on system security and business operations and we archived this where by the scanner was able to detect and assess then provide a risk analysis according to the severity of the risk.

For example the scanner discovered an XXS attack which could be exploited by the malicious actors to infect the system with malware. It analyzed the risk the system was exposed to and printed a blue color coded severity representing a critical vulnerability as programmed. Hence this analysis could prompt an organization to take an immediate action to patch the vulnerability.

### 5.1.5 Vulnerability detection

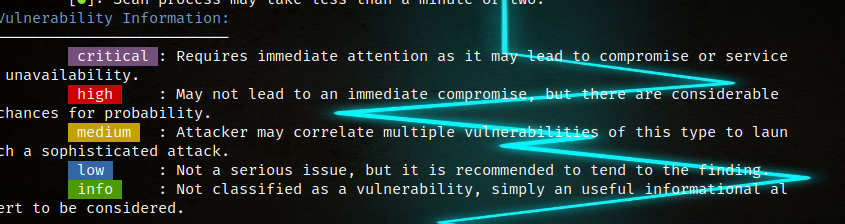
The scanner automatically initiates the preliminary scanning phase by integrating other tools to discover 80 possible known vulnerabilities like SQL injection, local file inclusion, cross site scripting and so on.

As seen above the scanner detected that XSS protection is not present in the scanned environment and the vulnerability is defined for the user to understand it well. The vulnerability is **assessed** and it seem to be medium according to severity, the scanner automatically crawls the web to find the best remediation for the vulnerability discovered. The scanner can also detect outdated systems like webserver that could make systems vulnerable when not updated

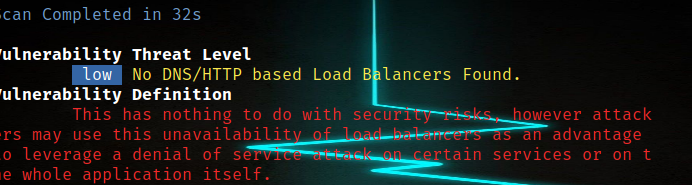


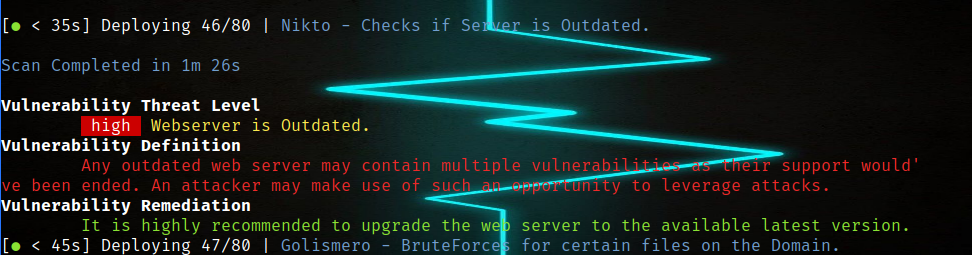
#### 5.1.4.2 Vulnerability Assessment:

The scanner asses the impact of the discovered vulnerability and prints out he severity of the vulnerability in colors for example when a vulnerability is critical or high the scanner will print red color.

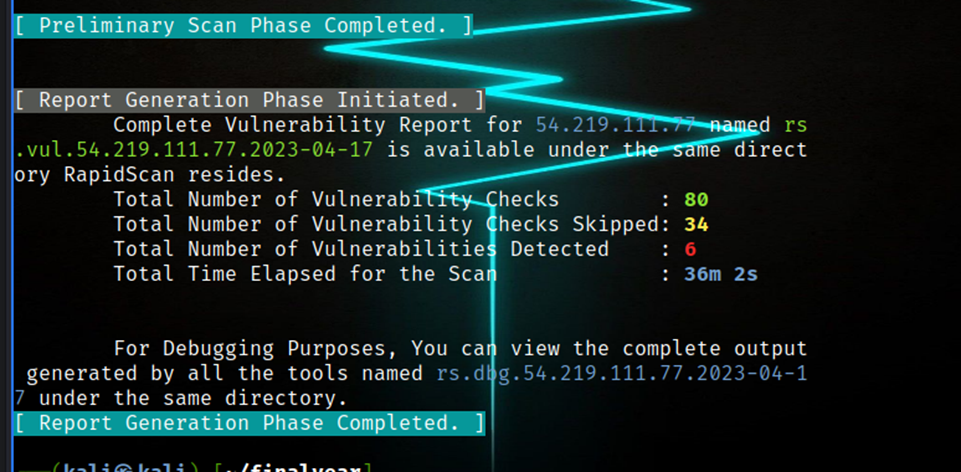
For example as we were testing our tool, it discovered a high vulnerability that it asses it according to the severity

During the testing we also discovered low vulnerabilities which could not require much attention and so it printed blue color as programmed.

