

VULNERABILITY DETECTION SYSTEM
(XPOSED)

NURIN AZYYATI BINTI KAMILIZAHRI

Bachelor of Computer Science
(Software Engineering)

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : NURIN AZYYATI BINTI KAMILIZAHRI

Date of Birth

Title : VULNERABILITY DETECTION SYSTEM (XPOSED)

Academic Session : SEMESTER II ACADEMIC SESSION 2022/2023

I declare that this thesis is classified as:

- CONFIDENTIAL (Contains confidential information under the Official Secret Act 1997)*
- RESTRICTED (Contains restricted information as specified by the organization where research was done)*
- OPEN ACCESS I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

(Student's Signature)

New IC/Passport Number

Date: 9/6/2023

(Supervisor's Signature)

Dr. Al-Fahim Bin Mubarak Ali

Name of Supervisor

Date: 9/06/2023

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name
Thesis Title

Reasons (i)

 (ii)

 (iii)

Thank you.

Yours faithfully,

(Supervisor's Signature)

Date:

Stamp:

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.



SUPERVISOR'S DECLARATION

I/~~We~~* hereby declare that I/~~We~~* have checked this thesis/project* and in my/~~our~~* opinion, this thesis/project* is adequate in terms of scope and quality for the award of the degree of Bachelor of Computer Science (Software Engineering) with Honours.

(Supervisor's Signature)

Full Name : Dr. Al-Fahim Bin Mubarak Ali

Position : Senior Lecturer

Date : 9/06/2023

(Co-supervisor's Signature)

Full Name :

Position :

Date :



STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

(Student's Signature)

Full Name : NURIN AZYYATI BINTI KAMILIZAHRI

ID Number: CB20155

Date : 9/6/2023

VULNERABILITY DETECTION SYSTEM
(XPOSED)

NURIN AZYYATI BINTI KAMILIZAHRI

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Computer Science (Software Engineering) with Honors

Faculty of Computing
UNIVERSITI MALAYSIA PAHANG

JUNE 2023

ACKNOWLEDGEMENTS

First, I am deeply grateful to my supervisor, Dr. Al-Fahim Bin Mubarak Ali, for his invaluable guidance, constant encouragement, and unwavering support throughout the project. His guidance was essential in helping me develop a successful project.

Besides, I also extend my sincere thanks to my parents for their unwavering love, care, and sacrifice throughout my life, which has been an inspiration to me from my earliest days. I am unable to fully express my appreciation for their support and belief in my abilities.

Lastly, I am grateful to my friends, who helped me in countless ways and made my degree journey at UMP a pleasant and unforgettable experience. I would like to particularly thank my classmates for their excellent cooperation, inspiration, and support during my studies. I will always remember this three-year experience with all of you.

I also want to extend my thanks to anyone who has contributed to this project directly or indirectly, your contributions will not be forgotten.

ABSTRAK

Apabila dunia menjadi semakin digital, sistem perisian telah menjadi sasaran utama penyerang siber. Dengan percambahan teknologi baharu dan kerumitan sistem perisian yang semakin meningkat, bilangan kelemahan yang terdapat dalam sistem ini juga telah meningkat. Kerentanan ini boleh terdiri daripada ralat pengkodan mudah kepada kecacatan seni bina yang lebih kompleks, dan ia boleh membawa akibat yang serius kepada organisasi dan individu. Untuk memerangi ancaman yang semakin meningkat ini, adalah penting untuk membangunkan sistem pengesanan kerentanan yang berkesan yang boleh mengenal pasti dan mengklasifikasikan risiko keselamatan dengan cepat dan tepat dalam sistem perisian. Sistem Pengesanan Kerentanan XPOSED, dibangunkan sebagai projek tahun akhir, menggunakan teknik pengimbasan untuk mengesan dan mengklasifikasikan kelemahan dalam sistem perisian, dengan itu membantu organisasi dalam mencegah serangan siber dan melindungi aset penting mereka.

ABSTRACT

As the world becomes increasingly digitized, software systems have become a primary target for cyber attackers. With the proliferation of new technologies and the growing complexity of software systems, the number of vulnerabilities present in these systems has also grown. These vulnerabilities can range from simple coding errors to more complex architectural flaws, and they can have serious consequences for organizations and individuals alike. To combat this growing threat, it is essential to develop effective vulnerability detection systems that can quickly and accurately identify and classify security risks in software systems. XPOSED Vulnerability Detection System, developed as a final year project, employs scanning techniques to detect and classify vulnerabilities in software systems, thereby assisting organizations in preventing cyber-attacks and safeguarding their essential assets.

TABLE OF CONTENT

DECLARATION	
TITLE PAGE	
ACKNOWLEDGEMENTS	ii
ABSTRAK	iii
ABSTRACT	iv
TABLE OF CONTENT	v
LIST OF TABLES	viii
LIST OF FIGURES	ix
LIST OF SYMBOLS	x
LIST OF ABBREVIATIONS	xi
CHAPTER 1 INTRODUCTION	1
1.1 Introduction	1
1.2 Problem Statement	2
1.3 Objectives	2
1.4 Scope	3
1.5 Thesis Organization	3
CHAPTER 2 LITERATURE REVIEW	4
2.1 Introduction	4
2.2 Software Vulnerability	4
2.2.1 OWASP Top 10	4
2.2.2 CWE SANS Top 25	6

2.2.3	Mapping of OWASP and CWE	8
2.3	Analysis of Existing Sources	10
2.4	Studies on Existing Systems	11
2.4.1	Acunetix	11
2.4.2	Nessus	12
2.4.3	Qualys	13
2.5	Comparison on Existing Systems	14
2.6	Summary	15
 CHAPTER 3 METHODOLOGY		16
3.1	Introduction	16
3.2	Software Development Life Cycle (SDLC)	16
3.2.1	Phase 1 – REQUIREMENTS PLANNING	17
3.2.2	Phase 2 – USER DESIGN	18
3.2.3	Phase 3 – CONSTRUCTION	18
3.2.4	Phase 4 – CUTOVER	18
3.3	Project Requirement	19
3.4	Propose Design	21
3.4.1	Flowchart	21
3.4.2	Context Diagram	24
3.4.3	Use Case Diagram	25
3.4.4	Storyboard	36
3.5	Data Design	39
3.5.1	Entity Relationship Diagram (ERD)	39
3.5.2	Data Dictionary	40
3.6	Proof of Initial Concept	45

3.6.1	Dashboard Interface	45
3.6.2	Add New Scan Interface	46
3.6.3	Scan List Interface	47
3.6.4	Scan Details Interface	48
3.6.5	Scan Info Interface	49
3.6.6	History Interface	50
3.7	Testing / Validation Plan	51
3.8	Potential Use of Proposed Solution	52
3.9	Gantt Chart	53
CHAPTER 4 RESULTS AND DISCUSSION		54
4.1	Introduction	54
4.2	Implementation	55
4.2.1	User Manual	56
4.2.2	Database Design	64
4.2.3	Coding Implementation	69
4.3	Testing and Result Discussion	71
CHAPTER 5 CONCLUSION		74
5.1	Introduction	74
5.2	Limitations and Constraints	76
5.3	Future Work	77

LIST OF TABLES

Table 2.1	List of OWASP Top 10 Vulnerabilities 2021	5
Table 2.2	List of CWE SANS Top 25 Weaknesses 2022	6
Table 2.3	Mapping between OWASP 2021 and CWE SANS 2022	8
Table 2.4	Comparison between all three existing system	14
Table 3.1	Project requirements for XPOSED system	19
Table 3.2	Use case description of XPOSED system	26

LIST OF FIGURES

Figure 3.1	Rapid Application Development Process	17
Figure 3.2	General workflow of XPOSED system	21
Figure 3.3	Detailed workflow of End Users	22
Figure 3.4	Detailed workflow of Administrator	23
Figure 3.5	Context diagram of XPOSED system	24
Figure 3.6	Use case diagram of XPOSED system	25
Figure 3.7	ERD for XPOSED system	39
Figure 3.8	Dashboard interface of XPOSED	45
Figure 3.9	Add New Scan interface of XPOSED	46
Figure 3.10	Scan Lists interface of XPOSED	47
Figure 3.11	Scan Details interfaces of XPOSED	48
Figure 3.12	Scan Info interface of XPOSED	49
Figure 3.13	Scan History interface of XPOSED	50
Figure 3.14	Gantt chart table of XPOSED project	53
Figure 4.1	Register page	56
Figure 4.2	Login page	57
Figure 4.3	Dashboard overview for users	58
Figure 4.4	Interface when uploading folder	59
Figure 4.5	List of scans page	60
Figure 4.6	Scan details page	61
Figure 4.11	Implementation of xposed database	64
Figure 4.14	Structure of migrations table	65
Figure 4.15	Structure of reports table	65
Figure 4.16	Structure of risks table	66
Figure 4.17	Structure of scans table	67
Figure 4.18	Structure of users table	68

LIST OF SYMBOLS

SBPWM	Simple Boost Pulse Width Modulation
ZSI	Z source inverter

LIST OF ABBREVIATIONS

SBPWM	Simple Boost Pulse Width Modulation
ZSI	Z source inverter

CHAPTER 1

INTRODUCTION

1.1 Introduction

Every day, more software vulnerabilities, particularly in systems and applications, are found. This is because software applications are now one of our most essential demands, particularly for reducing daily workload. To meet the needs of the customers, the number of software developers is growing in tandem with the growth of software applications. However, this situation gave rise to a wide range of vulnerabilities that might represent a serious risk of being exploited, loss of data, and a chance for attackers to find weaknesses in the systems of their victims.

With that, developers would find way to develop their software applications in more secure by referring to the most popular and trusted sources which are OWASP and CWE SANS. Both mentioned sources are important as a guidance to the developers to maintain a good practices to prevent dangerous attack from attackers. CWE SANS is more detailed compared to the OWASP as it is more general. CWE SANS is considered a high-level category as it has more detailed explanation on the security risks. While for the OWASP it is considered a low-level category as it is quite general when listing the top 10 security risks. Therefore, developers would review these sources to implement best practises into their software, and attackers would discover these sources to use as a means of attacking their target.

Thus, this work focuses on creating a system for detection and analysis of software vulnerabilities in software application to understand the types of security risks that attackers used to harm their targets. With the help of this system, it may be possible to obtain detailed information about the security risks that possible to damage any software application as well as solutions for defending the software against attacks.

1.2 Problem Statement

Individuals or organizations may protect their software application with its own kind of protection to prevent such bad things from happening. However, the problem is, it is possible that the protection set up is only for a specific form of security risk, and they are unaware of other types of security threats, which could explain why their software application is still under attack despite the protection they've put in place. This will happen, if they are unaware about the exact type of security threats on their application. Therefore, using this tool can aid in determining the exact nature of the threats as well as in determining how to stop those attacks from occurring against the software application.

Another problem is that it takes time to quickly comprehend the nature of security risks. Normally, people would simply use Google to look up similar cases involving security threats that occur on their application and to look for a solution in order to address the problem right away. However, it is difficult to find a website that gives direct answer to the issue they faced. Thus, with this tool, security threats may be precisely identified and ways to avoid them will also be provided.

1.3 Objectives

The aim of the project is to develop a vulnerability detection tool that can be used to identify, classify, and provide a more thorough explanation of any vulnerability that is found.

The aims will be supported by 3 objectives. The objectives are:

- i. To study the types of vulnerabilities based on the standard vulnerabilities benchmark.
- ii. To develop a software vulnerabilities detection tool for software application.
- iii. To evaluate the develop tool using User Acceptance Test.

1.4 Scope

The following are the scope of this project:

- i. The target user for this project is for those who need to determine the different types of vulnerabilities in their software application.
- ii. The programming language used for vulnerability detection is limited to PHP and Java applications.
- iii. Only source code-related risks, in line with the OWASP Top 10, are targeted for detection.

1.5 Thesis Organization

This thesis consists of five chapters. Chapter 1 will cover the concept and justification for creating a detection tool for various vulnerabilities. Chapter 2 generally explain and discuss about the OWASP Top 10 and CWE SANS Top 25 lists of different software vulnerabilities. A mapping is being created to compare the vulnerabilities listed on the two sources. Chapter 3 explains the methodology in which the Software Development Life Cycle (SDLC) model will be used to develop the system. Each phase of the SDLC model must clearly describe the activities related to the project. This chapter also includes the system design and the interface of the system. In Chapter 4, the results and discussion are presented, highlighting the implementation and testing of this project. Chapter 5 concludes by summarizing the final outcome of the project, discussing its limitations, and suggesting future directions for further work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter discuss the literature review on which two different sources or articles, OWASP Top 10 and CWE SANS Top 25 being compared and discussed to see the relationship between the listed top security risks. Before starting the development process, literature reviews must be conducted first for requirement gathering. In this literature review, two existing sources are being compared for their list of vulnerabilities. The comparison will help to understand the top security risks present in the two sources as well as help to identify the best approach to develop the tool. From the comparison it will help to identify which 25 vulnerabilities in CWE SANS belongs to group in OWASP Top 10. There are a lot of differences between these two sources in terms of detail explanation on each of the listed vulnerabilities. Thus, this section is related to the software vulnerabilities, categorized based on standard benchmarking from OWASP Top 10 and CWE SANS 25.

2.2 Software Vulnerability

2.2.1 OWASP Top 10

The Open Web Application Security Project, also known as OWASP Top 10, is sponsored by OWASP, and is primarily known for educating individuals and organisations about the need for software security that exists in this world and assisting them in securing their software against security threats (*OWASP Top 10:2021*, n.d.). The Top 10 list was most recently updated in 2021. The list of OWASP's Top 10 Vulnerabilities in the year of 2021 is shown in Table 2.1. The OWASP project has created

a list of the top 10 global web application security threats, explaining each vulnerability with a practical example and providing tips on how to avoid it.

Table 2.1 List of OWASP Top 10 Vulnerabilities 2021

OWASP Rank	OWASP vulnerabilities
1	A01:2021-Broken Access Control
2	A02:2021-Cryptographic Failures
3	A03:2021-Injection
4	A04:2021-Insecure Design
5	A05:2021-Security Misconfiguration
6	A06:2021-Vulnerable and Outdated Components
7	A07:2021-Identification and Authentication Failures
8	A08:2021-Software and Data Integrity Failures
9	A09:2021-Security Logging and Monitoring Failures
10	A10:2021-Server-Side Request Forgery

OWASP Top 10 will aid in educating developers, designers, architects, and organizations about the most important web application security weaknesses which each of the Top 10 security risks will outline basic ways to defend the software application against those high-risk vulnerabilities. The Top 10 listed risks are referenced usually by produced projects from developers as developers learn from the mistakes of their past projects (*OWASP and Its Importance to Application Security – Conviso AppSec*, n.d.). For each of the Top 10 risks, it will clearly outline the factors that determine how severe a risk is often considered to be. It also explains how to determine whether or not the produced software application possesses the characteristics necessary to be attacked. Not only that, but it will also offer suggestions for reducing the security risks associated with the application.

2.2.2 CWE SANS Top 25

The National Cyber Security Division of the US Department of Homeland Security sponsors the CWE SANS Top 25 project, also known as Common Weakness Enumeration, to categorise security flaws. The most recent version, which included feedback from about 40 security experts and a list of typical errors made by software developers, was released in 2022. The list of OWASP's Top 10 Vulnerabilities is shown in Table 2.2. The top 25 software application errors are listed in order of severity (*CWE - 2022 CWE Top 25 Most Dangerous Software Weaknesses*, n.d.). All the listed errors can result in serious software vulnerabilities since they commonly let attackers to take entire control of the software, steal data, or stop the software from functioning at all. Therefore, it can be claimed that these security risks are common, simple to detect, and easy to exploit. Each of the listed risks will be described in detailed, along with suggestions for reducing and avoiding them.

CWE Top 25 will assist in preventing vulnerabilities by educating designers and programmers on how to avoid common mistakes before releasing the completed software product (*What Is Common Weakness Enumeration (CWE)? | Definition from TechTarget*, n.d.). With that, programmers may guard against the vulnerabilities that harm their software applications. Not only that, but clients should also be able to use the list to ask software developers to create applications that are more secure, and software managers should be able to use it as a measuring stick of progress in their efforts to secure their software.

