**Web Application Security Testing  
Final Project Report**

By

Jahnavi Goud Gudidevuni

Vinitha Tathari

This project is submitted to the Gannon University graduate faculty in

partial fulfillment for the degree Master of Science in Computer and Information Science.

Option: Cybersecurity

Approved:

|  |
| --- |
| **BAZAN-ANTEQUERA, RONNY C** |
| Advising Professor in Charge of Research |

Gannon University

Erie, Pennsylvania 16541

OCTOBER 2025

# Acknowledgement

We would like to express our sincere gratitude to **Bazan-Antequera, Ronny C** for continuous guidance, encouragement, and expert advice throughout this project. Our appreciation also goes to the **Gannon University** for providing the resources and environment necessary to conduct our research effectively.

We extend thanks to our classmates and peers for their collaboration and feedback, and to the open-source communities behind tools such as OWASP ZAP, Burp Suite, and SQLMap, whose contributions made this work possible.

Lastly, we thank our families and friends for their unwavering support and motivation. Their patience and encouragement were instrumental in the successful completion of this capstone project.

**Table of Contents**

[Acknowledgement 2](#_Toc211961285)

[Chapter 1: Introduction 4](#_Toc211961286)

[Chapter 2: Requirements Management 5](#_Toc211961287)

[Chapter 3: Design 6](#_Toc211961288)

[Chapter 4: Verification and Validation 7](#_Toc211961289)

[Week 1–4 Summary 7](#_Toc211961290)

[Week 5: Manual Testing and Input Validation Checks 7](#_Toc211961291)

[Week 6: Exploitation Phase 8](#_Toc211961292)

[Week 7: Continued Exploitation & Privilege Escalation 9](#_Toc211961293)

[Week 8: Risk Assessment and CVSS Scoring 15](#_Toc211961294)

[Week 9: Final Verification and Project Conclusion 17](#_Toc211961295)

[Week 10 – Integrate Findings into Technical Documentation 20](#_Toc211961296)

[Week 11 – Finalize Report and Presentation Slides 21](#_Toc211961297)

[Week 12 – Submit Final Project and Deliver Capstone Presentation 22](#_Toc211961298)

[Chapter 5: Conclusion 22](#_Toc211961299)

[Chapter 6: Bibliography 25](#_Toc211961300)

[Appendix A: Glossary 26](#_Toc211961301)

[Appendix B: Use Case Analysis 27](#_Toc211961302)

[Appendix C: Analysis & Design Models 28](#_Toc211961303)

[Appendix D: Design Models 29](#_Toc211961304)

[Appendix E: Screenshots 30](#_Toc211961305)

# Chapter 1: Introduction

**Overview**

This project focuses on identifying and documenting security vulnerabilities in web applications using an ethics-first, lab-based methodology. The scope includes setup of an isolated Kali Linux virtual machine (VirtualBox), deployment of vulnerable web targets (OWASP Juice Shop, DVWA) in Docker, manual and automated testing, exploitation in a controlled lab, and risk assessment using CVSS v3.1.

**Curriculum Scope**

The project integrates coursework topics from secure coding, penetration testing, and risk assessment. It demonstrates practical application of tools and methodologies discussed in class.

**Key Stakeholder Needs**

The stakeholders (course instructor, project team, and hypothetical client) need a clear assessment of web application security posture, reproducible evidence of vulnerabilities, and prioritized remediation guidance.

**Product Perspective**

The deliverable is a documented security assessment and remediation plan for the target web applications. The project follows a standard pentest workflow (recon → scan → manual test → exploit → risk assessment).

**Project Management Plan**

The team worked in weekly sprints (Weeks 1–9). Each week had specific deliverables: environment setup, reconnaissance, scanning, manual testing, exploitation, LPE enumeration, CVSS scoring, and final reporting.

# Chapter 2: Requirements Management

**Functional Requirements**

The project must (1) deploy lab targets in Docker; (2) document reconnaissance and scan results; (3) perform manual testing for input validation; (4) execute controlled exploitation in lab; (5) attempt local privilege escalation checks; (6) score findings using CVSS v3.1; and (7) produce a final report.

**Nonfunctional Requirements**

All testing is to be performed in an isolated environment. Evidence must be reproducible, sanitized for PII, and organized in the project folder. Reports must follow the academic template and use IEEE citation style.

**Interface Requirements**

The primary interfaces are the web browser (target interaction), Burp Suite and OWASP ZAP (testing tools), Docker and VirtualBox (infrastructure), and standard terminal access for Kali tools.

# Chapter 3: Design

**Environment Design**

The project environment uses a Kali Linux VM in VirtualBox. Vulnerable applications (OWASP Juice Shop and DVWA) run in Docker containers on the same VM. This isolate testing and ensures no external systems are affected.