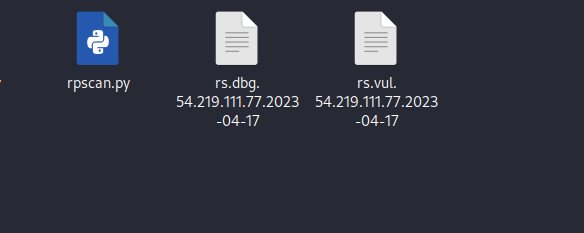
As we were testing another end point we also discovered a high vulnerability and the assessment colour coded it red as programmed since an outdated server could give a lee way for attackers to execute code remotely.



#### **5.1.4.3 Reporting:**

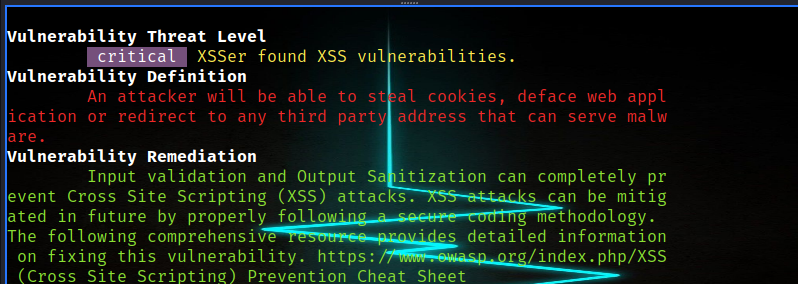
The scanner is able to produce a summarized executive report at the end of scanning all the system. As executive reports are always not detailed and not technical, the scanner out puts a report containing statistics of the scan for example the time taken, how many vulnerabilities discovered and which vulnerabilities giving their severity.

The scanner continues and generates a technical report that is an output file saved automatically in the same directory you ran the tool from. The report is always named the date of the scan and the environment scanned for example **rs.vul.google.com.2023-04-17** as another file is a debug file where the errors are dumped for the purposes of review of the scan. This file helped the developers to always keep track of how the tool behaves on different environment.



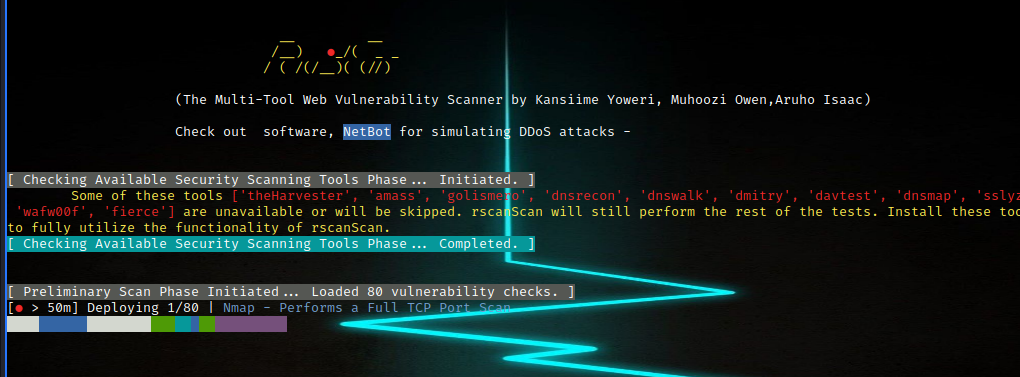
**5.1.4.4 Remediation Management:**

The scanner is able to remediate the vulnerability discovered by prividing clear guidance on how to patch the vulnerability. It continues to give links for reference on the how to remediate the discovered vulnerability.

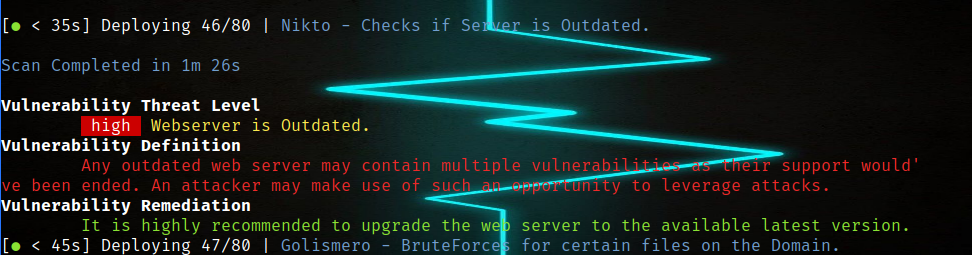


#### **5.1.4.5 Integration**

The scanner uses it integration capacity to find vulnerabilities as shown in the image above where nikto tool was able to detect the outdated web server and the scanner color codes the vulnerability as high since according to the vulnerability definition the attacker could make use of such vulnerability to l average attacks as seen the tools with red color will not be integrated with our scanner because they are not installed. Initially the tools first checks for tools to intergrate with before the start of scan

As shown the tools color coded with red are not installed in the system which means that the tool is not going to integrate with them.

