Dr. Babasaheb Ambedkar Technological University,
Lonere Maharashtra

A MINI PROJECT II REPORT ON
“Website Security Analysis”
In partial fulfilment of the requirement for the award of the Degree of
Bachelor of Technology in Computer Science & Engineering

Submitted by

Name: Vitthal Madhav Dhanve PRN-2121121242059

Under the Guidance of
Dr. P. S. Deshpande
Department of CSE

Department of Computer Science & Engineering
Shreeyash College of Engineering & Technology
(Approved by AICTE, New Delhi, accredited by NAAC, New Delhi & Affiliated to DBATU)

2023-24
SHREEYASH COLLEGE OF ENGINEERING & TECHNOLOGY
(Affiliated to DBATU, Lonere, MAHARASHTRA)
Department of Computer Science & Engineering

CERTIFICATE
This is to certify that the project work entitled “Website Security Analysis” is a bonafide work carried out by Vitthal Madhav Dhanve in partial fulfilment for the award of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING of Dr. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY, LONERE, MAHARASHTRA. It is certified that all correction / suggestion indicated for internal Assessment have been incorporated in the report deposited in the departmental library. The project seminar has been approved as it satisfies the academic requirement in respect of project work prescribed for the Bachelor of Technology Degree.

Name: Vitthal Madhav Dhanve PRN-2121121242059

........…………………… ..…………………………
Dr. P. S. Deshpande Dr. P. S. Deshpande

Project Guide Project Coordinator

…..………………………… ………………………....
Dr. P. S. Deshpande Dr. B. M. Patil
Head of the Department Principal

The mini project ii report entitled
“Website Security Analysis”

Submitted By:
Is approved for the degree of Bachelor of Technology (Computer Science & Engineering) award by Dr. Babasaheb Ambedkar Technological University, Lonere, Maharashtra.

Date:
Place: Sambhajinagar

ACKNOWLEDGEMENT

With a profound feeling of immense gratitude and affection, I would like to thank my mini project guide DR. P.S. Deshpande for her continuous support, motivation, enthusiasm and guidance. Her encouragement, supervision with constructive criticism and confidence enabled me to complete this project. I express my admiration for Dr. P. S. Deshpande mini project ii Coordinator for her valuable advice and support throughout this venture.
I also wish to extend my gratitude to DR. P. S. Deshpande, Head of Computer Science and Engineering Department for motivating me to put my best efforts in this project work. We express our deep gratitude towards Dr. B. M. Patil, Principal (SYCET) for constant motivation and providing necessary infrastructure. Finally, graceful thanks to family, friends, colleagues and everyone who has directly or indirectly contributed to make this mini project a success.

Vitthal Madhav Dhanve 2121121242059

ABSTRACT

This project focuses on the security analysis of a web application utilizing various penetration testing tools such as Wapiti, Nmap, Burp Suite, and SQLMap. Through this analysis, two critical vulnerabilities were identified: Reflected Cross-Site Scripting (XSS) and Server-Side Injection. Reflected XSS enables attackers to inject malicious scripts into web pages, potentially leading to various security threats. Server-Side Injection vulnerabilities, such as SQL Injection, allow attackers to execute arbitrary commands on the server, compromising data integrity and availability. The exploitation of the Server-Side Injection vulnerability highlighted the severe impact such vulnerabilities can have on web application security. Mitigation strategies, including input validation, output encoding, prepared statements, and security policies like Content Security Policy (CSP), were explored to address these vulnerabilities. Additionally, a literature survey provided insights into existing research and industry practices, shaping the project's methodology and emphasizing the importance of proactive security measures in web application development.

INDEX

INTRODUCTION

This mini project involves conducting a comprehensive security analysis on a web application by performing various penetration tests. The goal was to identify and exploit vulnerabilities to understand their potential impact on the application's security and stability. This report details the findings and provides recommendations for mitigating the identified vulnerabilities. Through the use of industry-standard tools such as Wapiti, Nmap, and Burp Suite, a thorough assessment was carried out to uncover and analyze security flaws within the web application.

LITERATURE SURVEY

Introduction

This literature survey reviews significant research and industry practices related to web application security, particularly focusing on Cross-Site Scripting (XSS) and Server-Side Injection vulnerabilities. The survey provides a foundation for understanding the current landscape, methodologies, and conclusions from previous studies. These insights have directly influenced the direction and goals of the current project.

