1 Bachelor In Computer Science Network Security Department of Computer Science 20212025 Prof Faisal Iradat Nosherwan Sheikh (25237) Tabish Imran (17905) Friday17th November, 2023 Karachi, Sindh

Network Security Project

Shahmeer Khan (25156)

December 12, 2023

A03:2021 – Injection
To my University..

Abstract

This report specifies the various processes and techniques used in gathering requirements, implementing and testing for the project on ‘A03:2021 – Injection’ which deals with one of the vulnerabilities from OWASP top 10 ie Injection. This project aims to use BURP suite to test the vulnerabilities and display the results of various tests.

Contents

0.1 Introduction

We will focus on the Injection Vulnerability by performing BURP SUITE tests on possible attacks and flaws, test result findings will be displayed accordingly, the aim is to further innovate and discover in the field of network security and also emphasize the benefit of BURP SUITE for network testing for candidates that have just gathered an interest in the field.

0.2 SQL Injections and the threat they propose

One significant and frequent kind of web application security flaw is SQL injection. Attackers can manipulate input to insert malicious SQL code when an application fails to properly validate or sanitise user input before utilising it in SQL queries. This may result in data tampering, unauthorised database access, and in certain situations, total system compromise.

SQL injection's risks include:

Unauthorised Entry:
SQL injection is a tool that attackers can use to get around authentication restrictions and access private information without authorization. Passwords, usernames, and other private information may fall under this category.

Data manipulation:
Data loss or corruption can result from the use of SQL injection to add, remove, or modify data in databases. This is especially worrisome when handling sensitive or important data.

Code Execution:
SQL injection may occasionally be used to run arbitrary code on the server, which could result in the host system being fully compromised.

0.3 Why are these issues overlooked?

There are a number of technological and organisational reasons why security flaws, such as SQL injection vulnerabilities, are often ignored. These are some explanations for why these problems might go unnoticed:

Budget limit:
Security measures often require additional resources in terms of both time and money. Organizations with limited resources may use resources to meet functional requirements, ignoring security considerations.

False beliefs regarding frameworks:
Security is not guaranteed if a specific programming language or framework is all that is used. Developers may fail to implement security properly because they believe that utilising a well-known framework renders their application automatically secure.

Lack of Security Education:
It's possible that many developers don't get enough instruction or training on safe coding techniques. Developers run the risk of unintentionally introducing vulnerabilities if they lack a thorough awareness of potential security risks.

0.4 Why choose Injections

SQL Injection is a malicious technique and poses serious security threats however it is not given the emphasis it deserves due to which attackers often exploit its effectiveness. Additionally, SQL Injection attacks can be particularly lethal because they exploit weaknesses in authentication mechanisms, for instance, by injecting a payload like '101 OR 1=1' into a login form, an attacker can manipulate the SQL query to always evaluate to true, effectively bypassing authentication checks and not needing a password, with this the attacker has access to all your protected data and with the fact that data is now more valuable than money itself, the damage can be critical. Another reason we chose SQL injections in particular is that we currently have a database course and we are working on a web app connecting to a database. By learning more about SQL injections we can find ways of securing not only our own , but databases all over the world, we hope to bring light to the damage that can be done by this and thus highlight the importance of security against it.

0.5 Why choose Burp Suite

Burp Suite is an ideal tool for testing SQL Injection vulnerabilities due to its features and user-friendly interface. It is particularly effective in identifying and exploiting SQL Injection weaknesses in web applications. Its scanning capabilities make it an excellent choice for testing vulnerabilities like SQL Injection. Its the ideal choice for beginners at network security and furthermore PortSwigger and Youtube provide several helpful tutorials for testing therefore leading to a better understanding of the topic at hand.

0.6 Work Breakdown/Methodology

1.Install Burp Suite and configure proxy settings.
2.Explore the OWASP web application to identify potential injection points.
3.Perform manual testing using Repeater.
4.Perform automated Testing using intruder.
5.Perform error-based SQL injection testing.
7.Perform time-based and boolean-based SQL injection testing.
8.Use Burp Scanner for automated vulnerability identification.
9.Document findings and prepare a detailed report.
10.Suggest possible solutions and workarounds against these vulnerabilities.