Integration functionality helped our scanner minimize false positives as those intergrated tools could discover vulnerabilities that our scanner could not discover hence making the success rate or our tool high. For example Nikto tool which is a stand alone tool was integrated to our scanner and hence it discovered an outdated server which could prompt remote code execution vulnerability as shown below Nikto checked if the server is outdated and true the server was.



#### **5.1.4.6 Compliance:**

Our vulnerability scanner tool is able to comply with relevant industry standards, regulations, and guidelines, such as PCI DSS, HIPAA, and ISO 27001 in the following ways.

Documentation: Keeping accurate records of our compliance efforts, including the results of our vulnerability scans and any actions taken to address identified vulnerabilities. This is achived in the out put reports after the scan where by the report gives details of the scan and what was scanned and at what time.

This part of the report helps the vendor to know whether the tool was used to the ethical roles or if it was misused to scan what doesn’t belong to the organization hence our tool’s compliance with standards of PCI and HIPPA and ISO 27001.

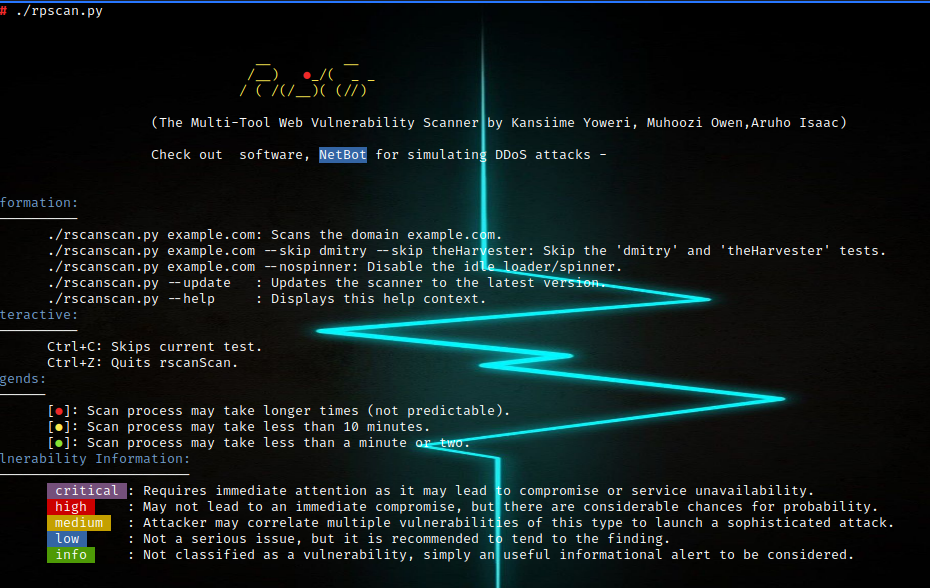
Payment Card Industry Data Security Standard (PCI DSS) requires regular vulnerability assessments of all systems and applications, including web servers. Our scanner can be used as part of a comprehensive vulnerability assessment program to meet this requirement.

Similarly, the Health Insurance Portability and Accountability Act (HIPAA) requires organizations to implement technical safeguards to ensure the confidentiality, integrity, and availability of electronic protected health information (ePHI). Regular vulnerability assessments are an important part of maintaining the security of ePHI, and rScan can be used as part of such assessments given its capability to integrate other tools to discover what malicious actors may exploit.

Overall, our tool can be used as part of a broader compliance strategy, but it is not a standalone solution for compliance with industry standards, regulations, and guidelines.

**5.2 User Guide**

In order to demonstrate the user friendliness of the tool few screen shots have been shown below. Further some other user interfaces are included in the Appendix.

The scanner operates beginning with the display of the scanner logo and the version of the scanner you are running and the manual guidance of the usage. For example, ctrl+c skips the current test while ctrl+z quits the scanner.

As the usage guidance requires the user to pass the IP address or URL, the console prompt for the for the user to pass a parameter which may be an IP address or URL, then the tool starts finding available tools to integrate with. The scanner reports the tool found on the system and then it colors codes tools un available

## 5.2 Testing

Software Testing is the process of executing a program or system with the intent of finding errors. The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions. Individual components were tested independently; this was followed by testing vulnerable and non-vulnerable systems both web based and network based systems.

### 5.2.1 Performance Testing

**USABILITY**  
The system can be used in user interface versions of windows /XP/7/8  
The system user interface shall be designed for ease of use.  
The system user interface shall provide help tips for the user to understand.

**RELIABILITY**  
The scanner is to be used 24 hours per day or more as long as its is scanning  
The scanner provides accurate results because it has integrated with other tools to limit false positives.  
The scanner is bugs free.