Table 2.2 List of CWE SANS Top 25 Weaknesses 2022

Rank	ID	Name
[1]	CWE-787	Out-of-bounds Write
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[3]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[4]	CWE-20	Improper Input Validation
[5]	CWE-125	Out-of-bounds Read

[6]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[7]	CWE-416	Use After Free
[8]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[9]	CWE-352	Cross-Site Request Forgery (CSRF)
[10]	CWE-434	Unrestricted Upload of File with Dangerous Type
[11]	CWE-476	NULL Pointer Dereference
[12]	CWE-502	Deserialization of Untrusted Data
[13]	CWE-190	Integer Overflow or Wraparound
[14]	CWE-287	Improper Authentication
[15]	CWE-798	Use of Hard-coded Credentials
[16]	CWE-862	Missing Authorization
[17]	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')
[18]	CWE-306	Missing Authentication for Critical Function
[19]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer
[20]	CWE-276	Incorrect Default Permissions
[21]	CWE-918	Server-Side Request Forgery (SSRF)
[22]	CWE-362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')
[23]	CWE-400	Uncontrolled Resource Consumption
[24]	CWE-611	Improper Restriction of XML External Entity Reference
[25]	CWE-94	Improper Control of Generation of Code ('Code Injection')

2.2.3 Mapping of OWASP and CWE

According to this project, the detection tool will be able to detect and categorize the CWE Top 25 vulnerabilities into the 10 categories of OWASP Top 10 security risks group. However, the focus of this project is specifically on vulnerabilities related to source code category. With that, the mapping between these two sources is depicted in Table 2.3. The mapping involves the top security risks from both OWASP Top 10 2021 and CWE Top 25 2022.


Table 2.3 Mapping between OWASP 2021 and CWE SANS 2022

OWASP Rank 2021	OWASP Vulnerability	SANS CWE ID
A01	Broken Access Control	CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') CWE-276: Incorrect Default Permissions CWE-352: Cross-Site Request Forgery (CSRF) CWE-862: Missing Authorization
A02	Cryptographic Failures	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')
A03	Injection	CWE-20: Improper Input Validation CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection') CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

		<p>CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')</p> <p>CWE-94: Improper Control of Generation of Code ('Code Injection')</p>
A04	Insecure Design	CWE-434: Unrestricted Upload of File with Dangerous Type
A05	Security Misconfiguration	CWE-611: Improper Restriction of XML External Entity Reference
A06	Vulnerable and Outdated Components	CWE-190: Integer Overflow or Wraparound
A07	Identification and Authentication Failures	<p>CWE-287: Improper Authentication</p> <p>CWE-306: Missing Authentication for Critical Function</p> <p>CWE-798: Use of Hard-coded Credentials</p>
A08	Software and Data Integrity Failures	CWE-502: Deserialization of Untrusted Data
A09	Security Logging and Monitoring Failures	<p>CWE-400: Uncontrolled Resource Consumption</p> <p>CWE-416: Use After Free</p> <p>CWE-476: NULL Pointer Dereference</p>
A10	Server-Side Request Forgery	<p>CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer</p> <p>CWE-125: Out-of-bounds Read</p> <p>CWE-787: Out-of-bounds Write</p> <p>CWE-918: Server-Side Request Forgery (SSRF)</p>

2.3 Analysis of Existing Sources

The difference between OWASP and CWE is not that much as both articles aim to assist developers and security engineers in addressing faults that could result in greater harm to the application they are working on. Using both lists together is a better idea as the OWASP is more broad and the CWE (*CWE - Frequently Asked Questions (FAQ)*, n.d.) can narrow down a specific risk type within the broad topics of the OWASP top 10 for better mitigation. Both articles are easy to understand and provide suggestions on how to prioritize risks and take precautions against them.

However, there is a slight difference in terms of the details. OWASP Top 10 is the main category and the CWE is a breakdown of each vulnerability. Their scope and objectives also differ (*CWEs vs OWASP Top 10? - DEV Community* , n.d.). The OWASP groups the most common web application security weaknesses into ten categories that correspond to broader cybersecurity issues. The CWE, on the other hand, lists the most prevalent issues from the Common Weakness Enumeration. Thus, CWE differs from the OWASP list as it more focuses on specific weaknesses rather than more general categories. As a result, each item on the list pertains to actual implementation problems that can be identified and fixed, making it more directly useful for developers and security engineers.

In addition, CWE Top 25 covers all software categories, while the OWASP Top 10 only covers web applications (*What Is OWASP? What Is the OWASP Top 10? | Cloudflare*, n.d.). Thus, OWASP Top 10 covers more general concepts and is more focused on web applications, while CWE Top 25 covers a wider range of vulnerabilities than what is covered by OWASP Top 10. Therefore, these two sources can be trusted as they provide unbiased and practical information on security risks. These two significant sources of information have been crucial in mitigating dangerous attacks from happening all over the world and have provided developers with the greatest security practices.

2.4 Studies on Existing Systems

2.4.1 Acunetix

Acunetix is a web vulnerability scanner that automates the process of detecting and reporting security issues on web applications and servers (*Introduction to Acunetix | Acunetix*, n.d.). It is intended to assist organizations in identifying and resolving vulnerabilities in their websites and web-based applications, such as SQL injection, cross-site scripting (XSS), and other common vulnerabilities. Furthermore, it can check for vulnerabilities on both the network and web application levels.

The scanner can uncover both known and unknown vulnerabilities, with its built-in technology that can detect even zero-day vulnerabilities. It supports various authentication methods, including forms-based authentication, NTLM, and Kerberos, and has advanced false positive management to minimize false alerts. It also includes pre-configured compliance checks for standards such as OWASP and PCI-DSS (*C-YBER - What You Need to Know about Acunetix?*, n.d.).

Acunetix provides detailed reporting, including information on affected pages and inputs, as well as recommendations for remediation. It can also be integrated with other tools, such as bug trackers, to simplify the management of vulnerabilities. The software can be run on Windows, Linux, and macOS.

Acunetix is a commercial software. The company offers email and phone support, as well as a knowledge base and user community to help with any issues that may arise.

2.4.2 Nessus

Nessus is a widely used vulnerability scanner that can detect various types of security problems on multiple platforms like Windows, Linux, and macOS. It can scan for vulnerabilities on different levels like web applications, network, and cloud. The purpose of Nessus is to assist organizations in identifying and resolving vulnerabilities in their IT infrastructure, including servers, network devices, and cloud environments (*What Is NESSUS and How Does It Work? - ITperfection - Network Security*, n.d.).

Nessus can scan for known and unknown vulnerabilities and includes pre-configured compliance checks for standards like OWASP and PCI-DSS. It supports different authentication methods like forms-based authentication, NTLM, and Kerberos and has advanced false positive management to reduce false alerts.

The tool provides a detailed report that includes information about the specific vulnerabilities found and suggestions for resolving the issues. Additionally, it can be integrated with other tools such as bug trackers to simplify the process of managing vulnerabilities.

Nessus is a commercial software with pricing based on the number of IPs. It offers email and phone support, a knowledge base and user community to assist users with any issues they may encounter. It also offers various licensing options such as Professional, Nessus Manager and Nessus Cloud, to suit the organization's needs and the scale of the deployment.

2.4.3 Qualys

Qualys is a security and compliance solution that operates on cloud, it helps organizations detect and defend against cyber threats by providing a variety of services including vulnerability management, compliance, and threat protection. The platform adopts the Software-as-a-Service (SaaS) model which facilitates users to quickly and easily access the necessary tools to secure their networks and data (*Qualys Vulnerability Scanner* | *Bugcrowd*, n.d.).

A major feature of Qualys is its vulnerability management capabilities, it can scan for vulnerabilities on a wide range of systems including servers, network devices, and cloud environments. The platform offers an extensive library of vulnerability checks and can identify both known and unknown vulnerabilities.




Qualys also provides threat protection capabilities which enables organizations to detect and respond to cyber threats in real-time. The platform uses machine learning and AI to analyze network traffic and detect potential threats, enabling organizations to respond quickly to incidents and minimize the impact of cyber-attacks.

The platform also provides detailed reporting and analytics which assist organizations in identifying vulnerabilities and threats and monitoring their progress in resolving them. It also offers integration with other security tools like firewalls and intrusion detection systems to streamline the process of managing security across an organization.

Qualys is a commercial software, pricing is based on the number of assets being scanned with different licensing options available to suit the organization's needs. It offers 24/7 customer support, a knowledge base, and user community to assist users with any issues they may encounter.

2.5 Comparison on Existing Systems

Table 2.4 Comparison between all three existing system

Feature	Acunetix	Nessus	Qualys
Logo			
Platform	Windows, Linux, macOS	Windows, Linux, macOS	Cloud-based
Scan Types	Web Application, Network	Network, Web Application, Cloud, Containers, IoT	Web Application, Network, Cloud
Vulnerability Detection	SQL Injection, XSS, XXE, File inclusion, and others	Comprehensive coverage of vulnerabilities	Comprehensive coverage of vulnerabilities
Reporting	Detailed reports with remediation advice	Customizable reports	Detailed reports with remediation advice
Integration	Integrates with bug tracking systems	Integrates with other security tools	Integrates with other security tools
Pricing	Commercial	Commercial, pricing based on number of IPs	Subscription-based, pricing based on number of IPs
Support	Offers email and phone support, as well as a knowledge base and user community	Offers email and phone support, as well as a knowledge base and user community	Offers email and phone support, as well as a knowledge base and user community

2.6 Summary

Based on the review of these two articles, it shows that the sources are the most popular and can be trusted. All the requirements gathered through this literature review will be practiced throughout the development process of the detection tool. In order to make sure the successful outcome of doing this vulnerability detection tool, all requirements must be fulfilled. The results and findings that have been identified will become the guideline during the development process.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter provides a detailed explanation of the approach and steps taken throughout the development of the proposed system, XPOSED. The gathering procedure used in collecting data on user requirements, system requirements, and the principle used in each stage of the chosen model will be explained. The activity involved in each principle will be explained in more detail from start until the completion of the project. All the principles applied are to be shown in the Software Development Life Cycle (SDLC). The flowchart will describe the flow of work. A Gantt chart will also be shown as it shows the track of development activities from start to finish. Besides, the design of the proposed system, interface design, and database design has also been made. For planning and implementation, the User Acceptance Test (UAT) form will be produced according to the system requirements. This chapter would be really important as it shows each process and steps to complete the project. Hence, a suitable methodology has been chosen to apply in the development of the XPOSED system.

3.2 Software Development Life Cycle (SDLC)

After some research on the suitable Software Development Life Cycle (SDLC) model, the RAD or Rapid Application Development approach has been chosen as it seems suitable for the development of this XPOSED system. Based on Figure 3.1, the RAD model is chosen based on the nature of this project and the methods to be used, as well as the deliverables that are required to develop the system. The detection tool system focuses on delivering the full functionality of the system and not just fulfilling all the user requirements. The function of the system might be changed based on the user requirements from time to time. This adaptable approach ensures active user involvement

throughout development, reducing the risk of non-conformance with user requirements and saving time and resources in the process.

Hence, designing and developing the system needs to be done first, only then will be shown to the user. If there are any unclear requirements after getting feedback from user, the developer will rework that particular problem. This iteration will be repeated until fulfilling all the requirements. With that, there will be a total of 4 stages in this RAD model which are Requirements Planning, User Design, Construction, and Cutover. Every single stage is to be explained more specifically and each stage characteristics will be pointed out.

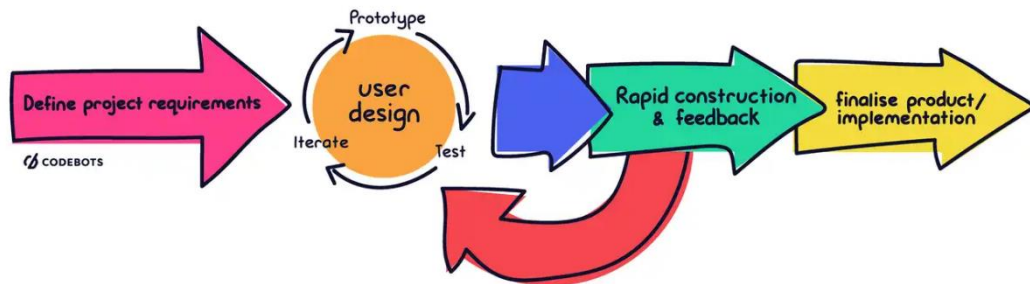


Figure 3.1 Rapid Application Development Process

3.2.1 Phase 1 – REQUIREMENTS PLANNING

The Requirements Planning stage involves the discussion of the important aspects such as the project name and the desired functionalities of the system. This includes identifying both functional requirements (features and capabilities the system should have) and non-functional requirements (performance, security, and usability criteria). After several meetings with the supervisor, a project name XPOSED Vulnerability Detection System is produced. Next, proceed with studies on existing trusted sources on top-listed security risks that relate to this project. The existing sources that are being chosen are OWASP Top 10 and CWE SANS Top 25. Each source has its uniqueness of deliverables on the information related to security risks.

3.2.2 Phase 2 – USER DESIGN

At this this stage, it involves the design of user interface of the developed system. The overall design of the system needs to be suitable for the project and the functions involved must be stated clearly. The interface design should contain a suitable layout used, appropriate colour used, and readable font size as well as responsiveness for both online and mobile views. All these elements are preferably maintained in the same position on all pages to practice the consistency concept in designing a responsive application suitable for all different sizes of display. Besides, the position where the information is being displayed is important as it tells the user what the system is all about.

3.2.3 Phase 3 – CONSTRUCTION

After the previous stage, the construction phase begins. This phase involves the actual construction of the system and the creation of the database according to the design developed in the earlier design phase. The user interface design is also implemented, incorporating the features identified in the previous stage. The main objective of the construction stage is to build a robust and efficient system that can effectively detect and analyze various types of software vulnerabilities.

3.2.4 Phase 4 – CUTOVER

The cutover stage is an important step when the developed system is ready to be put into action. At this stage, the team thoroughly tests the system, making sure to fix any bugs or issues that are found during testing. Once the system is stable and reliable, it is deployed to the intended users. The team provides necessary training and support to help users smoothly transition and start using the system effectively. Ongoing maintenance and updates are also part of the cutover stage to ensure the system stays up to date with the latest security practices and is protected against emerging vulnerabilities.

3.3 Project Requirement

The XPOSED project requirements are summarised in this section. The nature of this system, including its functional and non-functional requirements, is depicted in Table 3.1 below.

Table 3.1 Project requirements for XPOSED system

Background	XPOSED Vulnerability Detection System objective is to manage the detection of vulnerabilities that exist in applications systematically. For detection, the system focuses on the specific programming language of a few selected vulnerabilities. It focuses on the vulnerability caused by a source code attack.
Project Name	XPOSED Vulnerability Detection System
Objective/Vision	System needs to be able to handle the detection and provide appropriate solutions.
Users of the system	Individuals who want to find vulnerabilities in their application.
Functional Requirements	<ul style="list-style-type: none"> • The system should have the capability to identify vulnerabilities in source code. • The system should be able to classify risks based on the OWASP Top 10 categories. • Users should be able to upload source code files for vulnerability detection. • The system should provide a user-friendly interface to view reports and visualize analytics related to the vulnerabilities detected. • The system should allow admin to manage user accounts. • The system should allow admin to download comprehensive reports summarizing the scanning activities performed by all users. • The system should support essential functionalities such as adding, editing, and deleting different types of risk categories.

Non-functional Requirements	<ul style="list-style-type: none">• Support concurrent connections from multiple users.• Robust database design to handle a large number of users effectively.• Ensure acceptable response time for system operations.• Implement easy backup and recovery mechanisms for user-supplied information.• Develop a web-based platform for the XPOSED Vulnerability Detection System.• Limit the number of concurrent users to a few hundred to maintain optimal system performance and prevent lagging or delayed response times.• Require an active internet connection for accessing the XPOSED system to facilitate efficient data transfer between the server and users.• Restrict users to a single account per person to prevent duplication and simplify user tracking.
-----------------------------	--

3.4 Propose Design

3.4.1 Flowchart

Figure 3.2 illustrates the general step-by-step process for finding the security risks of the uploaded source code file. The flowchart visually displays the different decision points and paths that can be taken throughout the process, such as creating new scan, updating existing scan record, viewing results, and generating reports. The flowchart helps to clearly understand the flow of information and the interactions between different components of the system.

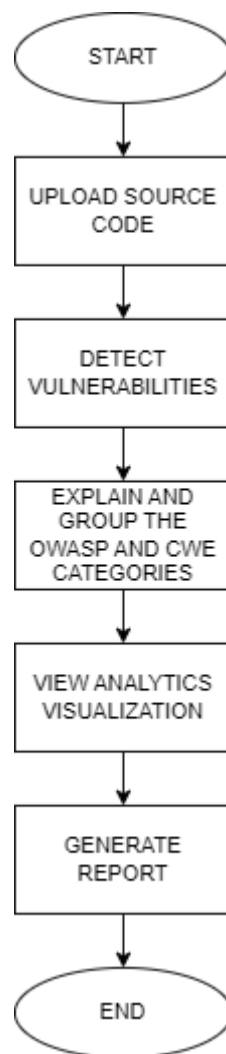


Figure 3.2 General workflow of XPOSED system

The flow of a user in the XPOSED system is shown in Figure 3.3. The user must first log in to the system and then has the option to initiate a new scan or review their previous scan activities. To initiate a new scan, the user needs to upload a zip file containing the source code into the system. After starting the scanning process, the user is required to wait for a notification from the system confirming the completion of the scan. They can then view the results, which are displayed in an analytics view and show any detected vulnerabilities in the source code, categorized according to OWASP Top 10 categories. These vulnerabilities are further broken down by specific security risk names and detailed explanations of the vulnerability type according to CWE SANS Top 25. Finally, the user can export the results as desired.