**Architecture**

The architecture is intentionally simple: the Kali VM hosts Docker containers that expose web services on distinct ports (e.g., Juice Shop on 3000, DVWA on 80/8080). The tester uses a browser configured to proxy through Burp Suite for request interception.

**User Interface Design**

The testing interface is primarily command-line and Burp Suite GUI. Screenshots of key interactions are referenced as figures.

# Chapter 4: Verification and Validation

## Week 1–4 Summary

Weeks 1–4 focused on project planning, environment setup, and reconnaissance. The Kali VM and Docker targets were installed and validated. Nmap scans and passive recon identified open ports and services. OWASP ZAP and other scanners produced initial vulnerability candidates that guided manual testing in Week 5.

## Week 5: Manual Testing and Input Validation Checks

During Week 5 the team performed manual testing of application inputs to identify poor validation and output encoding. Using Burp Suite, fields such as login forms, search, and feedback were intercepted and modified to inject special characters, SQL fragments, and script tags. Evidence of reflected input and database error messages were recorded.

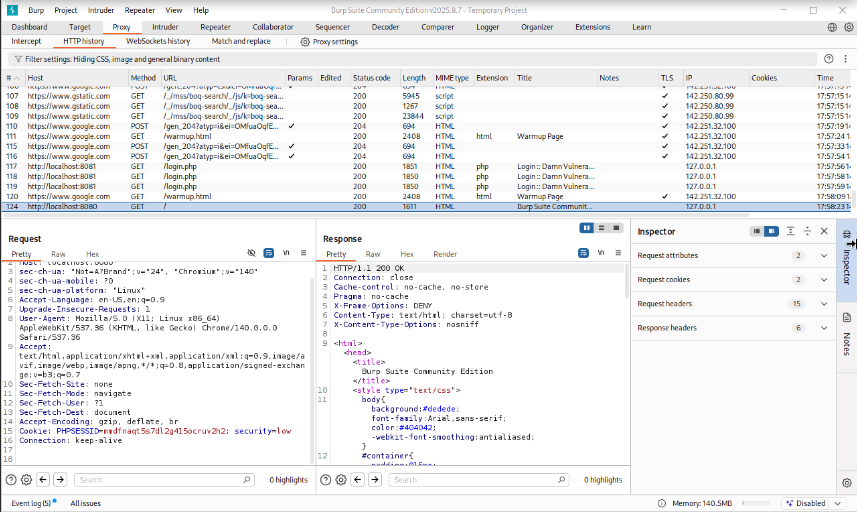


Figure 2 – Burp Suite proxy intercepting login request.

A screenshot of a computer

AI-generated content may be incorrect.

Figure – Burp Repeater showing SQL error response.

## Week 6: Exploitation Phase

Week 6 executed controlled exploitation of confirmed vulnerabilities. For SQL Injection, sqlmap was used against saved Burp requests to enumerate databases and run targeted queries. For XSS, safe PoC payloads were injected to demonstrate script execution without exfiltrating sensitive data. Evidence files and PoC steps were saved in the project folder.

A screenshot of a computer

AI-generated content may be incorrect.

Figure - sqlmap output for DB enumeration

A screenshot of a computer

AI-generated content may be incorrect.

Figure – Browser alert showing XSS PoC.

## Week 7: Continued Exploitation & Privilege Escalation

In Week 7 the team revalidated Week 6 PoCs and performed local enumeration on the Kali VM to identify privilege escalation vectors. Commands used included id, uname -a, sudo -l, find for SUID binaries, getcap, and checks for writable cron scripts. LinPEAS was used to aggregate findings and searchsploit was used to locate public exploit references.