**PE****RFORMANCE**  
The system will perform faster to found the threat in the given system, defined the threat in a very non-technical plain English and then find the best way to remediate the threat

**PO****RTABILITY**

The scanner can be used in any desktop user interface version Linux and in windows as long as the tools to integrate with the system are installed  
The scanner can be installed on a single command from GitHub

Table 4.1: Rscanner Reporting of All Vulnerabilities some Site tested

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| source | site | SQL injection | xxs | RCE | Path traversal | Open ports | Server misconfiguration | Command injection | config-File exposure |
| OWASP | <http://10.10.80.211/blood/view.php?id=1> | yes | no | no | yes | yes | no | no | no |
| tryhakme | http://13.57.220.91/index.php?view=contact-us.html | no | no | yes | no | yes | yes | yes | yes |
| tryhackme | https://tryhackme.com/room/basicpentestingjt | no | no | no | no | yes | yes | no | yes |
| cyber-talents | http://wcamxwl32pue3e6mrndvjeyse3y4e435p0m0b835-web.cybertalentslabs.com/ | no | no | yes |  | yes | no | yes | no |
|  |  |  |  |  |  |  |  |  |  |

• Specific filters prevent scanner from functioning in MCIR - these are either length filters, which prevent the large test strings used, or quote filters, which prevent the XSS payloads which always contain strings from being processed.  
• There are many missed cases on WAVSEP, which is a site specifically for testing scanners.  
These appear to fail for several reasons, all some variant of those found on other sites.

### 5.2.2 Functionality Testing

We carried out functionality testing and we archived excellent result because we tested individual components and functionalities.

The update functionality worked perfectly as the command update could visit our GitHub to find new release or new updates.

The cltl+c command also worked perfectly where by during the scanning this command could skip the current scan hence the functionality worked well and this functionality saves time of the scan.

The Algorithm that automatically matches the suitable tool to the exact vulnerability worked well as nmap tool could be matched to port scanning and what web tool could be matched to web technology detection and this process was automatic and perfect.

Integration. The scanner detected the available tools to integrate with and only programmed tools to integrate were the ones integrated as long as they are installed in the system hence the functionality worked well.

Assessment. The tool is able to detect and assess the vulnerability and out the severity of the vulnerability as either low, medium or high in color codes. This prompts the remediation functionality to crawl the web to find the suitable remediation after defining the vulnerability.

Report generation. The scanner was able to generate two reports one being a precise executive report and another being a comprehensive technical report as an out file in the same directory.

### 5.2.3 Challenges Faced

Testing our scanner on other platforms was a challenge since platforms like windows and macOS don’t have tools like nmap, wpscan pre-installed in them unlike in Linux where all tools integrated to our scanner are installed.

Choosing a reputable vendor: As we don’t intend to make this tool public, choosing a vendor with a good reputation and track record of providing reliable and effective vulnerability scanning services was a challenge since the vendors could not prove their good reputation background.

There was resistance during testing phase as some organizations refused us to test the scanner on their systems regardless of all possible confidential agreements.

The picking of appropriate tools to integrate with our scanner was a challenge as there are many scanners that are open source hence choosing the best to suite our scanner needs was a challenge.

Closed source tools. Some good tools could not allow us to integrate them with our scanner as they were closed source hence we could not read their source code to get the best concept of how to integrate it with our scanner.

Money challenges here and there to for necessities like transport to move to and from the area of Study, also stopping public transport for quite some time meant that we the researches had to improvise for most of the research online increasing expenses spent on data to carry out research.

# **CHAPTER SIX**

# **CONCLUSION AND RECOMMENDATION CONCLUSION**

## 6.0 Introduction

The aim of chapter six is to provide a clear and comprehensive understanding of the data collected and analyzed during the project process. It is an essential chapter that supports the project findings and contributes to the overall credibility of the project report. This chapter examines how the objectives of the study were achieved using the system that was developed.

## 6.1 CONCLUSION

Nowadays Internet pages are more complex than they have ever been containing a huge quantity of dynamical content, which is housed for the user.

Dynamical functionality leads to a higher sensitivity to cross-site scripting attacks, SQL Injection, CSRF and theft of private user data.

It is not vital whether the attack has occurred from the outer or the inner network, the only fact that data on personal computers are insecure threatens the business of companies which can easily  
lose the market competition if certain confidential information is stolen.

Firms which pay higher attention to security of their applications and computers have higher chances of survival on the market. Firms should realize that with the trust users give them while giving their private data, comes a big responsibility. For Instances of such an attack happened to the firm Sony PlayStation. Users' credit card information was stolen, and the firm not only lost money, but all  
of their credibility with its users as well. It is assumed that 77 million of user data was lost.

That attack is one of the biggest attacks on systems so far, and it really stresses the fact that firms have to pay a great deal of attention to data security.