Review of Related Work

1. OWASP Top Ten (2021)
Authors: OWASP Community
Methodology: This project by the Open Web Application Security Project (OWASP) identifies and ranks the ten most critical web application security risks based on data from a broad range of organizations.
Conclusion: The OWASP Top Ten highlights Cross-Site Scripting (XSS) and Injection flaws as prevalent and severe vulnerabilities. It emphasizes the need for robust input validation, output encoding, and the use of security frameworks.
Influence: This research guided the focus on XSS and Server-Side Injection as primary vulnerabilities in this project.

2. "An Empirical Study of Web Vulnerability Discovery Ecosystems" (2018)
Authors: Liang Gong, Shuo Chen, Hamed Okhravi
Methodology: This study analyzed vulnerability reports from multiple web applications to understand the discovery and exploitation patterns of web vulnerabilities.
Conclusion: The study found that automated tools are essential for initial vulnerability discovery, but manual analysis is crucial for uncovering complex issues.
Influence: This work underscored the importance of using both automated tools (like Wapiti and Burp Suite) and manual testing in this project.

3. "A Comprehensive Study of SQL Injection Attacks and Countermeasures for Web Applications" (2012)
Authors: Rajdeep Borgohain, Abhijit Bhattacharjee, Sushanta Biswas
Methodology: This paper surveyed various types of SQL Injection attacks and evaluated the effectiveness of different prevention techniques.
Conclusion: The study concluded that prepared statements and parameterized queries are the most effective defenses against SQL injection.
Influence: The findings informed the implementation of prepared statements and parameterized queries as part of the mitigation strategies in this project.

4. "Cross-Site Scripting (XSS) Attacks and Defense Mechanisms: Classification and State-of-the-Art" (2015)
Authors: E. Athanasopoulos, S. Ioannidis
Methodology: This paper classified XSS attacks and reviewed various defense mechanisms, including input validation, output encoding, and Content Security Policy (CSP).
Conclusion: Effective defense against XSS requires a combination of multiple techniques, emphasizing the need for a layered security approach.
Influence: This study's recommendations shaped the project's mitigation strategies, including the adoption of CSP and rigorous input/output handling.

5. "SQL Injection: Modes of Attack, Detection and Prevention" (2010)
Authors: Halfond, Viegas, Orso
Methodology: The paper reviewed various SQL injection attack methods and presented different detection and prevention techniques.
Conclusion: The study highlighted the effectiveness of dynamic analysis and runtime monitoring for detecting SQL injection attacks.
Influence: This research influenced the project's decision to use dynamic analysis tools like SQLMap for detecting injection vulnerabilities.

6. "Web Application Security: Threats and Countermeasures" (2013)
Authors: Shifali, V. & Nidhi, M.
Methodology: This paper analyzed common web application threats and evaluated countermeasures based on their effectiveness and implementation complexity.
Conclusion: The paper concluded that a comprehensive security strategy involves regular updates, code reviews, and the use of security libraries and frameworks.
Influence: The project's emphasis on regular security assessments and the use of security frameworks is directly influenced by these findings.
7. "A Survey on Automated Dynamic Analysis Techniques for Web Application Security" (2017)
Authors: Marco Cova, Davide Balzarotti, Giovanni Vigna
Methodology: The survey evaluated different automated dynamic analysis tools for their ability to detect and prevent web application vulnerabilities.
Conclusion: Dynamic analysis tools, while effective, need to be complemented with manual testing to cover a broader range of vulnerabilities.
Influence: This study reinforced the project's approach of using automated tools like Wapiti and Burp Suite in conjunction with manual penetration testing.
Influence on Current Project
The reviewed research has significantly influenced the current project's methodology and goals:
Focus on XSS and Server-Side Injection: Guided by the OWASP Top Ten and other studies, the project targets these critical vulnerabilities.
Use of Automated and Manual Testing: Informed by the empirical studies, the project employs both automated tools (Wapiti, Burp Suite, Nmap, SQLMap) and manual testing to ensure comprehensive vulnerability detection.
Mitigation Strategies: Recommendations from the literature on input validation, output encoding, CSP, prepared statements, and regular assessments have shaped the project's security measures.