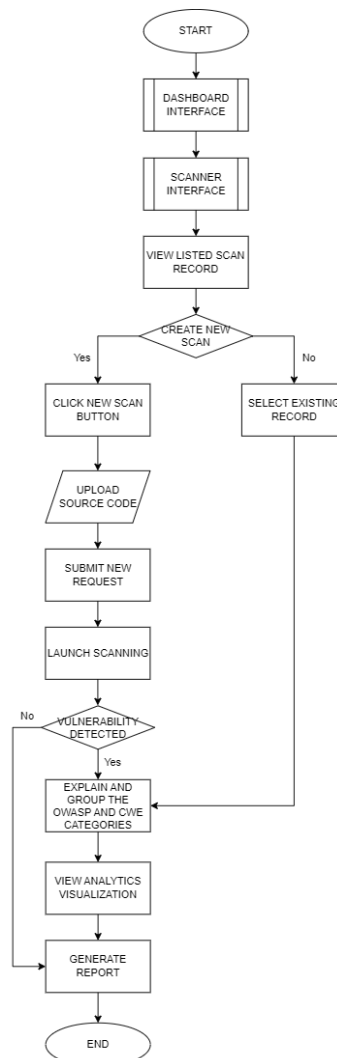


Figure 3.3 Detailed workflow of End Users

In Figure 3.4, the system flow for Admin is depicted. The main focus of an admin's tasks within the XPOSED system is twofold: managing the list of risk categories and handling user data. This process commences with the admin's login to the system. By accessing the analytics dashboard, the admin can obtain informative statistics regarding various categories. These statistics include the total number of users, total vulnerabilities, and the number of existing categories. The manage user page allows the admin to view a list of registered users. Moreover, the admin has the capability to generate reports concerning scan analysis and user information. This enables the identification of patterns and trends in user activity.

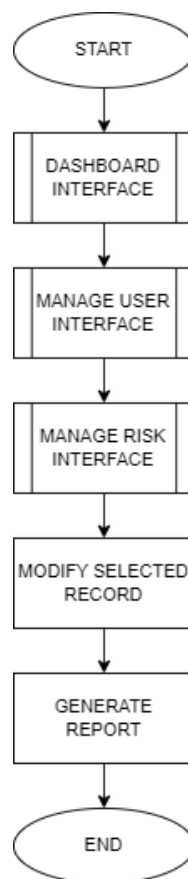


Figure 3.4 Detailed workflow of Administrator

3.4.2 Context Diagram

The context diagram in Figure 3.5 provides an overview of the XPOSED system and how it interacts with external entities such as end users or administrators. In this system, end users and administrators are the entities that interact with the system. The data flow between the entities and the system indicates the exchange of information. The diagram shows that all users are required to log in to the system in order to use it. Users have the ability to upload source code, initiate a scan, view the results of the scan, and generate reports. The administrator can manage the risk information and generate reports based on the user activity, as well as identifying trends and patterns.

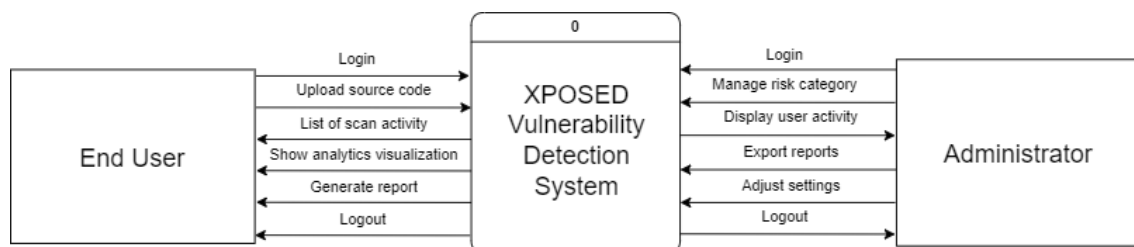


Figure 3.5 Context diagram of XPOSED system

3.4.3 Use Case Diagram

The use case diagram is a visual representation of the XPOSED system's behaviour and functionality as it responds to requests made by different actors, such as User and Admin. The Figure 3.6 shows how these actors interact with the system and the specific actions they are able to perform. The User actors are required to log in to the system before they can use it and are able to log out once they are finished. On the other hand, Admin actors have the capability to oversee risk categories and generate reports containing relevant information about scans and user data. One behaviour that is common to both actors is the ability to generate reports based on their own representations of the system. This allows both User and Admin actors to gain insights and understand how the system is being used. Besides, Table 3.2 shows the use case functionality for XPOSED system.

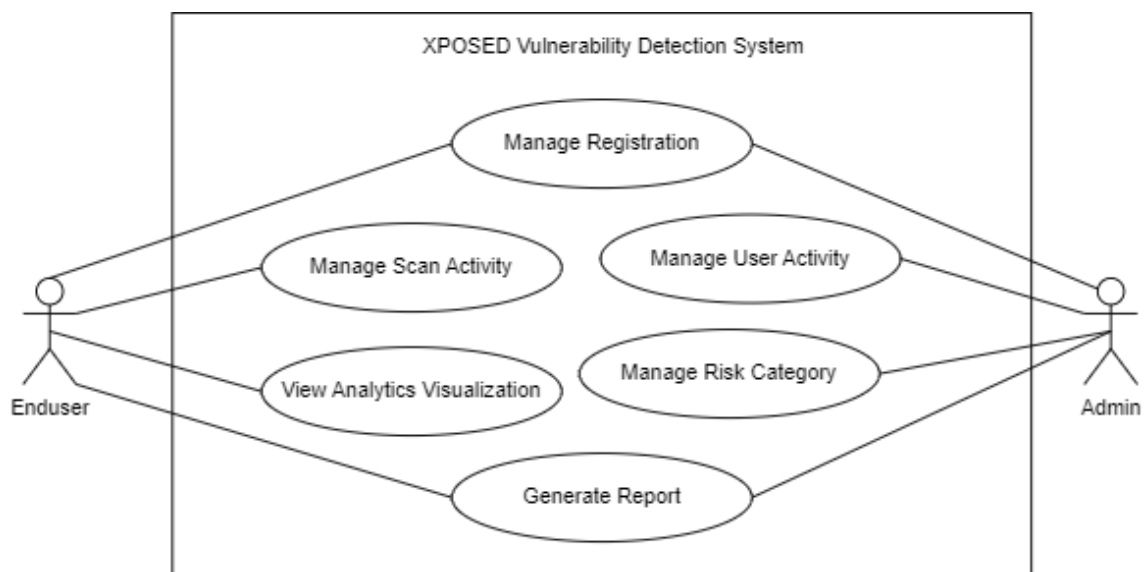
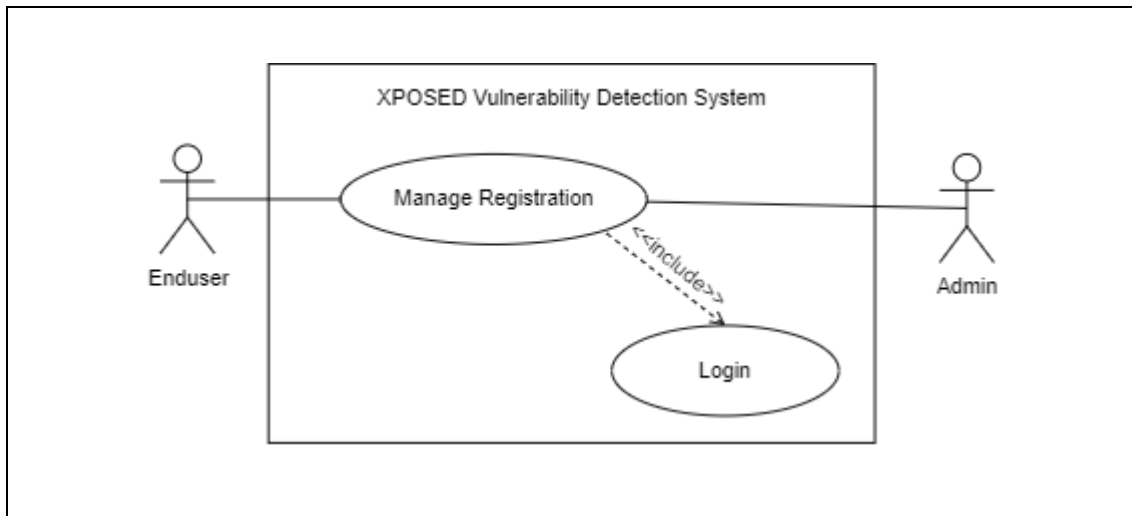


Figure 3.6 Use case diagram of XPOSED system

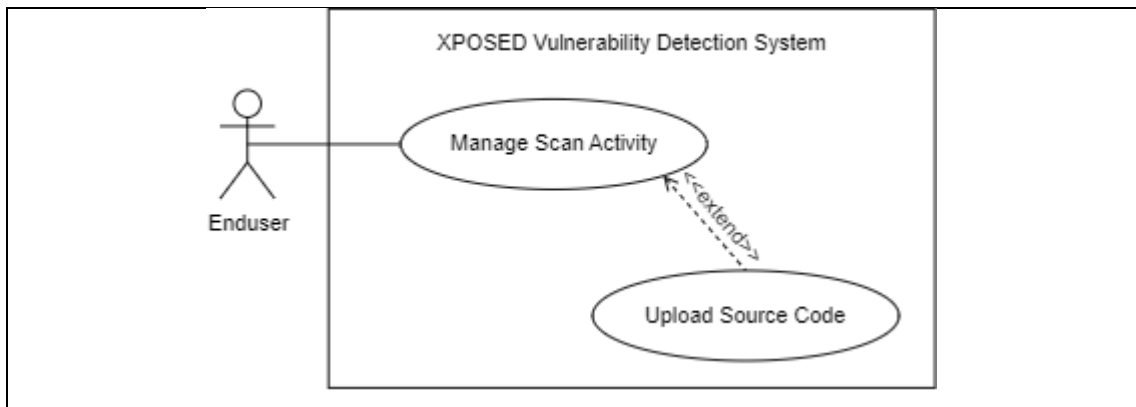
Table 3.2

Use case description of XPOSED system



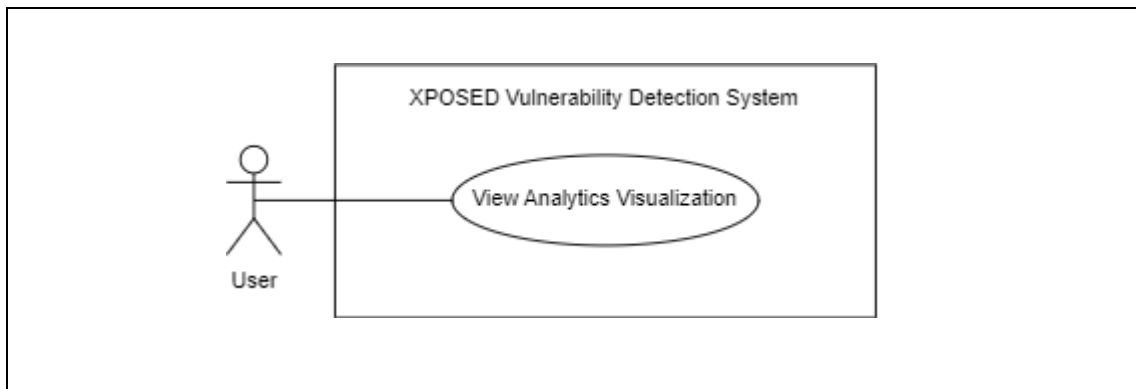
Use Case ID	XPS-UCD-100 (Manage Registration)
Brief Description	This use case involves Endusers creating new accounts and both Endusers and Admin logging in with existing accounts.
Actor	Enduser and Admin
Pre-Conditions	Endusers must register an account prior to accessing the system.
Basic Flow	<p>[Enduser]</p> <ol style="list-style-type: none"> 1. The use case begins when the system displays the Registration page. 2. User enters username, email address, and password for account creation. 3. System validates the format of the entered details. [C1: Password length] 4. User clicks <<Create Account>> button. 5. System validates the input against existing accounts. [E1: User already exist] 6. System redirects to Login page, indicating successful registration. 7. User enters registered email address and password. 8. User clicks <<Login>> button. [E2: Invalid

	<p>credentials]</p> <p>9. The use case ends.</p> <p>[Admin]</p> <ol style="list-style-type: none"> 1. The use case begins when the system displays the Login page. 2. Admin enters username and email address. 3. Admin clicks the <<Login >> button. 4. The use case ends.
Exception Flow	<p>[E1: User already exist]</p> <ol style="list-style-type: none"> 1. System shows message “User already existed”. 2. Use case continues at Step 2 in Basic Flow (Enduser). <p>[E2: Invalid credentials]</p> <ol style="list-style-type: none"> 1. System shows message “Username or email is not matched”. 2. Use case continues at Basic Flow Step 2 for Enduser.
Post-Conditions	User successfully logs in or registers, gaining access to their account and functionalities.
Rules	Not Applicable
Constraints	<p>[C1: Password length]</p> <p>Password length should be in between 10 to 30 alphabets.</p>

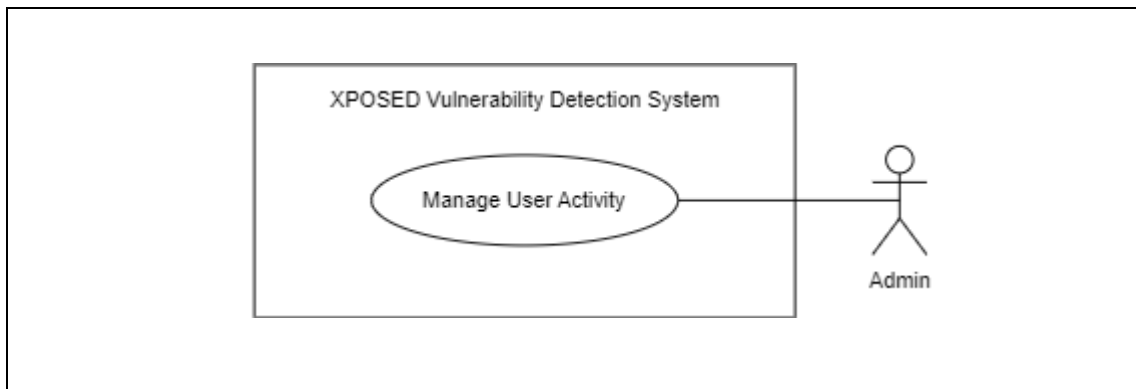


Use Case ID	XPS-UCD-200 (Manage Scan Activity)
Brief Description	This use case enables the Enduser to view, create, and delete their scan activity.
Actor	Enduser
Pre-Conditions	Enduser is logged in.
Basic Flow	<ol style="list-style-type: none"> 1. The use case begins when the system displays the My Scan page. 2. System displays all the existing scan list with the information regarding that scan. 3. From the My Scan page, the Enduser is able to: <ol style="list-style-type: none"> i. Create a new scan. [A1: Add New Scan] ii. Export report of existing scan. [A2: Download Scan] iii. Delete old scan. [A3: Delete Scan] 4. The use case ends.
Alternative Flow	<p>[A1: Add New Scan]</p> <ol style="list-style-type: none"> 1. Clicks <<Add New Scan>> button. 2. Enter scan name and scan description. 3. Upload zip file. [E1: Wrong file format] 4. Click <<Scan>> button to submit. 5. System verifies and processes the uploaded file. 6. System shows loader to indicate the progress of scanning process. 7. The system saves the data to the database. 8. System shows analytics report of the scan activity.