## 6.2 Recommendations

The findings of the scanner recommended Continuously improve the scanner to ensure continuously evaluate and improve our vulnerability scanning tool by incorporating feedback from stakeholders, monitoring performance metrics, and adjusting the program as needed to meet changing security requirements.

From the testing results, it is recommended to conduct regular scans as conducting regular scans of systems helps to ensure that vulnerabilities are identified and addressed in a timely manner.

As organizations have different preferences, it is recommended to customize the tool: Configuring this vulnerability scanner tool could help to meet the specific needs of the organization or client using it. This may involve adjusting the tool's settings or integrating the tool with other security tools not included in the original program.

It is recommended that Organizations should consult with their compliance teams to determine the appropriate use of this scanner in their specific compliance context so that compliance standards like ISO

are not misused.

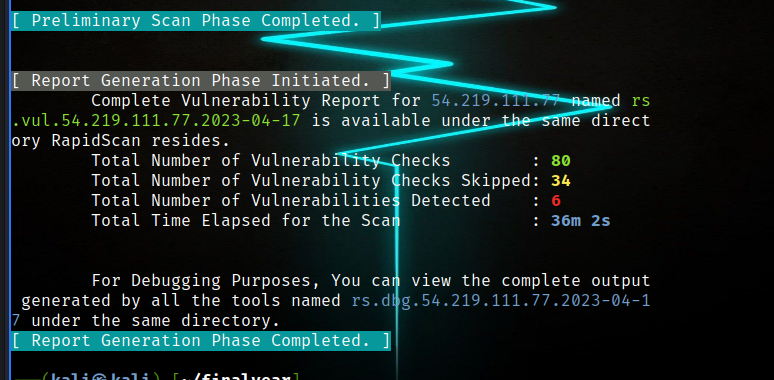
.

## 6.3 Future study

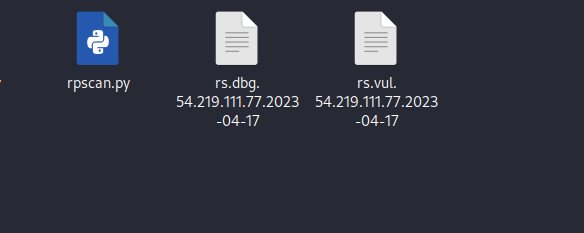
This work can be extended by creating dynamic code analyzer plugin for all programming languages. Another follow-up work may focus on detection of new kind of vulnerabilities with respect to source code.

Increased automation: As cyber threats become more sophisticated and widespread, there is likely to be a growing demand for vulnerability scanners that can automate the identification and remediation of vulnerabilities. This could involve the development of more advanced scanning algorithms, as well as the integration of machine learning and artificial intelligence technologies.

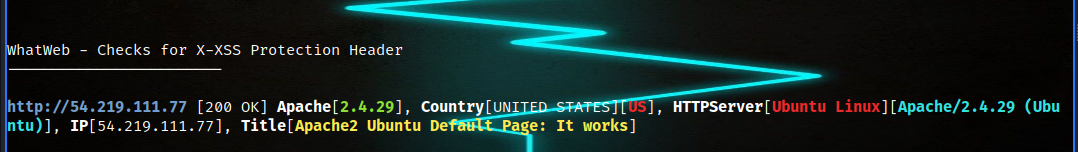
LIST OF APPENDICES  
APPENDIX 1



apendix 2



Appendix 3



SAMPLE CODE

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

# \_\_ \_\_

# /\_\_)\_ '\_/( \_ \_

# / ( (//)/(/\_\_)( (//)

# /

#

# Author : KANSIIME Y0WER | MUHOOZI OWEN | ARUHO ISAAC

# Tool : rScan v1.0

# Usage : python3 rscan.py example.com

# Description: This scanner automates the process of security scanning by using a

# multitude of available security tools and some custom scripts.

#

# Importing the libraries

import sys

import argparse

import subprocess

import os

import time

import random

import threading

import re

import random

from urllib.parse import urlsplit

CURSOR\_UP\_ONE = '\x1b[1A'

ERASE\_LINE = '\x1b[2K'

# Scan Time Elapser

intervals = (

('h', 3600),

('m', 60),

('s', 1),

)

def display\_time(seconds, granularity=3):

result = []

seconds = seconds + 1

for name, count in intervals:

value = seconds // count

if value:

seconds -= value \* count

result.append("{}{}".format(value, name))

return ' '.join(result[:granularity])

def terminal\_size():

try:

rows, columns = subprocess.check\_output(['stty', 'size']).split()

return int(columns)

except subprocess.CalledProcessError as e:

return int(20)

def url\_maker(url):

if not re.match(r'http(s?)\:', url):

url = 'http://' + url

parsed = urlsplit(url)

host = parsed.netloc

if host.startswith('www.'):

host = host[4:]

return host

def check\_internet():

os.system('ping -c1 github.com > rs\_net 2>&1')

if "0% packet loss" in open('rs\_net').read():

val = 1

else:

val = 0

os.system('rm rs\_net > /dev/null 2>&1')

return val

# Initializing the color module class

class bcolors:

HEADER = '\033[95m'

OKBLUE = '\033[94m'

OKGREEN = '\033[92m'

WARNING = '\033[93m'

BADFAIL = '\033[91m'

ENDC = '\033[0m'

BOLD = '\033[1m'

UNDERLINE = '\033[4m'

BG\_ERR\_TXT = '\033[41m' # For critical errors and crashes

BG\_HEAD\_TXT = '\033[100m'

BG\_ENDL\_TXT = '\033[46m'

BG\_CRIT\_TXT = '\033[45m'

BG\_HIGH\_TXT = '\033[41m'

BG\_MED\_TXT = '\033[43m'

BG\_LOW\_TXT = '\033[44m'

BG\_INFO\_TXT = '\033[42m'

BG\_SCAN\_TXT\_START = '\x1b[6;30;42m'

BG\_SCAN\_TXT\_END = '\x1b[0m'

# Classifies the Vulnerability's Severity

def vul\_info(val):

result =''

if val == 'c':

result = bcolors.BG\_CRIT\_TXT+" critical "+bcolors.ENDC

elif val == 'h':

result = bcolors.BG\_HIGH\_TXT+" high "+bcolors.ENDC

elif val == 'm':

result = bcolors.BG\_MED\_TXT+" medium "+bcolors.ENDC

elif val == 'l':

result = bcolors.BG\_LOW\_TXT+" low "+bcolors.ENDC

else:

result = bcolors.BG\_INFO\_TXT+" info "+bcolors.ENDC

return result

# Links the vulnerability with threat level and remediation database

def vul\_remed\_info(v1,v2,v3):

print(bcolors.BOLD+"Vulnerability Threat Level"+bcolors.ENDC)

print("\t"+vul\_info(v2)+" "+bcolors.WARNING+str(tool\_resp[v1][0])+bcolors.ENDC)

print(bcolors.BOLD+"Vulnerability Definition"+bcolors.ENDC)

print("\t"+bcolors.BADFAIL+str(tools\_fix[v3-1][1])+bcolors.ENDC)

print(bcolors.BOLD+"Vulnerability Remediation"+bcolors.ENDC)

print("\t"+bcolors.OKGREEN+str(tools\_fix[v3-1][2])+bcolors.ENDC)

# rScan Logo

def logo():

print(bcolors.WARNING)

logo\_ascii = """

\_\_ \_\_

/\_\_)\_ """+bcolors.BADFAIL+" ●"+bcolors.WARNING+"""\_/( \_ \_

/ ( (//)/(/\_\_)( (//)

/

"""+bcolors.ENDC+"""(The Cmd-line based Vulnerability Scanner developed by Kansiime Yoweri, Muhoozi Owen,Aruho Isaac)

FINAL YEAR PROJECT, """+bcolors.BG\_LOW\_TXT+"""Version 1.0"""+bcolors.ENDC+""" for new release visit - https://github.com/khanzjob/rSCANNER/

"""

print(logo\_ascii)

print(bcolors.ENDC)

# Command that is used to initiate the tool (with parameters and extra params)

tool\_cmd = [

#1

["host ",""],

#2

["wget -O /tmp/rscan\_temp\_aspnet\_config\_err --tries=1 ","/%7C~.aspx"],

#3

["wget -O /tmp/rscan\_temp\_wp\_check --tries=1 ","/wp-admin"],

#4

["wget -O /tmp/rscan\_temp\_drp\_check --tries=1 ","/user"],

#5

["wget -O /tmp/rscan\_temp\_joom\_check --tries=1 ","/administrator"],

#6

["uniscan -e -u ",""],

#7

["wafw00f ",""],

#8

["nmap -F --open -Pn ",""],

# Tool Responses (Begins) [Responses + Severity (c - critical | h - high | m - medium | l - low | i - informational) + Reference for Vuln Definition and Remediation]

tool\_resp = [

#1

["Does not have an IPv6 Address. It is good to have one.","i",1],

#2

["ASP.Net is misconfigured to throw server stack errors on screen.","m",2],

#3

["WordPress Installation Found. Check for vulnerabilities corresponds to that version.","i",3],

#4

["Drupal Installation Found. Check for vulnerabilities corresponds to that version.","i",4],

#5

["Joomla Installation Found. Check for vulnerabilities corresponds to that version.","i",5],