TOOLS USED
Wapiti:
Description: Wapiti is an open-source web application vulnerability scanner that allows for comprehensive assessments of web applications. It performs black-box testing by crawling the web pages and analyzing the scripts to detect various types of vulnerabilities.
Functionality: Wapiti scans a web application by injecting payloads into forms and URL parameters to test for vulnerabilities such as XSS, SQL injection, file inclusion, and more. It generates detailed reports that help in understanding the security posture of the application.
Advantages: Being open-source, it is accessible and customizable for various needs. It supports a wide range of vulnerabilities and provides a detailed output that is useful for both detection and remediation efforts.
Nmap:
Description: Nmap (Network Mapper) is a powerful network scanning tool used to discover hosts and services on a computer network by sending packets and analyzing the responses. It helps in network discovery, security auditing, and vulnerability detection.
Functionality: Nmap can identify open ports, running services, and the operating system of the target hosts. It uses various scanning techniques such as TCP connect scans, SYN scans, and UDP scans to gather information about the network infrastructure.
Advantages: Nmap is versatile and can be used for both simple network scans and complex vulnerability assessments. It is widely regarded in the industry for its efficiency, reliability, and comprehensive capabilities.
Burp Suite:
Description: Burp Suite is a comprehensive platform for web application security testing. It includes various tools such as a proxy server, web spider, intruder, repeater, and scanner that facilitate the

identification and exploitation of vulnerabilities.
Functionality: Burp Suite allows intercepting and modifying HTTP/S traffic between the browser and the target application. The scanner automates the detection of common web vulnerabilities like XSS, SQL injection, and CSRF. The intruder can be used for custom attacks, while the repeater helps in manually testing and verifying issues.
Advantages: Burp Suite's intuitive interface and powerful tools make it a favorite among security professionals. It offers both automated and manual testing capabilities, which are essential for thorough security assessments.

VULNERABILITIES FOUND

REFLECTED CROSS-SITE SCRIPTING (XSS)

Description: Reflected Cross-Site Scripting (XSS) is a type of security vulnerability that occurs when a web application takes user input and includes it in the output without proper validation or escaping. This allows attackers to inject malicious scripts into web pages viewed by other users.
Mechanism: The attacker sends a crafted URL to a victim, which includes malicious script code as part of the input parameters. When the victim clicks the link, the web application reflects the input back to the browser in a way that executes the script.
Detection Method: Detected using Burp Suite's scanner, which identified input fields where user-supplied data is directly returned in the HTML response. The scanner flagged these fields as vulnerable to XSS.
Impact:
Session Hijacking: Attackers can steal session cookies, allowing them to impersonate the victim and access sensitive information.
Defacement: Malicious scripts can modify the appearance and content of web pages.
Phishing: Users can be redirected to malicious sites designed to steal credentials or other personal information.
Malware Distribution: Injected scripts can download and install malware on the victim’s device.
Example Scenario:
An attacker crafts a URL containing a malicious script that prompts the user to enter login credentials. When the victim clicks on the link, the script executes, displaying a fake login page and stealing the entered credentials.

Server-Side Injection

Description: Server-side injection vulnerabilities occur when an application incorporates untrusted data into a command or query sent to an interpreter, without proper sanitization. This can alter the execution of the command or query, leading to unauthorized actions.
Mechanism: An attacker sends malicious input to the server, which is executed as part of a system command, database query, or other interpretable instruction. This can result in arbitrary code execution, data exposure, or complete system compromise.
Detection Method: Identified using Wapiti and Nmap scans. These tools highlighted insecure handling of user input in server-side scripts, where input data was directly used in command execution.
Impact:
Unauthorized Access: Attackers can gain unauthorized access to sensitive data and functionalities.
Data Leakage: Confidential information can be exposed or exfiltrated.
Full Server Control: Exploiting these vulnerabilities can give attackers control over the entire server, allowing for further exploitation and damage.
Denial of Service: Malicious commands can disrupt server operations, leading to downtime and loss of service.
Example Scenario:

An attacker injects a command into a vulnerable script that deletes critical files or shuts down the server. The injected command is executed by the server, resulting in loss of data and service disruption.