	<p>9. The use case continues at Step 4 in Basic Flow.</p> <p>[A2: Download Scan]</p> <ol style="list-style-type: none"> 1. Clicks the download icon which parallel to the selected record. 2. System process the download request. 3. The use case continues at Step 4 in Basic Flow. <p>[A3: Delete Scan]</p> <ol style="list-style-type: none"> 1. Clicks the trash icon parallel to the selected record. 2. System pop-up message to confirm the delete action. 3. User clicks <<Delete>> button. 4. System displays successful message. 5. The use case continues at Step 4 in Basic Flow.
Exception Flow	<p>[E1: Wrong file format]</p> <ol style="list-style-type: none"> 1. System shows message “Only .zip type accepted”. 2. The use case continues at Step 3 in Basic Flow.
Post-Conditions	Source code file is successfully uploaded and ready for vulnerability scanning.
Rules	Not Applicable
Constraints	Not Applicable



Use Case ID	XPS-UCD-300 (View Analytics Visualization)
Brief Description	This use case involves Endusers to access analytics data for the identified vulnerabilities in their scan.
Actor	Enduser
Pre-Conditions	Endusers have initiated a scan and there are vulnerabilities identified in the system.
Basic Flow	<p>[Enduser]</p> <ol style="list-style-type: none"> 1. The use case begins when the system displays the scan detail page. 2. The user selects one of the records they want to explore further. 3. The system presents the selected record's information along with relevant data analytics visualizations. 4. The user views and analyzes the displayed information and data visualizations. 5. The use case ends.
Exception Flow	[None]
Post-Conditions	User has successfully viewed and analyzed the selected record's information and data analytics visualization.
Rules	Not Applicable
Constraints	Not Applicable



Use Case ID	XPS-UCD-400 (Manage User Activity)
Brief Description	This use case involves Admin to monitor and review all user activities related to their scan.
Actor	Admin
Pre-Conditions	Admin must be logged into the system.
Basic Flow	<p>[Admin]</p> <ol style="list-style-type: none"> 1. The use case starts when the system presents the user movement page. 2. The admin chooses a specific record for further exploration. 3. The admin verifies the accuracy of the displayed report. 4. The use case ends.
Exception Flow	[None]
Post-Conditions	Any necessary actions or decisions based on the report can be taken.
Rules	Not Applicable
Constraints	Not Applicable

<p>XPOSED Vulnerability Detection System</p> <p>Manage Risk Category</p> <p>Admin</p>	
Use Case ID	XPS-UCD-500 (Manage Risk Category)
Brief Description	This use case enables the Admin to add, edit, and delete risk category.
Actor	Admin
Pre-Conditions	Admin is logged in.
Basic Flow	<ol style="list-style-type: none"> 1. The use case begins when the system displays the Add New Word page. 2. System displays all the existing risk list with the information regarding that risk. 3. From this page, admin able to: <ol style="list-style-type: none"> i. Add a new word for the risk. [A1: Add New Risk] ii. Edit any of existing risk record. [A2: Edit Risk] iii. Delete unwanted risk type. [A3: Delete Risk] 4. The use case ends.
Alternative Flow	<p>[A1: Add New Scan]</p> <ol style="list-style-type: none"> 1. Clicks <<Add New Word>> button 2. System displays a modal form to be insert with several details regarding the risk. 3. Click <<Save>> to submit the risk. 4. The system saves the data into the database. 5. The use case continues at Step 4 in Basic Flow. <p>[A2: Edit Risk]</p> <ol style="list-style-type: none"> 1. Clicks the <<Edit>> button. 2. Edit the risk form modal.

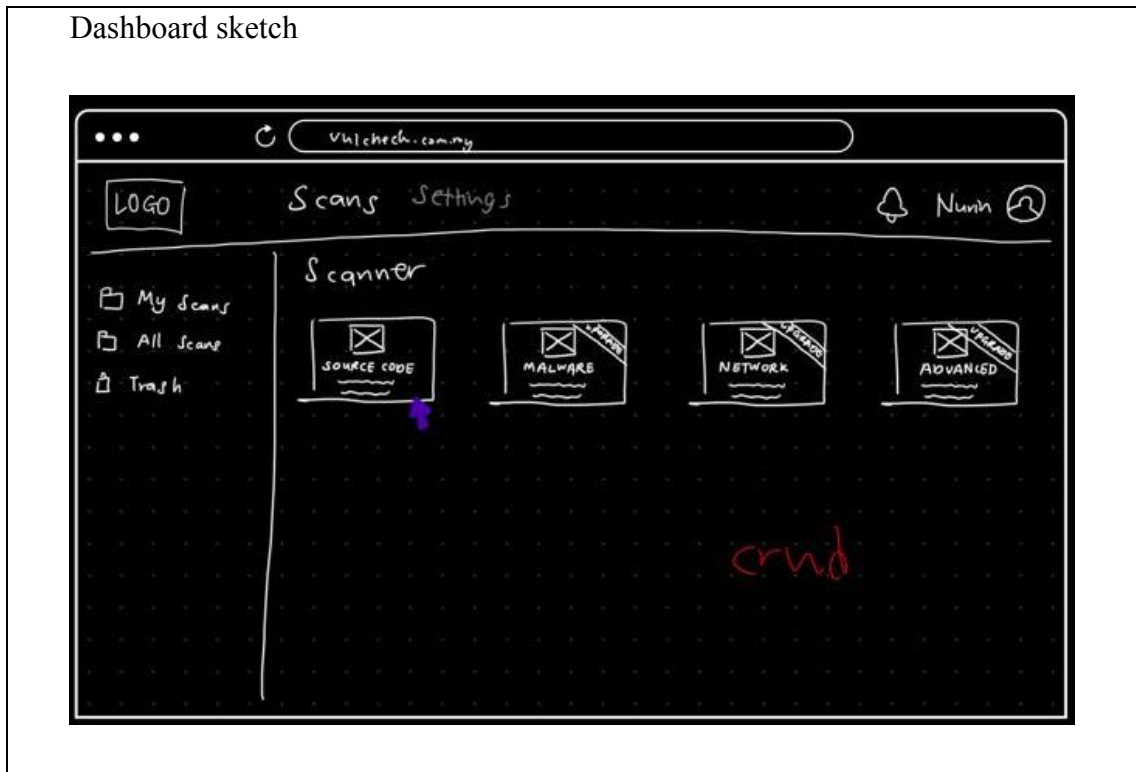
	<ol style="list-style-type: none"> 3. Clicks <<Save>> button. 4. System saves the data to the database. 5. System displays successful message. 6. The use case continues at Step 4 in Basic Flow. <p>[A3: Delete Risk]</p> <ol style="list-style-type: none"> 1. Clicks <<Delete>> button to delete any of the unwanted risk. 2. System pop-up message to confirm the delete action. 3. User clicks <<Delete>> button. 4. System deletes the data from database. 5. System displays successful message. 6. The use case continues at Step 4 in Basic Flow.
Exception Flow	[None]
Post-Conditions	Risks are successfully added, edited, or deleted within the system as per the admin's actions.
Rules	Not Applicable
Constraints	Not Applicable

Use Case ID	XPS-UCD-600 (Generate Report)
Brief Description	This use case enables the Enduser and Admin to generate a comprehensive report based on the scan data.
Actor	Enduser and Admin
Pre-Conditions	The presence of data from the scanning process.
Basic Flow	<p>[Enduser]</p> <ol style="list-style-type: none"> 1. The use case starts when the system shows the scan detail page. 2. The system displays the scan activity report. 3. The user selects the <<Export>> button. 4. The system processes the request and shows the report in PDF format. 5. The user saves the file. 6. The use case ends. <p>[Admin]</p> <ol style="list-style-type: none"> 1. The use case begins when the admin navigates to the Manage Settings menu. 2. The system redirects to the corresponding page, providing access to various settings. 3. Admin clicks on the Export menu tab. 4. The admin triggers the export functionality by clicking the designated <<Export>> button. 5. The system promptly processes the export request and generates a detailed report in PDF format. 6. The user is then able to save the PDF file for future reference or further analysis. 7. The use case ends.

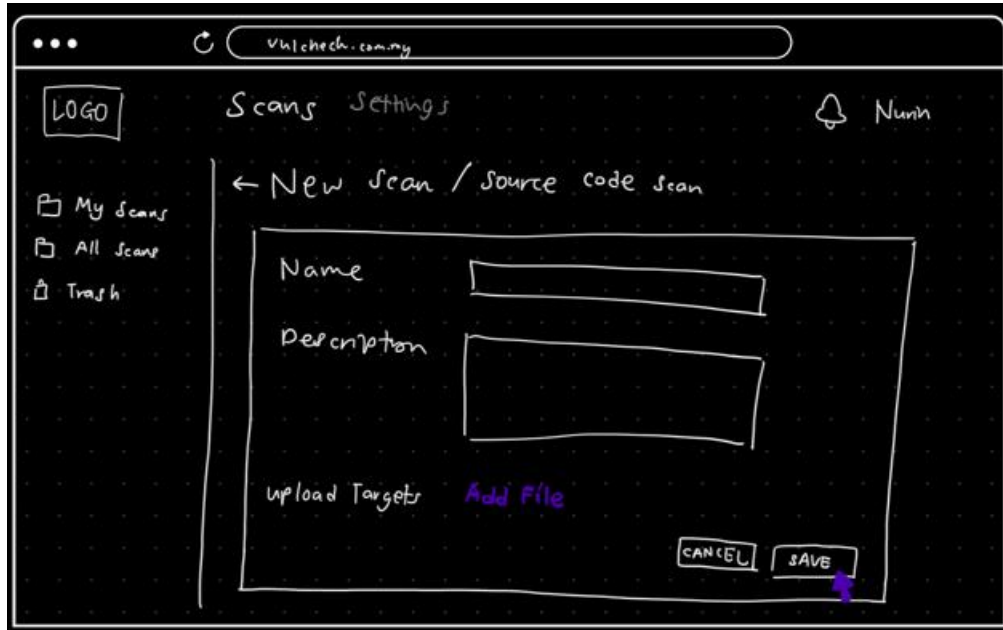
Alternative Flow	[None]
Exception Flow	[None]
Post-Conditions	Admin successfully exports and saves the report in PDF format.
Rules	Not Applicable
Constraints	Not Applicable

3.4.4 Storyboard

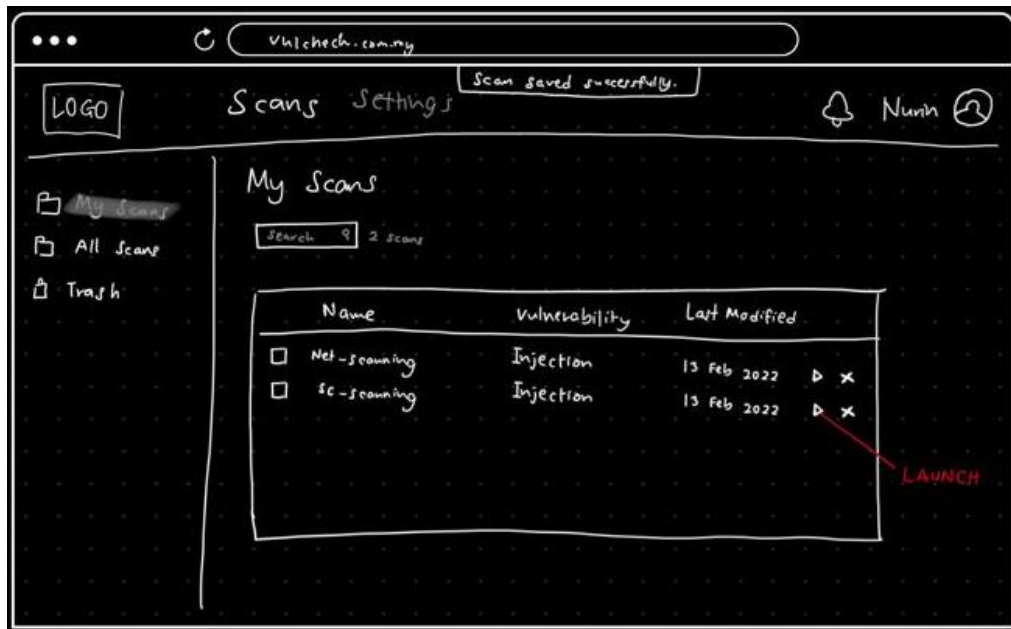
The storyboard illustrates the sketched user interface for the XPOSED system. The storyboard includes a visual representation of the different screens and how they will be navigated by the user.



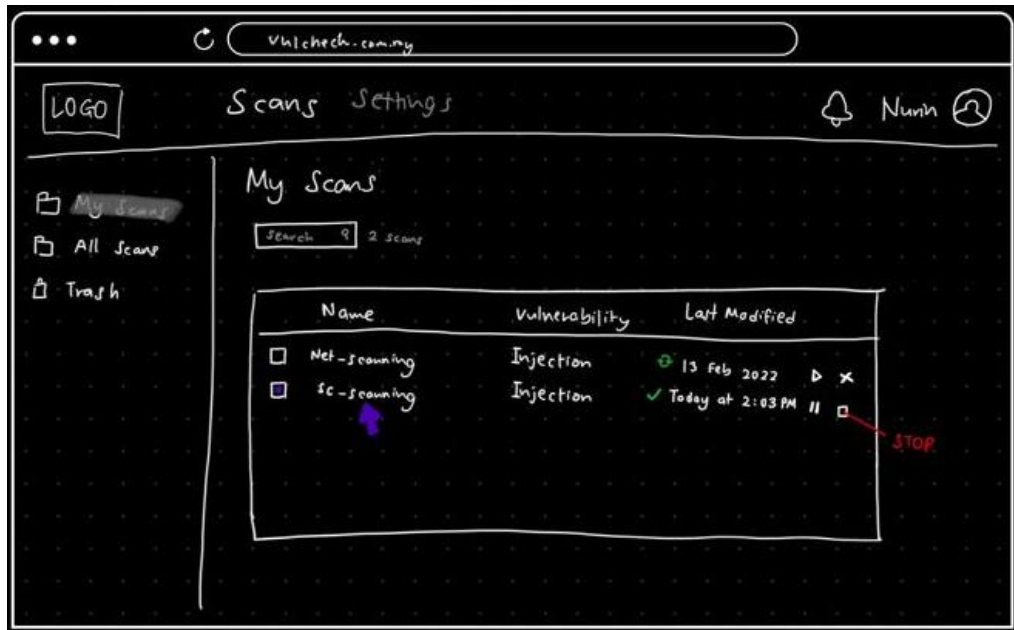
Add new scan sketch



Scan lists sketch



Re-launch existing scan sketch



3.5 Data Design

3.5.1 Entity Relationship Diagram (ERD)

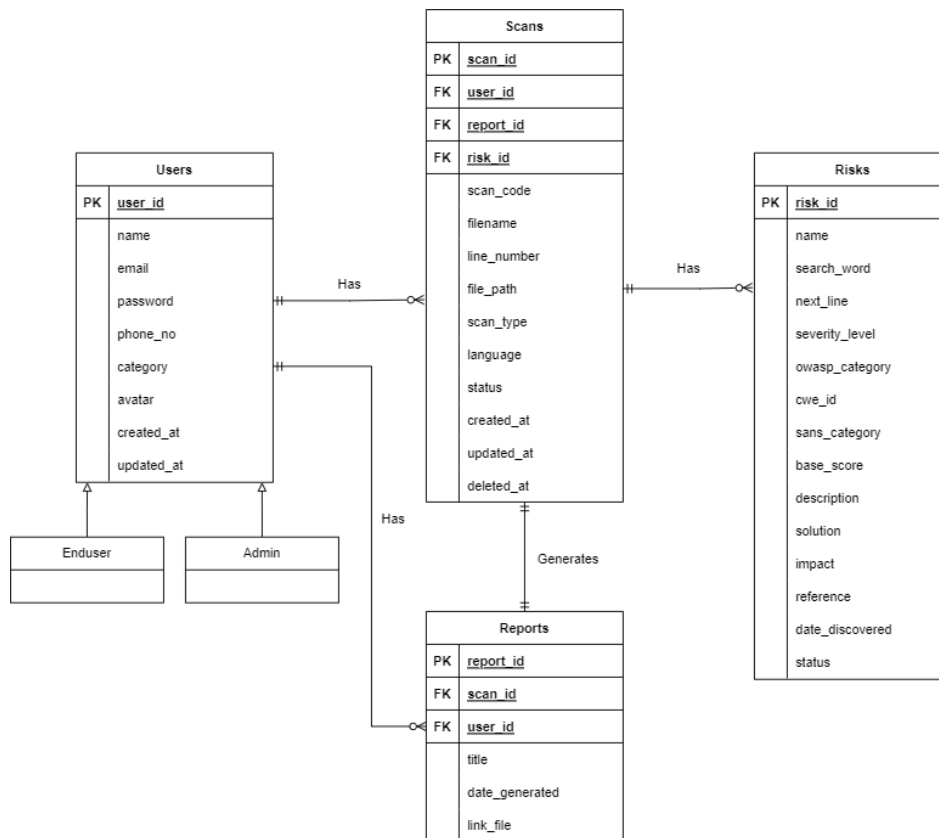


Figure 3.7 ERD for XPOSED system

The business rule for this XPOSED system is:

- 1) A vulnerability can be detected on multiple files (one-to-many).
- 2) One file can have multiple vulnerabilities (one-to-many).
- 3) A scan can detect multiple vulnerabilities (one-to-many).
- 4) A user can have many scanning activities.
- 5) A user can run one scanning at a time.
- 6) Each scanning is limited to having a single report at any given time.
- 7) Users can view the results of the scanning process as well as view the reports.
- 8) Many scanning projects can belong to one user.