Figure – id output confirming test user

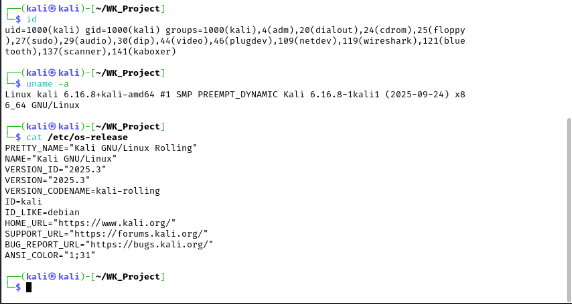




Figure 7 – sudo -l output

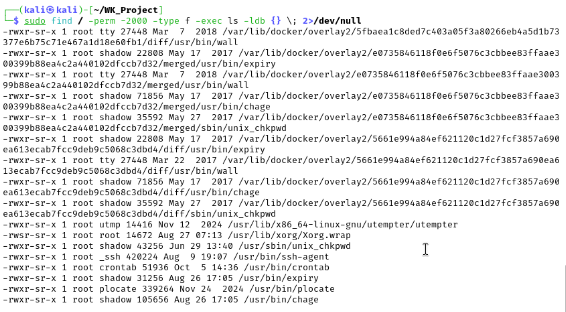
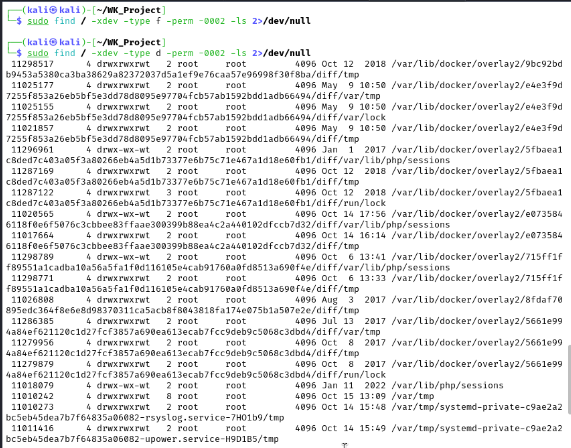


Figure – Listing of SUID binaries.



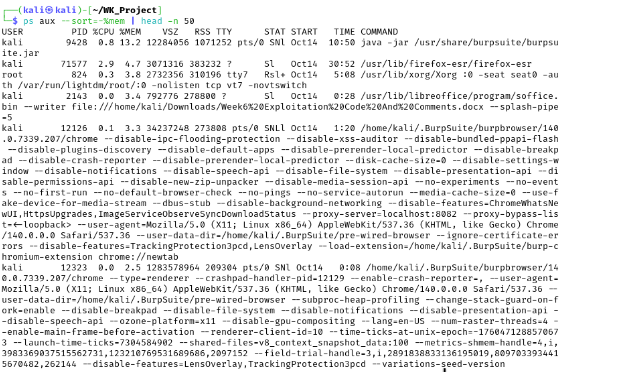
Posix capabilities



Word writable files and directories

**Processes and listening services**







**Search for known sploits**





Figure – LinPEAS summary output

## Week 8: Risk Assessment and CVSS Scoring

For Week 8 the confirmed vulnerabilities were recorded in a risk register and scored using CVSS v3.1. Representative vector strings and estimated base scores were recorded for each finding (SQLi, XSS, and LPE paths). The team recommended prioritized remediation steps.

****

Figure 10 – Risk register snapshot.

A screen shot of a computer

AI-generated content may be incorrect.

Figure 11 – CVSS calculator output.

## Week 9: Final Verification and Project Conclusion

Week 9 comprises final verification (retesting after hypothetical fixes), consolidation of evidence, and the writing of the final report. The team prepared remediation verification steps and compiled all evidence into a sanitized deliverable package for submission.

**Screenshot placeholders**

A screenshot of a computer program

AI-generated content may be incorrect.

Figure A – Juice Shop: server log after patch (juice\_shop\_logs\_after\_patch.txt)

A screenshot of a computer

AI-generated content may be incorrect.

Figure B – DVWA: server log after patch (dvwa\_logs\_after\_patch.txt)

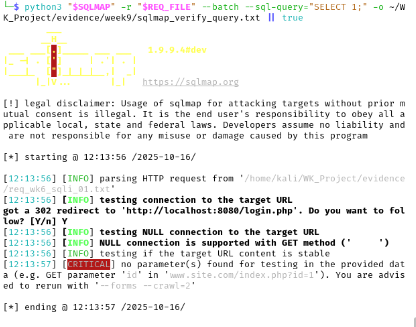


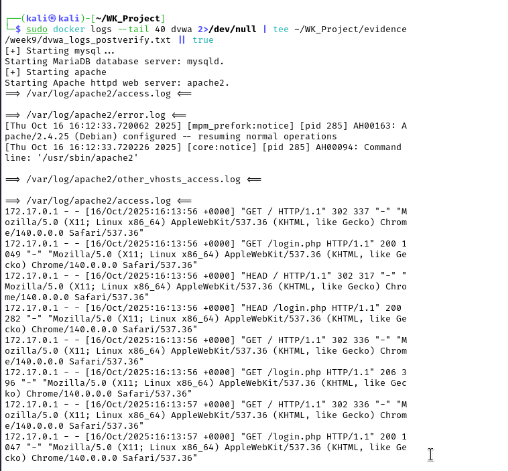
Figure C – sqlmap verification output (sqlmap\_db\_enum\_postpatch.txt)

A computer code with numbers and letters

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.





During Week 9 we implemented remediation strategies and security patches to address the critical issues found earlier. We applied parameterized queries to eliminate SQL injection attack vectors, added server-side output encoding and Content Security Policy to mitigate XSS, and hardened system configurations to prevent local privilege escalation. All changes were validated using safe verification techniques (sqlmap with non-destructive queries, curl-based XSS checks, and linpeas re-scan). Evidence and patched files are included in the project repository under the patches and evidence folders.

Prepared by: Jahnavi Goud Gudidevuni and Vinitha Tathari.