EXPLOITATION OF VULNERABILITY

Reflected cross-site scripting (xss) Exploitation
Method:
Identified a vulnerable input field in the web application that reflected user input in the response.
Crafted a URL containing a malicious script designed to steal session cookies.
Sent the URL to a target user, who clicked the link and executed the script.
Outcome:
The malicious script executed in the user's browser, capturing the session cookie.
The captured session cookie was sent to the attacker's server, allowing the attacker to hijack the user's session.
Impact Demonstration:
Showed the ease with which an attacker can steal sensitive information using reflected XSS.
Highlighted the potential for significant user data compromise and unauthorized account access.
Server-Side Injection Exploitation
Method:
Identified a vulnerable server-side script that took user input and executed it without proper sanitization.
Crafted a malicious input that included additional commands to be executed by the server.
Submitted the input via the vulnerable endpoint, causing the server to execute the injected commands.
Outcome:
The exploitation led to arbitrary command execution on the server, demonstrating the severity of the vulnerability.
Specific commands were used to disrupt server operations, causing the web application to become unavailable.
Impact Demonstration:
The server was brought down as a direct result of the exploitation, highlighting the potential for severe damage.
Showcased the ability of attackers to gain full control over the server, posing significant risks to data integrity and availability.

IMPACT ANALYSIS

Reflected cross-site scripting (xss)
Potential Damage:
Session Hijacking: Attackers can steal session cookies and impersonate users, leading to unauthorized access to sensitive data.
Defacement: Malicious scripts can alter the appearance of web pages, damaging the website’s

credibility.
Phishing: Users can be redirected to malicious websites designed to steal personal information.
Malware Distribution: Injected scripts can download and execute malware on the victim’s device.
User Trust: Reflected XSS can severely impact user trust, as users might be exposed to phishing attacks or have their sessions hijacked.
Legal and Compliance Issues: Organizations may face legal consequences if user data is compromised due to XSS vulnerabilities.
Server-Side Injection
Potential Damage:
Unauthorized Access: Attackers can gain access to sensitive information and functionalities, leading to data breaches.
Data Leakage: Confidential information can be exposed, causing financial and reputational damage.
Full Server Control: Exploiting this vulnerability can give attackers full control over the server, allowing for further exploitation.
Denial of Service: Malicious commands can disrupt server operations, leading to significant downtime and loss of service.
Server Downtime: The demonstrated exploitation resulted in server downtime, affecting the availability of the web application.
Financial Impact: Downtime and data breaches can lead to significant financial losses due to remediation costs, lost revenue, and potential fines.

RECOMMENDATIONS

Input Validation and Sanitization:
Implement robust input validation on both the client and server sides to ensure only valid data is accepted.
Use whitelisting techniques to define acceptable input formats and reject any data that does not conform.
Sanitize user input to remove or escape characters that could be used for XSS or injection attacks.
Escape User Input:
Properly escape user input in HTML, JavaScript, and SQL queries to prevent the execution of malicious scripts or commands.
Use libraries and frameworks that automatically escape user input, reducing the risk of human error.
Use Prepared Statements:
For database queries, use prepared statements and parameterized queries to separate user input from code execution.
This approach ensures that user input is treated as data and not executable code, mitigating the risk of SQL injection.
Regular Security Audits:
Perform regular security assessments using automated tools and manual testing to identify and remediate vulnerabilities.
Conduct periodic penetration tests to simulate real-world attacks and evaluate the effectiveness of security measures.
Keep all software and dependencies up to date with the latest security patches to protect against known vulnerabilities.
Web Application Firewall (WAF):
Deploy a WAF to help filter out malicious traffic and detect common web application attacks.
Configure the WAF to block requests that match patterns associated with XSS, SQL injection, and other common vulnerabilities.
Secure Coding Practices:
Train developers on secure coding practices and common vulnerabilities, such as those outlined in the OWASP Top Ten.
Incorporate security checks into the development lifecycle, including code reviews, static analysis, and

dynamic testing.
Establish a secure development environment with proper access controls and monitoring.

CONCLUSION

The security analysis of the web application revealed critical vulnerabilities, specifically Reflected XSS and Server-Side Injection. The successful exploitation of these vulnerabilities highlighted the potential for severe damage, including server downtime and user data compromise. Implementing the recommended security measures will help mitigate these risks and enhance the overall security posture of the web application. Regular security assessments, secure coding practices, and the use of protective technologies such as WAFs are essential for maintaining a robust security environment.

REFERENCES

OWASP Top Ten: A list of the most critical security risks to web applications, maintained by the Open Web Application Security Project (OWASP).
: The official website for Nmap, providing documentation, download links, and resources.
Wapiti Documentation: Official documentation for Wapiti, including usage instructions and feature descriptions.
Burp Suite Documentation: Comprehensive documentation for Burp Suite, covering all tools and features.