3.5.2 Data Dictionary

USERS TABLE

Field Name	Data Type	Purpose	Constraints
user_id	Integer	A unique identifier for each user in the system.	PRIMARY KEY
name	String	A unique username chosen by the user.	
email	String	The email address associated with the user's account.	
password	String	The password chosen by the user for their account. (Hashed or Encrypted)	
phone_no	String	The user's phone number.	
category	String	The role of the user in the system (e.g. administrator, user, etc.).	
avatar	String	The image that represents a user's profile	
created_at	Date	The date the user registered for the system.	
updated_at	Date	The system login date of the user.	

RISKS TABLE

Field Name	Data Type	Purpose	Constraints
risk_id	Integer	Unique identifier for each vulnerability.	PRIMARY KEY
name	String	Name of the vulnerability.	
search_word	String	Refer to a keyword or phrase that is used in a search operation.	
next_line	String	Text or word present in the next line of a source code.	
severity_level	Integer	Severity level of the vulnerability. (e.g. high, medium, low)	
owasp_category	String	The OWASP Top 10 category the vulnerability belongs to, if applicable.	
cwe_id	String	The Common Weakness Enumeration (CWE) ID associated with the vulnerability.	
sans_category	String	The SANS Top 25 category the vulnerability belongs to, if applicable.	
base_score	Float	The Common Vulnerability Scoring System (CVSS) v3 base score for the vulnerability.	

description	String	A brief description of the vulnerability and how it can be exploited.	
solution	String	Suggested solution to mitigate the vulnerability.	
impact	String	The potential impact of the vulnerability on the system and data.	
reference	String	Link for more information about the vulnerability.	
date_discovered	Date	The date the vulnerability was discovered.	
status	String	The current status of the vulnerability (e.g. unpatched, patched, under investigation, etc.).	

SCANS TABLE

Field Name	Data Type	Purpose	Constraints
scan_id	Integer	A unique identifier for each scan in the system.	PRIMARY KEY
user_id	Integer	A unique identifier for the user who initiated the scan.	FOREIGN KEY
report_id	String	A unique identifier assigned to a specific report.	FOREIGN KEY
risk_id	String	A unique identifier assigned to a specific risk.	FOREIGN KEY
scan_code	String	A numeric and alphabet code associated with a specific scan.	
filename	String	The name of uploaded file.	
line_number	String	The line numbers where the vulnerability is located.	
file_path	String	The specific location or address of a file.	
scan_type	String	The type of scan performed (e.g. full, partial).	
language	String	The programming language used.	
status	String	The current status of the scan (e.g. completed, in progress, etc.).	
deleted_at	Datetime	The point in time when the deletion operation occurred.	

REPORTS TABLE

Field Name	Data Type	Purpose	Constraints
report_id	Integer	A unique identifier for each report in the system.	PRIMARY KEY
scan_id	Integer	The ID of the scan the report is associated with.	FOREIGN KEY
user_id	Integer	A unique identifier for the user who generated the report.	FOREIGN KEY
title	String	The name of report (e.g. vulnerability report, penetration test report, etc.)	
date_generated	Date Time	The date and time the report was generated.	
link_file	String	Link to the report file.	

3.6 Proof of Initial Concept

3.6.1 Dashboard Interface

The dashboard displayed in Figure 3.8 within the XPOSED system offers a clear and organized presentation of vital information. It specifically highlights the current severity level of security risks, which are categorized as Low, Medium, or High.

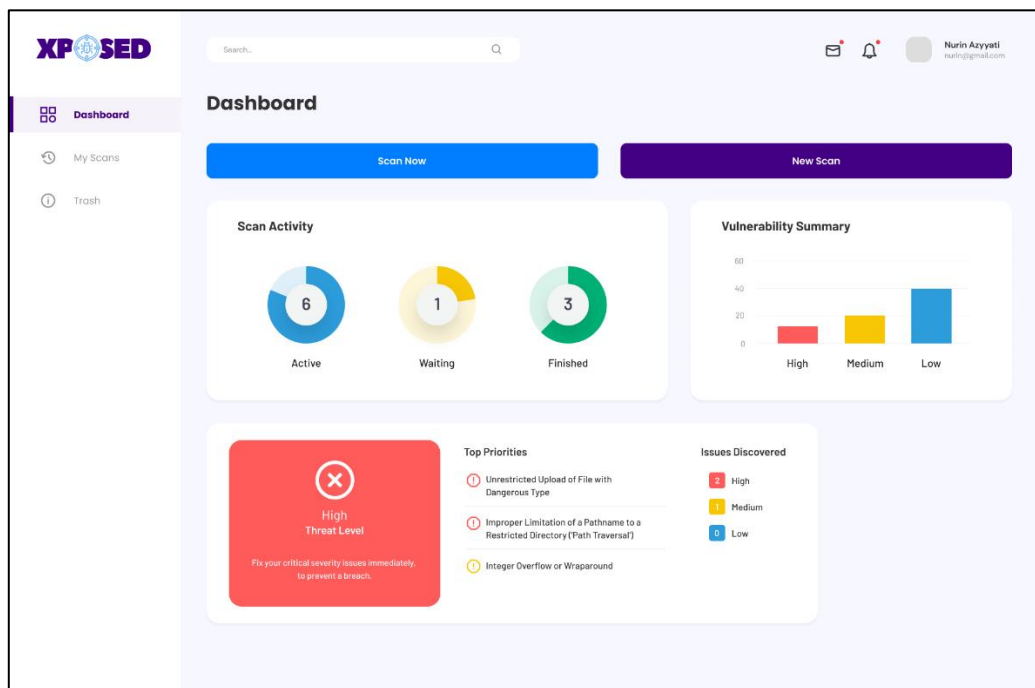


Figure 3.8 Dashboard interface of XPOSED

3.6.2 Add New Scan Interface

The interface displayed in Figure 3.9 is designed to facilitate the scheduling and initiation of new scans. Users can input all necessary details before starting the scanning process, making it easy to create a new scan record.

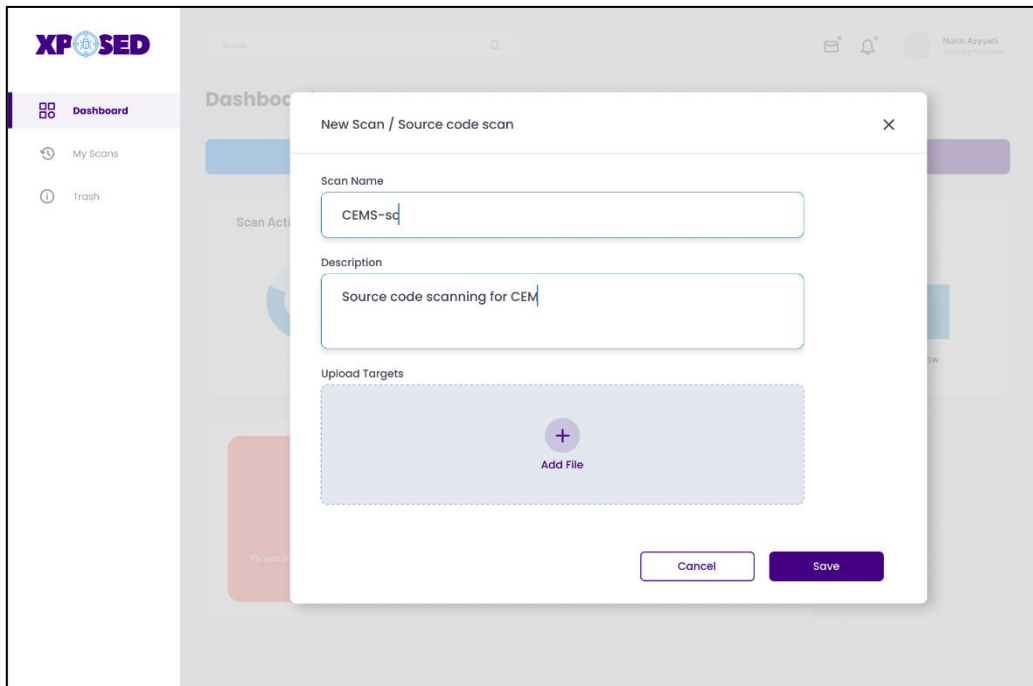
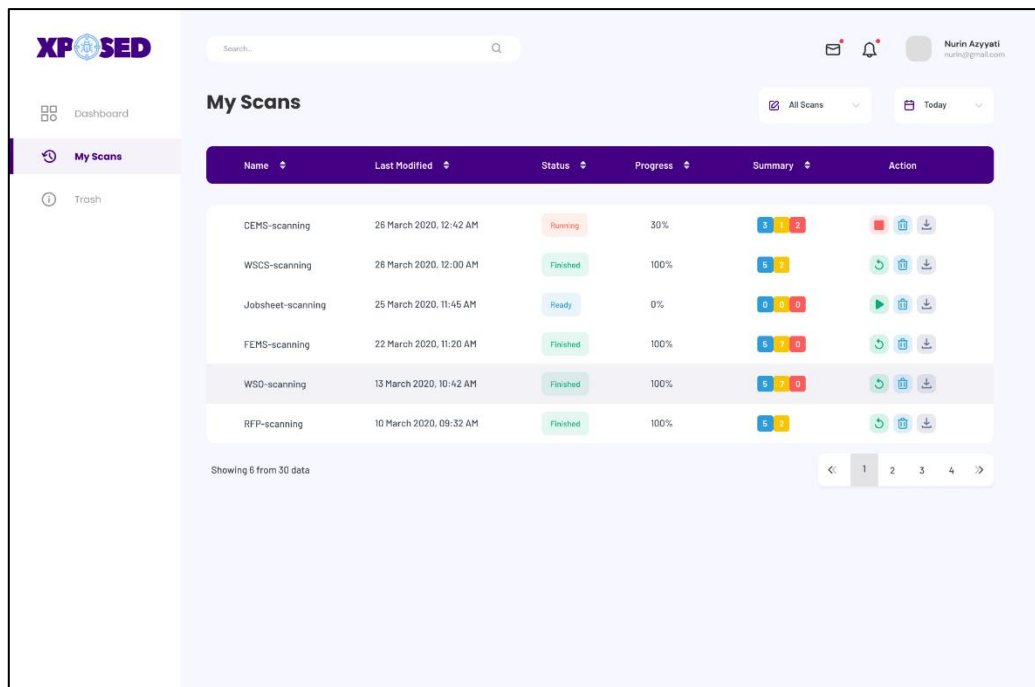


Figure 3.9 Add New Scan interface of XPOSED

3.6.3 Scan List Interface

The interface illustrated in Figure 3.10 is designed to offer users a comprehensive overview of all the scan records that have been executed within the system. Users can sort and filter the records using various criteria such as the latest scan, name of the scan, and date range.



The screenshot displays the 'My Scans' interface of XPOSED. It features a sidebar with navigation options: Dashboard, My Scans (selected), and Trash. The main content area shows a table of scan records with the following data:

Name	Last Modified	Status	Progress	Summary	Action
DEMS-scanning	26 March 2020, 12:42 AM	Running	30%	5 1 2	[Stop] [Refresh] [Download]
WSDS-scanning	26 March 2020, 12:00 AM	Finished	100%	5 1	[Refresh] [Download]
Jobsheet-scanning	25 March 2020, 11:45 AM	Ready	0%	0 0 0	[Play] [Download]
FEMS-scanning	22 March 2020, 11:20 AM	Finished	100%	5 7 0	[Refresh] [Download]
WSD-scanning	13 March 2020, 10:42 AM	Finished	100%	5 9 0	[Refresh] [Download]
RFP-scanning	10 March 2020, 09:32 AM	Finished	100%	5 1	[Refresh] [Download]

At the bottom of the table, it indicates 'Showing 6 from 30 data' and includes a pagination control showing page 1 of 4.

Figure 3.10 Scan Lists interface of XPOSED

3.6.4 Scan Details Interface

As depicted in Figure 3.11, the interface displays the outcome of the scanning process. It shows a list of all the vulnerabilities discovered in the uploaded project file. Users can gain a more detailed understanding of the security risks by clicking the "INFO" button.

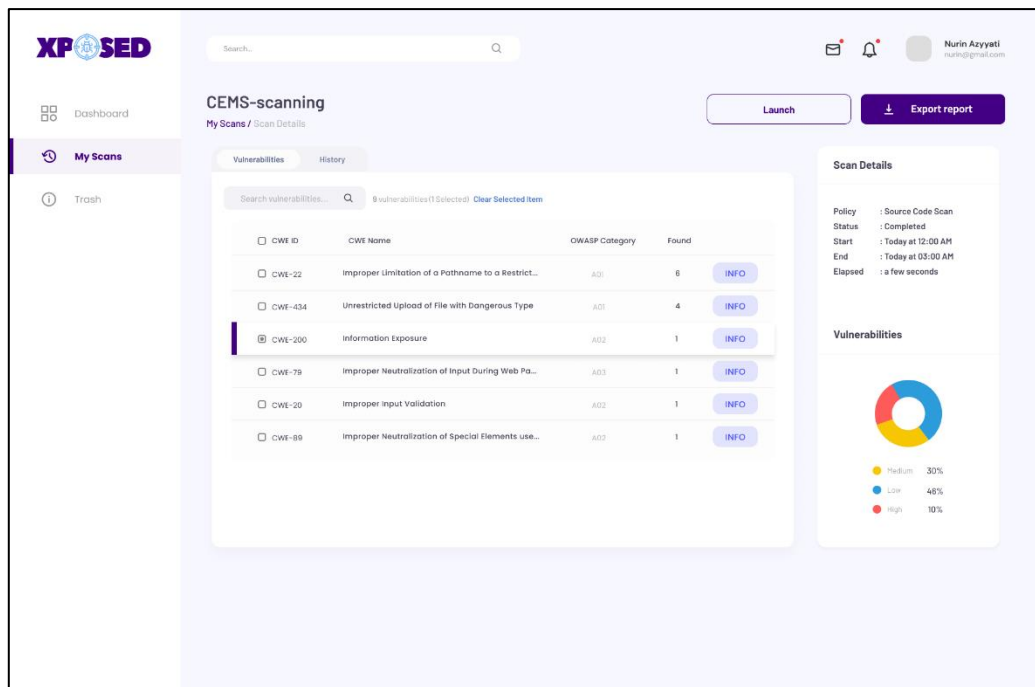


Figure 3.11 Scan Details interfaces of XPOSED

3.6.5 Scan Info Interface

Figure 3.12 offers users in-depth information about a selected vulnerability. It provides information such as the name of the vulnerability, a description of the vulnerability, the level of risk associated with it, and the solution to prevent it from happening.

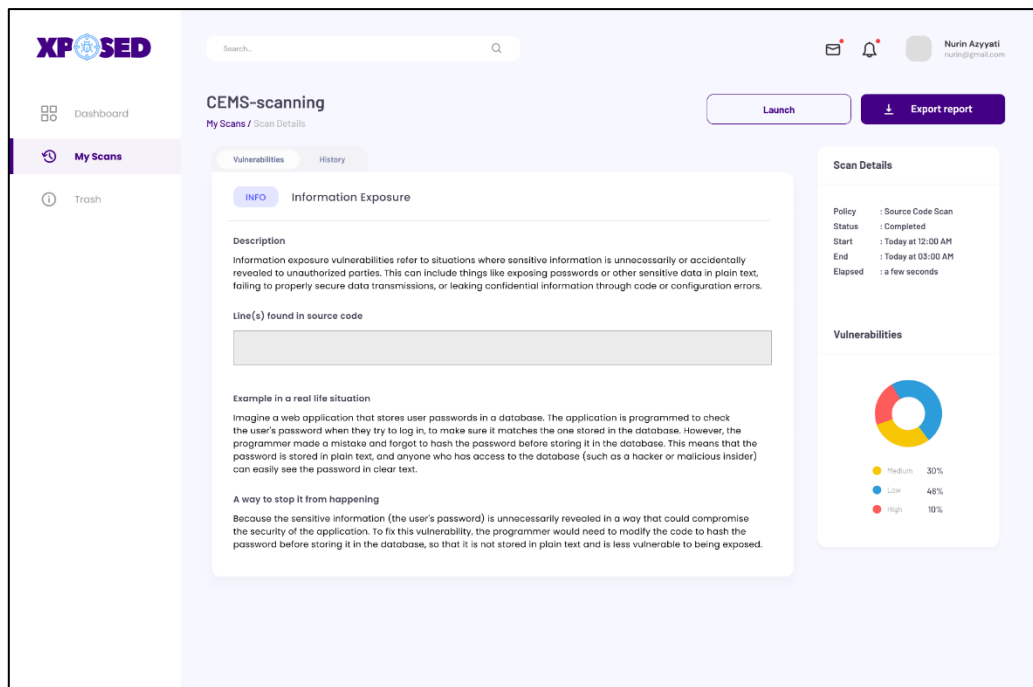


Figure 3.12 Scan Info interface of XPOSED

3.6.6 History Interface

The history interface, as depicted in Figure 3.13, allows users to view a chronological record of all previous scans. Users can access data such as the date and time of each scan and can see the total number of scans that have been performed.