#6

["robots.txt/sitemap.xml found. Check those files for any information.","i",6],

#7

["No Web Application Firewall Detected","m",7],

#################### Report & Documentation Phase ###########################

date = subprocess.Popen(["date", "+%Y-%m-%d"],stdout=subprocess.PIPE).stdout.read()[:-1].decode("utf-8")

debuglog = "rs.dbg.%s.%s" % (target, date)

vulreport = "rs.vul.%s.%s" % (target, date)

print(bcolors.BG\_HEAD\_TXT+"[ Report Generation Phase Initiated. ]"+bcolors.ENDC)

if len(rs\_vul\_list)==0:

print("\t"+bcolors.OKGREEN+"No Vulnerabilities Detected."+bcolors.ENDC)

else:

with open(vulreport, "a") as report:

while(rs\_vul < len(rs\_vul\_list)):

vuln\_info = rs\_vul\_list[rs\_vul].split('\*')

report.write(vuln\_info[arg2])

report.write("\n------------------------\n\n")

temp\_report\_name = "/tmp/rscan\_temp\_"+vuln\_info[arg1]

with open(temp\_report\_name, 'r') as temp\_report:

data = temp\_report.read()

report.write(data)

report.write("\n\n")

temp\_report.close()

rs\_vul = rs\_vul + 1

# Writing all scan files output into RS-Debug-ScanLog for debugging purposes.

for file\_index, file\_name in enumerate(tool\_names):

with open(debuglog, "a") as report:

try:

with open("/tmp/rscan\_temp\_"+file\_name[arg1], 'r') as temp\_report:

data = temp\_report.read()

report.write(file\_name[arg2])

report.write("\n------------------------\n\n")

report.write(data)

report.write("\n\n")

temp\_report.close()

except:

break

report.close()

print("\tTotal Number of Vulnerability Checks : "+bcolors.BOLD+bcolors.OKGREEN+str(len(tool\_names))+bcolors.ENDC)

print("\tTotal Number of Vulnerability Checks Skipped: "+bcolors.BOLD+bcolors.WARNING+str(rs\_skipped\_checks)+bcolors.ENDC)

print("\tTotal Number of Vulnerabilities Detected : "+bcolors.BOLD+bcolors.BADFAIL+str(len(rs\_vul\_list))+bcolors.ENDC)

print("\tTotal Time Elapsed for the Scan : "+bcolors.BOLD+bcolors.OKBLUE+display\_time(int(rs\_total\_elapsed))+bcolors.ENDC)

print("\n")

print("\tFor Debugging Purposes, You can view the complete output generated by all the tools named "+bcolors.OKBLUE+debuglog+bcolors.ENDC+" under the same directory.")

print(bcolors.BG\_ENDL\_TXT+"[ Report Generation Phase Completed. ]"+bcolors.ENDC)

os.system('setterm -cursor on')

os.system('rm /tmp/rscan\_te\* > /dev/null 2>&1')

# Clearing previous scan files

# **REFERENCES**

1. Mark Ryan M. Talabis, D. Kaye, in Information Security Analytics.
2. 2015 Almantas Kakareka, in Computer and Information Security Handbook (Third Edition),2013 W3af Web Application Attack and Audit Framework (More Information: [http://w3af.org/)](http://w3af.org/)
3. The Ultimate Guide to Vulnerability Scanning([https://www.intruder.io/guides)](https://www.intruder.io/guides)

1. Kumar Singh and S. Roy, "A network based vulnerability scanner for detecting SQLI attacks in web applications," 2012 1st International Conference on Recent Advances in

Information Technology (RAIT), Dhanbad, 2012, pp. 585-590, doi:

10.1109/RAIT.2012.6194594.

1. Aweke, A. M., Alemneh, E. T., & Mihretie, G. W. (2018). Automated vulnerability scanner tool for network security assessment. Journal of Cybersecurity and Information Management, 5(2), 45-54.
2. K. D'silva, J. Vanajakshi, K. N. Manjunath and S. Prabhu, "An effective method for preventing SQL injection attack and session hijacking," 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, 2017, pp. 697-701, doi: 10.1109/RTEICT.2017.8256687.

1. C. Ping, "A second-order SQL injection detection method," 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, 2017, pp. 1792-1796, doi: 10.1109/ITNEC.2017.8285104.

1. Adjei, E., & Nyarko-Boateng, F. (2019). Automated vulnerability assessment: A review. Journal of Information Security, 10(2), 68-78. Deng, H., Li, Q., Ma, X., Li, H., Li, Z., & Huang, H. (2017).

1. Chen, X., Qian, C., Xie, Q., & Shi, W. (2019). A comparative study of web application vulnerability scanners. Journal of Computer Security, 27(3), 331-358.

1. Li, X., & Li, H. (2020). Comparative evaluation of web application vulnerability scanners. Journal of Computer Science and Technology, 35(2), 306-327.

1. PortSwigger. (n.d.). Burp Suite. Retrieved from https://portswigger.net/burp

1. Torrecillas, J., Díaz-Verdejo, J. E., & Torres-Huitzil, C. (2020). Web application security testing using open-source tools: A case study. Journal of Information Security and Applications, 50, 102446.