0.7 Inclusions and Exclusions

Inclusions
.Manual testing of user input fields.
.Automated testing using Burp Intruder.
.Use of various SQL injection techniques (error-based, time-based, boolean-based).
.Utilization of Burp Scanner for automated vulnerability scanning.
.Documentation of successful and unsuccessful attempts.
.Reporting on identified vulnerabilities.
.More efficient solutions/workarounds to deal with vulnerabilities

Exclusions
.Full-scale penetration testing beyond SQL injection.
.Vulnerabilities other than SQL Injections.
.Testing on websites other than OWASP web app.

0.8 Functional Requirements(hw/sw)

.Burp Suite installation and proper configuration.
.OWASP web application being functional for testing.
.Internet connectivity for resources,testing and documentation.
.Sufficient computing resources for efficient testing.

Hardware Requirements .Computer with enough memory and processing ability for efficient testing

Software Requirements .Burp Suite Community version , could switch to professional version if needed
.Web browser configured to use Burp as a proxy
.Access to OWASP web app for testing

0.9 Vulnerability Data Analysis Methodology

.Categorize vulnerabilities based on severity(low risk to high risk)
.Prioritize vulnerabilities that pose the highest risk.
.Verify and validate each identified vulnerability.
.Provide recommendations for possible new solutions or work on making existing solutions more efficient
.Provide images and results for every test run

0.10 FrameWork

OAWSP top ten (SQL Injection):
Role: Use OWASP top ten projects to check for SQL Injection and related CWE's
Incorporation: Integrate these CWE's into our research as base points to test

OWASP web security Testing Guide (WSTG):
Role: Utilize WSTG as a comprehensive guide for testing web applications, covering various security aspects.
Incorporation: Integrate WSTG into the testing framework to ensure well thought out testing procedures.

Portswigger's guide for testing vulnerabilities via BURP Suite:
Role: Use Portswigger's guide to effectively use BURP Suite for vulnerability testing.
Incorporation: Follow Portswigger's guide to align BURP Suite testing with best practices and techniques.

0.11 Methodology

Defining Objectives:
Objective: Clearly define the goal of the project.
Activities: Understand the overall security objectives of the OWASP web app with regards to SQL Injection
Define specific goals for SQL injection testing.

Scope Definition:
Objective: Clearly define the scope of the testing.
Activities: Identify the specific functionalities and areas within the OWASP app to be tested for SQL injection vulnerabilities.
Set boundaries for the testing scope.

Resource Allocation:
Objective: Allocate necessary resources for testing.
Activities: Ensure the availability of tools for eg including Burp Suite, for the testing process.

Test Planning:
Objective: Develop a comprehensive plan for SQL injection testing.
Activities: Create a test plan outlining the testing approach and tools.
Define roles and responsibilities for testing team members.

Execution:
Objective: Execute the defined testing plan.
Activities: Perform information gathering on the OWASP app using Burp Suite.
Conduct threat modeling to identify potential SQL injection points.
Develop and execute SQL injection test cases.

Analysis and Validation:
Objective: Analyze test results and validate findings.
Activities: Analyze error messages and responses for indications of SQL injection vulnerabilities.
Validate identified vulnerabilities to ensure they are not false positives.

Reporting:
Objective: Document and communicate the testing results.
Activities: Generate a comprehensive report detailing identified SQL injection vulnerabilities, their severity and categorization.
Provide evidence and documentation to support findings.

0.12 Process

Information Gathering:
Get familiar with the OWASP web app and understand its architecture, endpoints, and user inputs.

Threat Modeling:
Identify potential SQL injection points by analyzing user inputs, parameters, and data flow.

Burp Suite Configuration:
Configure Burp Suite on our system to intercept and analyze traffic.

Test Case Development:
Develop SQL injection test cases covering various techniques and contexts.
Create realistic use cases involving SQL queries to simulate user interactions.

Injection Testing:
Use Burp Suite's tools (Intruder, Repeater) for manual injection testing.

Error Handling Analysis:
Investigate error messages returned during injection attempts for clues about vulnerable points.

Authentication Bypass Testing:
Verify if SQL injection can lead to unauthorized access by bypassing authentication mechanisms.

Data Extraction Testing:
Check if it's possible to extract sensitive information from the database using SQL injection.

Logging and Reporting:
Log all testing activities, including successful injections, false positives, and issues encountered.
Generate a detailed report outlining identified vulnerabilities, their impact.

Post-Assessment:
Conduct a post-assessment to categorize the vulnerabilities based on severity , commonness and other variables.

0.13 Tests (will be decided later on)

:Test1 :Test2 :Test3 etc

0.14 Use Cases

Will be defined after testing takes place

List of Figures

List of Tables

chapters/bibliography