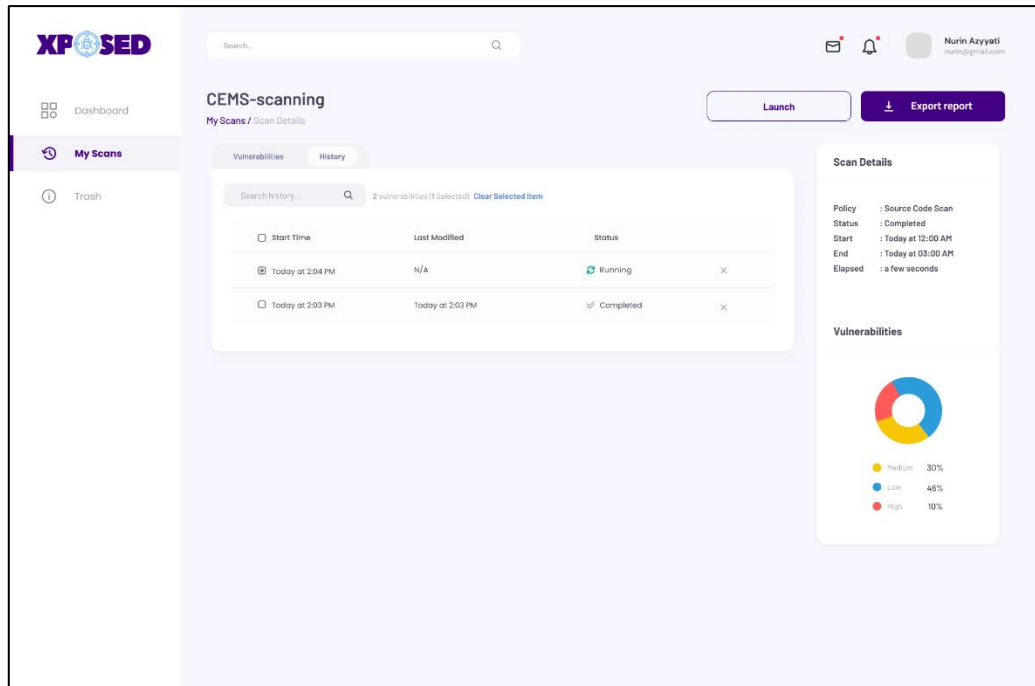


Figure 3.13 Scan History interface of XPOSED

3.7 Testing / Validation Plan

The primary objective of the UAT (User Acceptance Test) form is to thoroughly evaluate the performance of the XPOSED Vulnerability Detection System in real-world scenarios. The UAT form as shown in Figure below is specifically designed for end-users who actively utilize the system. By collecting valuable user feedback through the UAT form, the development team can gain insights into the system's strengths and areas for improvement. This feedback serves as a vital input for implementing necessary enhancements or modifications to the system, ensuring that it meets the end-users' requirements and expectations. Additionally, the feedback received from the UAT form contributes to an overall evaluation of the system's effectiveness, allowing for continuous refinement and optimization.

No.	Module	Activities	Status		Comments
1	Login	User login	Yes ()	No ()	
		User logout	Yes ()	No ()	
		Forgot password	Yes ()	No ()	
2	Register	New user registration	Yes ()	No ()	
		Form validation	Yes ()	No ()	
		Submit registration	Yes ()	No ()	
3	Dashboard	Display correct data	Yes ()	No ()	
		Chart visualization	Yes ()	No ()	
4	Manage Scan	Upload source code	Yes ()	No ()	
		Submit new scan	Yes ()	No ()	
		Display scan results	Yes ()	No ()	
		Delete scan record	Yes ()	No ()	
		Restore scan record	Yes ()	No ()	
		Download report	Yes ()	No ()	
		Email Notification	Yes ()	No ()	
5	Manage Risk	View word risk	Yes ()	No ()	
		Edit selected record	Yes ()	No ()	
		Delete record	Yes ()	No ()	
		Add new risk	Yes ()	No ()	

This test has been performed by:

Name : _____
 Signature : _____
 Date : _____

3.8 Potential Use of Proposed Solution

XPOSED is a tool that can be used to identify potential vulnerabilities in software application in real-time. This can include identifying vulnerabilities in software and hardware, as well as in the organization's overall network infrastructure.

In a real-time situation, XPOSED can be used to detect and alert IT staff to potential vulnerabilities as they occur. This can help organizations to quickly respond to and address potential security threats. For example, if a new vulnerability is discovered in a widely used software application, this system can be used to scan the running source code in order to determine if any files are running the affected software. The XPOSED system can then alert IT staff, who can take action to patch the vulnerability before an attacker can exploit it.

XPOSED solutions in the future, can also be commercialized for companies and organizations that want to protect their systems from vulnerabilities. These solutions can be used by companies of all sizes and in various industries, including healthcare, finance, and government. They can be deployed on-premises or in the cloud and can be customized to meet the specific needs of the organization.

Besides, XPOSED system offer additional services such as vulnerability information to help organizations identify the nature of the security risks. They also provide a detailed report on the vulnerabilities found, which can be used for compliance purpose.

Thus, XPOSED is a valuable tool that can help users and organizations to proactively identify and address potential security threats in real-time. It may also be commercialised in the future to be able to offer a service to other organisations.

3.9 Gantt Chart

The Gantt chart is a tool used to visualize the timeline of a project by breaking down tasks into bars on a timeline. It is often used in conjunction with the Incremental and Iterative model to ensure that the project is completed on schedule. The chart in Figure 3.14 illustrates the planned schedule for the project using this method.

UNDERGRADUATE PROJECT SCHEDULE														
Activities	Duration (Week)	UNDERGRADUATE PROJECT I 2022/2023 SEM 1												
		1	2	3	4	5	6	7	8	9	10	11	12	13
PHASE 1: PLANNING	1-2													
Define problem statement, objective, scope	1	█												
PHASE 2: ANALYSIS	2-4													
Define OWASP and CWE SANS	2		█											
Construct mapping between two sources	2		█											
Compare existing detection tools	3			█										
Determine suitable SLDC	4				█									
Identify project requirements	4				█									
PHASE 3: DESIGN	5-13													
Construct flowchart	5					█								
Design use case diagram	6					█	█							
Develop context diagram	6					█	█							
Create an activity diagram	6					█	█							
Design entity relationship diagram	7						█	█						
Construct data dictionary	7						█	█						
Develop design for database	8							█	█					
Design storyboard	9-10								█	█				
Develop design for user interface	10-11									█	█			
Design UAT form	12										█	█		
Identify the significant of the project	13												█	█

Figure 3.14 Gantt chart table of XPOSED project

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, the process and methodology used to deploy the XPOSED Vulnerability System will be explained. The system flow will be described in detail, starting from the initial stages until the completion of the system. Additionally, a user manual has been created to guide users on how to use the completed system. The user manual will include screenshots and explanations that will provide step-by-step instructions for the user to follow and understand. To ensure that the system performs optimally, a User Acceptance Test (UAT) form will be designed. This form will be used to test all the functional and non-functional requirements of the system, and users will be asked to provide feedback on their experience. The feedback received will be used to improve the system, and all the results will be recorded in the UAT form. It is important to note that several tests are needed to examine the performance of the system thoroughly. By doing so, it can be ensured that the system meets all the requirements and performs as expected. The UAT form will be an essential tool in this process, as it will provide valuable insights into the system's performance and areas that need improvement.

4.2 Implementation

The developed system has been implemented using three different web-based programming languages, namely PHP, CSS, and JavaScript. During the development process, Visual Studio Code was used as a source-code editor to write code for each interface design of the existing page. Before deploying the system into a real server, XAMPP was used as a demo web-server. The web-server will connect the web with the database, which is MySQL. This connection is essential for the system to function correctly and to store and retrieve data from the database. The XPOSED system consists of two types of users, namely End Users and Admin. Each of these users will have their own interface, and each of them will have different implementations towards the system. The End Users will be able to use the system to identify vulnerabilities in their systems, while the Admin will have access to additional features that allow them to manage the system and its users. The system includes several pages, such as the sign-up page, sign-in page, profile page, and notifications page. Each of these pages serves a specific purpose and is designed to provide a seamless user experience.

4.2.1 User Manual

This section provides a detailed description of each existing interface in the XPOSED system. Step-by-step instructions are included to guide users on how to use the system effectively. Each interface is designed to be user-friendly and intuitive, with clear instructions and easy-to-use features.

4.2.1.1 User Sign Up or Register

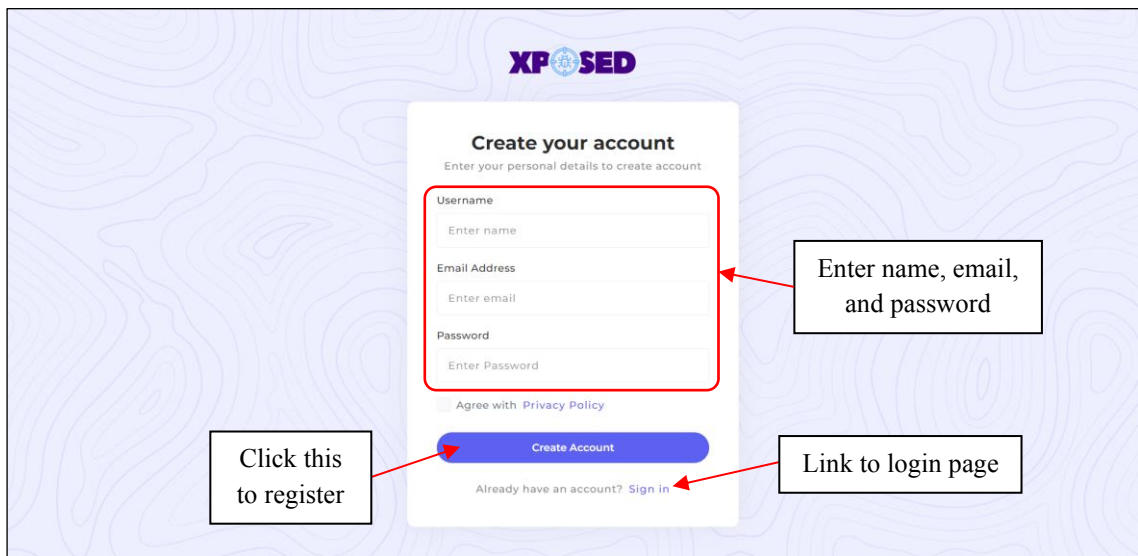


Figure 4.1 Register page

Displayed in Figure 4.1 is an essential part of the XPOSED system, which is the page that allows new users to create an account and gain access to the system's features. To create an account, users must provide their personal information, including their name, email address, and password. This information is necessary to ensure that each user has a unique account and to protect the system from unauthorized access. For users who already have an account, they can click on the Login link to access their existing account.

4.2.1.2 User Login

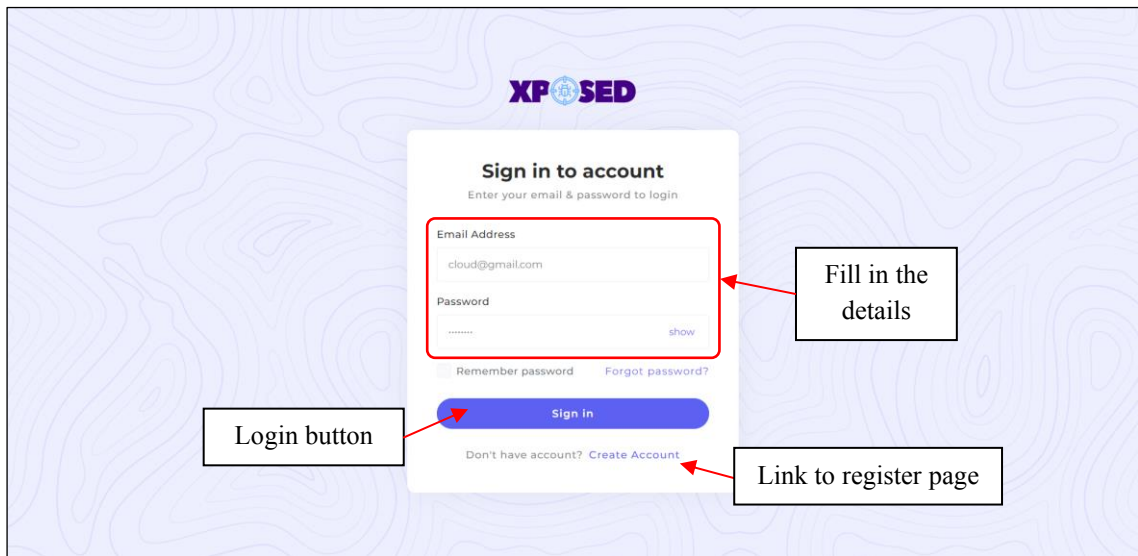


Figure 4.2 Login page

Figure 4.2 shows the login page, which allows existing users to log in to the system. Users must enter their existing email and password before clicking on the Login button. If a user wishes to register or create an account, they simply need to click on the Register link, which will redirect them to the registration page.

4.2.1.3 End User

4.2.1.3.1 Viewing dashboard overview

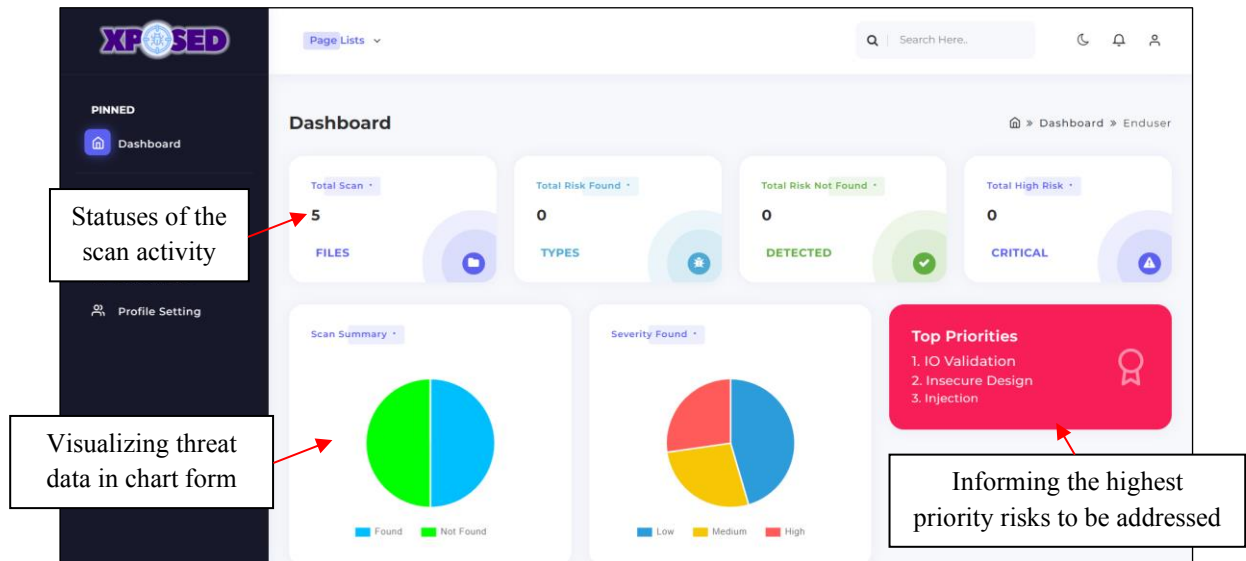


Figure 4.3 Dashboard overview for users

The dashboard page, displayed in Figure 4.3, is a vital component of the XPOSED system. It provides users with real-time information about their scan activity, allowing them to stay informed and take proactive measures to prevent vulnerabilities from occurring. The dashboard is designed to be user-friendly and intuitive, with clear visualizations and easy-to-use features.