1. Li, X., & Li, H. (2020). Comparative evaluation of web application vulnerability scanners. Journal of Computer Science and Technology, 35(2), 306-327.

1. Rai, A. K., Chauhan, S., & Singh, S. (2020). Comparative analysis of popular vulnerability scanners. International Journal of Computer Science and Information Security, 18(10), 186-193.

1. Santos, T. L., Gonçalves, L. H., & Almeida, T. A. (2019). Vulnerability assessment and penetration testing: A case study in a corporate network. International Journal of Advanced Computer Science and Applications, 10(10), 234-239.

1. Almehmadi, S., Alqahtani, S., & Altuwaijri, A. (2018). Web application security: Case study of Acunetix in identifying and remedying vulnerabilities. International Journal of Computer Applications, 181(38), 39-44.

1. Kaur, G., & Singh, P. (2018). Performance evaluation of web application vulnerability scanners. International Journal of Computer Science and Mobile Computing,
2. Ferreira, L., Neves, R., & Carvalho, T. (2019). Rapid7 vulnerability scanner: A case study. Journal of Information Systems Engineering & Management, 4(4), 38.

1. Rapid7. (n.d.). Vulnerability management. Retrieved from <https://www.rapid7.com/solutions/vulnerability-management/>

1. Tan, B. Y., Tan, C. C., & Chua, T. L. (2020). Comparative evaluation of vulnerability scanners. Journal of Information Security, 11(3), 105-114.

1. Lee, C., Kim, K., & Kim, H. (2018). Vulnerability assessment and management of information systems using OpenVAS. Journal of Information Processing Systems, 14(2), 475-488.

1. Madani, S., et al. (2020). OpenVAS: A comparative study on open-source security scanners. Journal of King Saud University-Computer and Information Sciences, 32(4), 420-430.

1. OpenVAS. (n.d.). OpenVAS scanner. Retrieved from <https://openvas.org/>

1. Kiran, K., et al. (2019). A comparative study of vulnerability assessment tools for web application security. International Journal of Innovative Technology and Exploring Engineering, 8(12), 1750-1756.

1. Rapid7. (n.d.). Nexpose. Retrieved from https://www.rapid7.com/products/nexpose/

1. Shen, Y., et al. (2018). A comparative study of network vulnerability assessment tools. International Journal of Security and Its Applications, 12(1), 91-102.

1. Wang, J., et al. (2018). Network security risk assessment using vulnerability scanning

1. Qualys. (n.d.). Qualys Vulnerability Management. Retrieved from https://www.qualys.com/products/vulnerability-management/

1. Shen, Y., et al. (2018). A comparative study of network vulnerability assessment tools. International Journal of Security and Its Applications, 12(1), 91-102.

1. Sharma, N., et al. (2020). Identification of network vulnerabilities using the Qualys tool. Journal of Engineering and Applied Sciences, 15(2), 427-432.

1. Wang, J., et al. (2018). Network security risk assessment using vulnerability scanning. Journal of Physics: Conference Series, 1065(5), 052016.

1. CIRT. (n.d.). Nikto - The Open Source Web Server Scanner. Retrieved from https://cirt.net/Nikto2

1. Ahmad, S., et al. (2018). A comparative study of web vulnerability scanners. In Proceedings of the 3rd International Conference on Computing, Communication and Security (ICCCS'18) (pp. 1-5). IEEE.

1. Hossain, M. A., et al. (2019). Nikto: a web server vulnerability scanner for security assessment. In Proceedings of the 4th International Conference on Electrical Engineering and Information & Communication Technology

1. Nmap. (n.d.). Nmap - Free Security Scanner For Network Exploration & Security Audits. Retrieved from https://nmap.org/

1. Ahmad, S., et al. (2018). A comparative study of web vulnerability scanners. In Proceedings of the 3rd International Conference on Computing, Communication and Security (ICCCS'18) (pp. 1-5). IEEE.

1. Yadav, A., et al. (2020). A study on Nmap as a vulnerability scanner tool. International Journal of Advanced Research in Computer Science and Software Engineering, 10(5), 450
2. OpenVAS: Greenbone Networks. (n.d.). OpenVAS. Retrieved from https://www.greenbone.net/en/openvas/

1. Qualys: Qualys. (n.d.). Vulnerability management. Retrieved from https://www.qualys.com/solutions/vulnerability-management/

1. Rapid7: Rapid7. (n.d.). Vulnerability management. Retrieved from https://www.rapid7.com/solutions/vulnerability-management/

1. Acunetix: Acunetix. (n.d.). Web vulnerability scanner. Retrieved from https://www.acunetix.com/vulnerability-scanner/

1. Burp Suite: PortSwigger. (n.d.). Burp Suite. Retrieved from https://portswigger.net/burp

1.