The upper section of the dashboard displays the status of existing scan activity, including total scan, total risk found etc. This information keeps users informed about the scan's activity. The middle section of the dashboard provides users with a simplified threat summary in a pie chart view, indicating the severity level of vulnerabilities. Additionally, the dashboard informs users which type of vulnerability they need to pay attention to more, according to the rank of OWASP Top 10.

By clicking on the view button in the lower section of the dashboard, users can access detailed information about their recent scans. This action will redirect them to the My Scan page.

4.2.1.3.2 Uploading source code folder

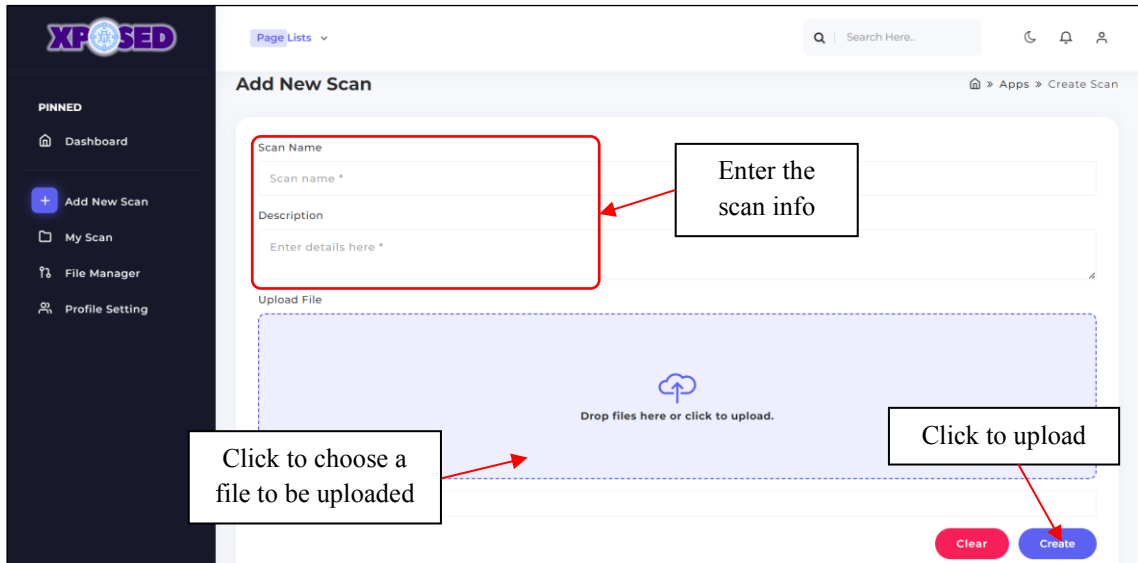


Figure 4.4 Interface when uploading folder

Figure 4.4 shows the interface that allows users to upload their source code folder into the XPOSED system. To upload their folder, users can click on the "Add New Scan" button located on the left navigation bar. This button provides users with a quick and easy way to initiate the scanning process. The system only accepts zip file format for the folder that can be uploaded, and any other file format will result in an error message. To upload a folder, users must first enter the scan name and then select the zip folder they want to upload by clicking on the "Choose File" button. Once the folder is selected, users can upload it into the system by clicking the "Create" button. The system will then save the uploaded zip folder into the database and display a success message to indicate that the uploading process is complete.

4.2.1.3.3 Managing scan records

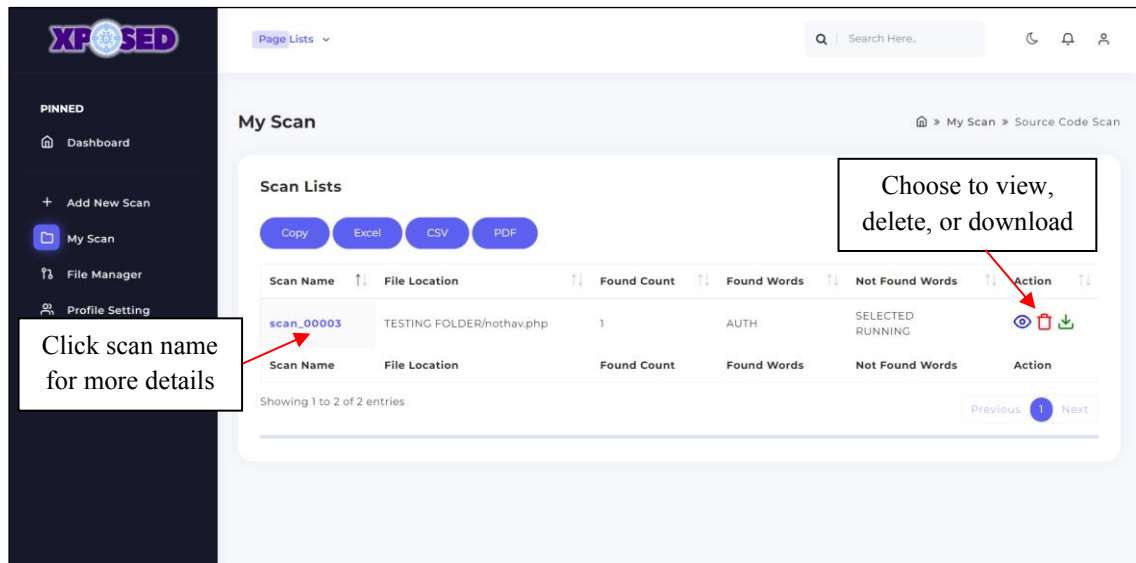


Figure 4.5 List of scans page

In Figure 4.5, the user interface displays a list of scan activities in a table format, providing an organized and visually appealing layout. The user can sort the scan names and use the search feature to find specific information. The primary focus of the table is to present crucial data that pertains to the presence or absence of specific words in the uploaded zip file. This concise presentation offers users a rapid summary of the content within their uploaded file. Moreover, the user has the ability to perform various actions on each listed scan, including viewing, deleting, or downloading the scan activity.

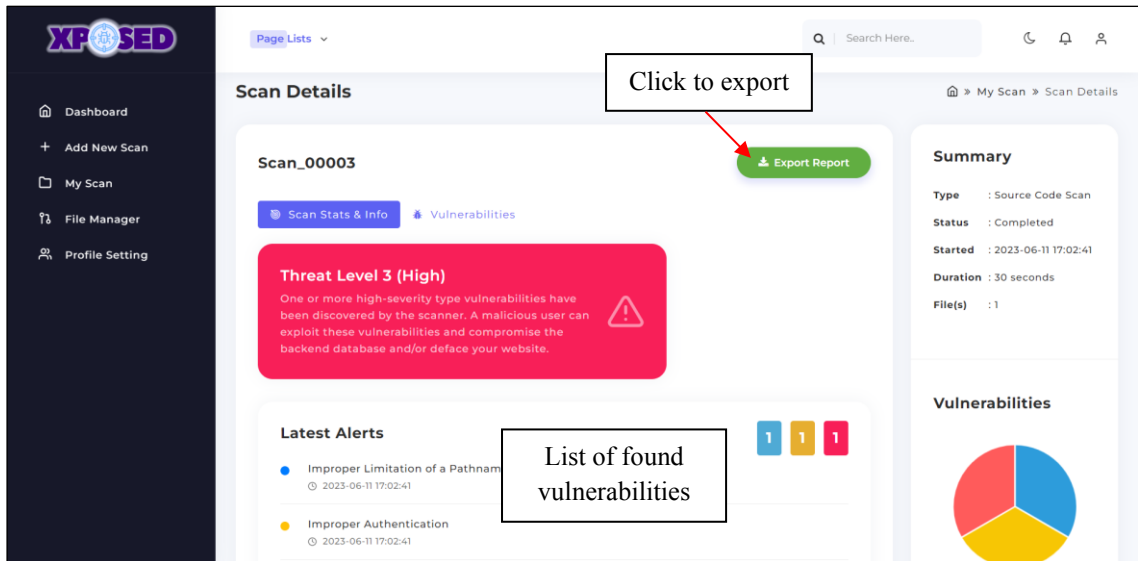


Figure 4.6 Scan details page

Figure 4.6 displays the interface that appears when the user clicks on a scan name from the previous interface. This interface lists all the vulnerabilities found during the scan process. For a comprehensive understanding of a particular security risk, users can effortlessly scroll down the page to access a dedicated section containing detailed information about the risk type. Additionally, the right section of the interface offers a concise summary of the completed scan. To export a report of the displayed information, users can simply click on the Export button, enabling them to conveniently obtain a comprehensive record of the relevant details. The interface is designed to be minimalist, making it easy for the user to navigate and use the system with ease.

4.2.1.3.4 Restoring Deleted Scan

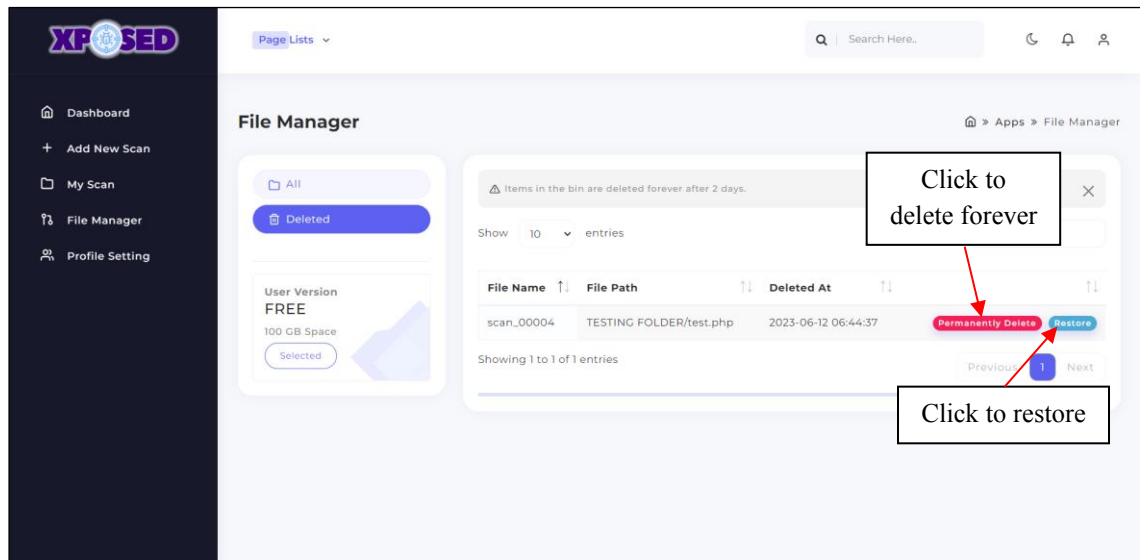


Figure 4.7 Bin page

Figure 4.7 shows the interface that gathers all the deletions made by the user, providing them with a safety net to restore any crucial folders that they may have accidentally deleted. The bin interface displays the file name, file path, deleted at, and actions, which include two buttons: the "Permanently Delete" and "Restore" buttons. If users click on the "Permanently Delete" button, the system will ask for confirmation of their delete action. This feature is designed to prevent users from accidentally deleting folders and to ensure that they are aware of the consequences of their actions. The "Restore" button is a feature in the bin interface that allows users to restore any accidentally deleted folders. When a user clicks on the "Restore" button, the system will restore the selected folder back into the list of scan activity, and users will be able to use the restored folder for the scan process.

4.2.1.3.5 Updating user account

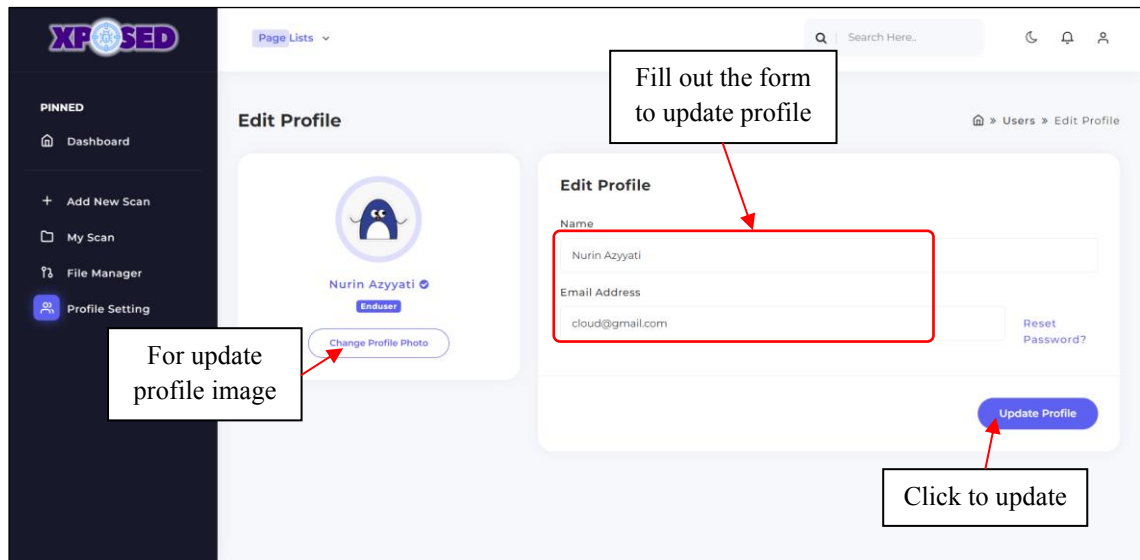
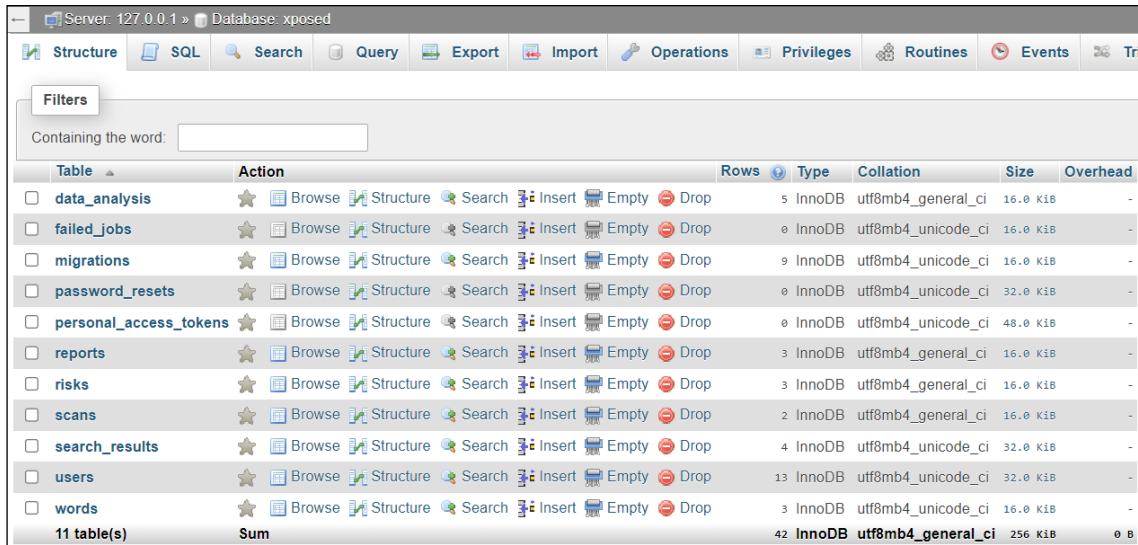


Figure 4.8 User profile settings page

This interface in Figure 4.8 allows users to update or change several details related to their account. Users can update their account name and username, as well as their profile image by following the specified image file type and size requirements. To update their account details, users can change any of the displayed data and click the "Update Account" button. This will save the new changes made in the database. Additionally, if users forget their current password, they can reset it by clicking on the "Reset Password" button.

4.2.2 Database Design

This section focuses on the testing of the XPOSED system with the MySQL database. The database, named "xposed," records all the data completed by users. To record all the information entered or changed by users, several tables have been created in the database. These tables are designed to store all the necessary information related to the system, ensuring that all user data is accurately recorded and easily accessible.



The screenshot shows the MySQL database structure for the 'xposed' database. The interface includes a menu bar with options like Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, and Events. Below the menu is a search filter for 'Containing the word:'. The main area displays a table with columns: Table, Action, Rows, Type, Collation, Size, and Overhead. The table lists 11 tables with their respective row counts and sizes.

Table	Action	Rows	Type	Collation	Size	Overhead
data_analysis	★ Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_general_ci	16.0 KiB	-
failed_jobs	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
migrations	★ Browse Structure Search Insert Empty Drop	9	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
password_resets	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
personal_access_tokens	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
reports	★ Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	16.0 KiB	-
risks	★ Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	16.0 KiB	-
scans	★ Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	16.0 KiB	-
search_results	★ Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
users	★ Browse Structure Search Insert Empty Drop	13	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
words	★ Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
11 table(s)	Sum	42	InnoDB	utf8mb4_general_ci	256 KiB	0 B

Figure 4.7 Implementation of xposed database

The table used for the "xposed" database is listed in Figure 4.11 above. Each existing page in the XPOSED system will have its data and information recorded in its corresponding table. This means that the data for each page will be stored in a structured manner, making it easier to access and analyze.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(10)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
2	migration	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
3	batch	int(11)			No	None			Change Drop More

Figure 4.8 Structure of migrations table

The migrations table in Figure 4.14 is used to keep track of changes made to the database schema over time. When developing a web application, it is common to make changes to the database schema as the application evolves and new features are added. The migrations table is used to store a record of these changes, allowing developers to easily track and manage the evolution of the database schema.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	scanID	varchar(191)	utf8mb4_general_ci		No	None			Change Drop More
3	userID	varchar(191)	utf8mb4_general_ci		No	None			Change Drop More
4	title	varchar(100)	utf8mb4_general_ci		Yes	NULL			Change Drop More
5	dateGenerated	timestamp			Yes	NULL			Change Drop More
6	linkFile	varchar(191)	utf8mb4_general_ci		Yes	NULL			Change Drop More

Figure 4.9 Structure of reports table

The "reports" table is depicted in Figure 4.15, and it is responsible for storing all the data related to the export reports generated by users. Whenever a user clicks on the "generate report" button, new data is added to this table and saved in the database for future reference. The data inserted into this table includes information such as the scanID, the user ID who exported the report, the title of the report, the date it was generated, and the link to the report.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	name	varchar(191)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	severity	int(10)			Yes	NULL			Change Drop More
4	owaspCategory	varchar(191)	utf8mb4_general_ci		Yes	NULL			Change Drop More
5	cwelID	varchar(191)	utf8mb4_general_ci		Yes	NULL			Change Drop More
6	sansCategory	varchar(191)	utf8mb4_general_ci		Yes	NULL			Change Drop More
7	baseScore	varchar(191)	utf8mb4_general_ci		Yes	NULL			Change Drop More
8	description	varchar(191)	utf8mb4_general_ci		Yes	NULL			Change Drop More
9	solution	varchar(191)	utf8mb4_general_ci		Yes	NULL			Change Drop More
10	impact	varchar(191)	utf8mb4_general_ci		Yes	NULL			Change Drop More
11	reference	varchar(191)	utf8mb4_general_ci		Yes	NULL			Change Drop More
12	dateDiscovered	timestamp			Yes	NULL			Change Drop More
13	status	varchar(191)	utf8mb4_general_ci		Yes	NULL			Change Drop More

Figure 4.10 Structure of risks table

The "risks" table, as shown in Figure 4.16, is a crucial component of the system that stores the latest information about software vulnerabilities that follow the current OWASP Top 10 and CWE SANS Top 25. The admin is responsible for updating the table with the latest rank of security risks according to the two sources being referred to. It is important for the admin to edit the records in the table every time the latest rank is released by the two sources, ensuring that users have access to the most up-to-date information regarding software vulnerabilities. The "risks" table contains columns that are based solely on the information provided by the sources' websites, as it is not appropriate to assume the software risks. The admin inserts data such as the risk name, severity level, the OWASP Top 10 category to which the vulnerability belongs, the base score, a brief description of the risk, and suggested solutions to mitigate the vulnerability. By storing this information in a dedicated table, users can easily access and retrieve the latest information about software vulnerabilities.

Server: 127.0.0.1 » Database: xposed » Table: scans

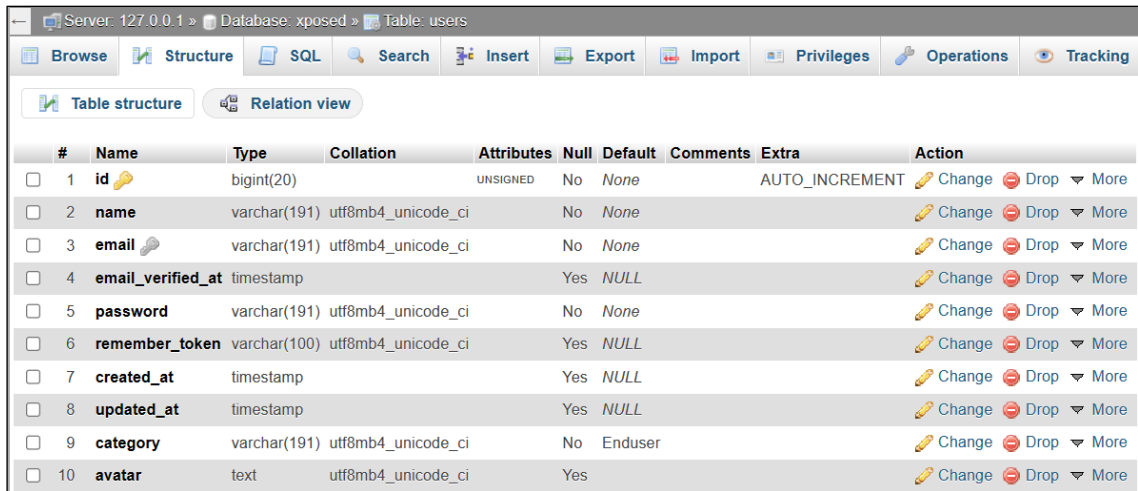
Table scans has been altered successfully.

```
ALTER TABLE `scans` CHANGE `id` `id` BIGINT(20) NOT NULL AUTO_INCREMENT;
```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)			No	None		AUTO_INCREMENT	Change Drop More
2	userID	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
3	reportID	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
4	riskID	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
5	scan_code	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
6	filename	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
7	line_number	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
8	file_path	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
9	scan_type	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
10	language	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
11	status	varchar(255)	utf8mb4_general_ci		No	None			Change Drop More
12	created_at	datetime			No	None			Change Drop More
13	updated_at	datetime			No	None			Change Drop More
14	deleted_at	datetime			No	None			Change Drop More

Figure 4.11 Structure of scans table

The "scans" table, as depicted in Figure 4.17, is responsible for keeping all the scan activities produced by the user. The data recorded in this table is listed on the List Scan page, and users can view or edit the information as needed. Any changes made by the user will be reflected in this table, ensuring that the information is always up-to-date. The "scans" table contains important data such as the userID, start time of the scanning activity, type of scan, riskID, scan status, and reportID.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2 name	varchar(191)	utf8mb4_unicode_ci		No	None			Change Drop More
<input type="checkbox"/>	3 email	varchar(191)	utf8mb4_unicode_ci		No	None			Change Drop More
<input type="checkbox"/>	4 email_verified_at	timestamp			Yes	NULL			Change Drop More
<input type="checkbox"/>	5 password	varchar(191)	utf8mb4_unicode_ci		No	None			Change Drop More
<input type="checkbox"/>	6 remember_token	varchar(100)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
<input type="checkbox"/>	7 created_at	timestamp			Yes	NULL			Change Drop More
<input type="checkbox"/>	8 updated_at	timestamp			Yes	NULL			Change Drop More
<input type="checkbox"/>	9 category	varchar(191)	utf8mb4_unicode_ci		No	Enduser			Change Drop More
<input type="checkbox"/>	10 avatar	text	utf8mb4_unicode_ci		Yes				Change Drop More

Figure 4.12 Structure of users table

The "users" table in Figure 4.18 is also a crucial table, as it stores all the records of users who have registered their accounts and also data related to user accounts, including their personal information and login credentials. This table contains important data such as the user's ID, name, email address, password, category, and avatar. By keeping a record of this information, users can easily access and manage their account information, while also ensuring that the system remains secure and reliable. Users can easily manage their account information, update their personal details, and change their login credentials as needed.

4.2.3 Coding Implementation

TO VIEW

```
55 public function dashboard()
56 {
57     if (Auth::check()) {
58
59         $category = Auth::user()->category;
60         if ($category == 'Enduser') {
61
62             $files = SearchResult::select('id', 'scan_code', 'filename')
63                 ->groupBy('id', 'scan_code', 'filename')
64                 ->get();
65
66             return view('dashboards.enduser', compact('files'));
67         }
68         if ($category == 'Admin') {
69             return view('dashboards.admin');
70         }
71     }
72
73     return redirect("login")->withSuccess('Oops! You do not have access!');
74 }
```

TO UPDATE

```
11 public function updateAvatar(Request $request)
12 {
13     $request->validate([
14         'avatar' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
15     ]);
16
17     $user = Auth::user();
18
19     $avatarName = $user->id . '_avatar.' . request()->avatar->getClientOriginalExtension();
20
21     $request->avatar->move(public_path('uploads'), $avatarName);
22     $user->avatar = $avatarName;
23     User::where('id', '=', $user->id)->update(['avatar' => $avatarName]);
24
25     return back()
26         ->with('success', 'You have successfully upload image.');
```

TO DELETE

```
142 public function deleteFile(Request $request, $id)
143 {
144     if ($request->ajax()) {
145
146         SearchResult::where('scan_code', '=', $id)->delete();
147
148         return response()->json(array('success' => true));
149     }
150 }
```

TO ADD

```
public function store(Request $request)
{
    $category = $request->input('category');
    $description = $request->input('description');
    $word = $request->input('word');
    $line = $request->input('line');
    $severity = $request->input('severity');
    $language = $request->input('language');
    $example = $request->input('example');
    $solution = $request->input('solution');

    $data = array(
        'category' => $category,
        "description" => $description,
        "word" => $word,
        "line" => $line,
        "severity" => $severity,
        "language" => $language,
        "example" => $example,
        "solution" => $solution
    );

    //dd($data);

    DB::table('words')->insert($data);

    Alert::success('Word added successfully');

    return redirect()->route('words');
}
```

4.3 Testing and Result Discussion

In this section, testers will test the system to identify any problems or areas for improvement that need to be addressed to ensure that the system functions well and meets all the intended requirements. The UAT form will be divided into several modules for testing, and based on the user acceptance test, all testers will provide feedback on the system's functionality and any issues they encounter while using it. Testers will also provide comments on any areas where the system could be improved to make it more efficient and effective for users.

This testing is crucial before the XPOSED system is released for public use, as it helps to ensure that the system is functioning as intended and meets the needs of its users. By identifying any issues or areas for improvement during the testing phase, developers can make the necessary changes to the system to ensure that it is reliable, efficient, and effective. Thus, this testing phase is an essential component of the system development process, helping to ensure that the system meets the needs of its users and functions well in real-world scenarios.

CHAPTER 5

CONCLUSION

5.1 Introduction

It can be said that this development of XPOSED Vulnerability Detection System represents a significant advancement in the field of cybersecurity by providing a comprehensive solution for identifying and mitigating vulnerabilities in software applications. The project aimed to enhance the security posture by offering an automated and efficient vulnerability detection process.

Throughout the development of the XPOSED system, several key features were implemented to achieve its objectives. These features included advanced scanning algorithms, comprehensive vulnerability databases, and real-time monitoring capabilities. By leveraging these capabilities, the system effectively identified and alerted users to potential vulnerabilities, allowing them to take proactive measures to protect their systems.

However, it is important to acknowledge that the XPOSED Vulnerability Detection System has some limitations that need to be addressed. While the system demonstrated promising results in detecting known vulnerabilities, further research and development are required to improve its effectiveness in identifying zero-day vulnerabilities and emerging threats. Additionally, the system's performance and scalability could be enhanced to accommodate larger networks and ensure efficient scanning across diverse environments.

Furthermore, user feedback and evaluations played a crucial role in assessing the usability and effectiveness of the XPOSED system. User Acceptance Testing (UAT) form were distributed to gather insights and opinions from users regarding their experience with the system. The feedback received indicated a positive response, with

users expressing satisfaction with the system's usability and its ability to enhance their organization's security posture.

In conclusion, the XPOSED Vulnerability Detection System has made significant progress in achieving its objectives of automating vulnerability detection and enhancing cybersecurity. Moving forward, it is recommended to address the identified limitations and flaws to further improve the system's effectiveness and user satisfaction. This includes conducting research to detect zero-day vulnerabilities, optimizing system performance, and expanding its capabilities to adapt to evolving security threats. By continuously enhancing the system, the XPOSED project can make a valuable contribution to the ongoing effort to protect computer systems from vulnerabilities and ensure the resilience of digital infrastructure.

5.2 Limitations and Constraints

XPOSED encounters several factors that limit its capabilities and impose constraints:

- Time constraints: The project faced limitations in terms of the available development time, which resulted in a reduced number of modules being incorporated into the system. This means that certain desired features or functionalities might not have been fully implemented or explored due to the limited timeframe for development.
- Insufficient knowledge about integrating social media account registration: Encountered challenges in implementing the functionality that allows users to register using their social media accounts, such as Facebook or Google. This might be due to a lack of understanding or expertise in integrating third-party authentication systems, which ultimately limited the system's ability to offer this convenient registration option for users.
- Limited knowledge about handling file types other than PHP and Java: Faced difficulties in enabling users to upload file types other than PHP and Java. This limitation might have been caused by a lack of familiarity with file handling techniques or a limited understanding of how to implement file scanning functionality for different file extensions. As a result, the system might not be able to effectively scan and analyze files of various formats, restricting its overall capability.
- Lack of proper guidance and instructions: The system lacks comprehensive guidance and instructions for users. The absence of highlighted features or a dedicated tour might make it challenging for users to understand how to navigate and utilize the system effectively. Clear and intuitive instructions, tooltips, or a step-by-step tutorial would greatly improve the user experience and ensure users can leverage the system's functionalities to their fullest extent.

5.3 Future Work

The detection tool system poses significant challenges, and while XPOSED has made progress, there is still ample room for improvement. The following enhancements could be considered for future work:

- Predictive Modelling for detecting upcoming vulnerability types: This feature could leverage the data gathered from user scan activities to develop predictive models. By analyzing patterns and trends, the system could potentially identify new types of vulnerabilities before they are officially documented. This predictive capability would provide users with more accurate information and enable proactive measures to address emerging threats.
- Source code editor and vulnerability highlighting: Adding a feature that allows users to view the uploaded source code in an integrated editor would enhance the system's usability. Upon completion of the scanning process, the system could indicate the specific lines of code where vulnerabilities may occur. Users would then have the ability to make changes or edits directly within the editor and save/download the modified source code. This feature would facilitate efficient vulnerability mitigation and reduce the risk of introducing new issues.
- Expanded chart visualization options: Consideration could be given to incorporating a wider range of chart types for visualizing the data generated by XPOSED. By offering different chart formats, users would have additional ways to comprehend and interpret the information presented. This visual diversity can enhance data analysis and promote a deeper understanding of the displayed chart data.

REFERENCES

Agile methodology — Zomato Case study | by Bhavz Kakarla | Medium. (n.d.). Retrieved January 22, 2023, from <https://medium.com/@srisayi.bhavani/agile-methodology-zomato-case-study-311da3388518>

CWE - 2022 CWE Top 25 Most Dangerous Software Weaknesses. (n.d.). Retrieved January 22, 2023, from https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html

CWE - Frequently Asked Questions (FAQ). (n.d.). Retrieved January 22, 2023, from <https://cwe.mitre.org/about/faq.html>

CWEs vs OWASP top 10? - DEV Community  (n.d.). Retrieved January 22, 2023, from <https://dev.to/caffiendkitten/cwes-vs-owasp-top-10-4imm>

C-YBER - What you need to know about Acunetix? (n.d.). Retrieved January 22, 2023, from <https://c-yber.com/what-you-need-to-know-about-acunetix/>

Introduction to Acunetix | Acunetix. (n.d.). Retrieved January 22, 2023, from <https://www.acunetix.com/support/docs/introduction/>

OWASP and its importance to Application Security – Conviso AppSec. (n.d.). Retrieved January 22, 2023, from <https://blog.convisoappsec.com/en/owasp-and-its-importance-to-application-security/>

OWASP Top 10:2021. (n.d.). Retrieved January 22, 2023, from <https://owasp.org/Top10/>

Qualys Vulnerability Scanner | Bugcrowd. (n.d.). Retrieved January 22, 2023, from <https://www.bugcrowd.com/glossary/qualys-vulnerability-scanner/>

What is Common Weakness Enumeration (CWE)? | Definition from TechTarget. (n.d.). Retrieved January 22, 2023, from <https://www.techtarget.com/searchsecurity/definition/Common-Weakness-Enumeration>

What is NISSUS and How Does it Work? - ITperfection - Network Security. (n.d.). Retrieved January 22, 2023, from <https://www.itperfection.com/network-security/network-monitoring/what-is-nessus-and-how-does-it-work-network-munitoring-vulnerabilit-scanning-security-data-windows-unix-linux/>

What is OWASP? What is the OWASP Top 10? | Cloudflare. (n.d.). Retrieved January 22, 2023,
from <https://www.cloudflare.com/learning/security/threats/owasp-top-10/>

