HACKYBARA

Requester Inc

Web Application Penetration Test Report

July 20, 2023

Submitted by: User

Contents

Document Information

This document contains information retrieved by an independent contractor using the Hackybara platform. It is provided to Requester Inc in accordance with the terms of any agreement between the independent contractor and Requester Inc. Information within this document contains sensitive information related to Requester Inc and should not be released to another vendor, business partner or contractor without written approval from Requester Inc. Furthermore, this document may not be copied, distributed, reproduced, or retained by the independent contractor without approval from Requester Inc.

The contents of this document does not constitute legal advice. The independent contractor using Hackybara’s platform offer of services or deliverables that relate to compliance, litigation, or other legal interests is not intended as legal counsel and should not be taken as such.

Contacts

Hackybara Platform Penetration Tester		
Name	**Role**	**Contact Information**
John Jacob	Hackybara Independent Contractor (HIC)	Johnjacob1899@gmail.com

Requester Inc		
Name	**Role**	**Contact Information**
Bob Cyber	Application Coordinator	bobcyber@gmail.com
Alice Cyber	Security Officer	Aliceccyber@gmail.com

Scope

The penetration test was performed against the following Requester Inc application:

Address/URL	Visibility
192.168.193.0/24	Requester Inc external network

Accounts Utilized

The following accounts were used during the assessment:

Account	Account Type
BOBUSER	User
BOBADMIN	Administrator
ALICEREQUESTER	Business Account
ALICEBUILDER	Developer

Executive Summary

Requester Inc contacted Hackybara to perform a Web Application Penetration Test of its Blog application. A Web Application Penetration Test is intended to identify security risks, vulnerabilities, needed best security practices, impact of vulnerabilities, document all security findings in a descriptive and clear manner, and provide mitigation recommendations. The assessment took place between July 1st and July 20th, 2023, and was conducted remotely.

During the assessment, HIC identified four (4) findings, including four (4) high risk vulnerabilities.

Conclusions

Overall, The HIC observed that the application exhibited some security controls but identified a pattern of weak permission checks and input sanitization that ultimately led to the access of user accounts and disclosure of sensitive information, this affected the confidentiality and integrity of the application.

The first very high-risk finding stems from a SQL injection vulnerability allowing access to a MySQl database. This ultimately allows an attacker to enumerate account information and extract sensitive information. Consultants were able to examine information from multiple databases.

The second high risk finding describes a cross site scripting vulnerability which could allow for arbitrary code to be run on the application. Cross site scripting (XSS) can be dangerous because it can allow for malicious scripts to possibly be run on an application resulting in the compromise of multiple security components.

The third high risk finding describes Unencrypted Communications aspects of the application. Consultants found that passwords were only encoded with base64 encoding which can easily be decoded. By decoding these passwords, consultants were able to view the clear text sent passwords.

The last high risk was weak authentication controls. A user account was able to be discovered via brute force attempts. This type of vulnerability is very serious because it can lead to multiple accounts being compromised.

Recommendations

Recommend Mitigation Factors

SQL injection vulnerability

Prevent SQL Injection attacks by ensuring that all user input is properly validated and that every single database call utilizes prepared statements with parameterized queries or securely coded stored procedures.

Prevent Use of Plaintext Credentials or Base64 encoding

Do not store credentials in plain text or through reversible methods.

Decrease likelihood of successful XSS

Enforce better sanitation of infection areas to reduce the threat of XSS.

Increase authentication requirements.
Make users have a stronger password policy to reduce brute forcing attempts.

Tools used

- **SQLMap**
- **BurpSuite**

Technical Summary

During the assessment the HIC identified four (4) findings, including two (4) high risk vulnerabilities. The HIC observed that the application lacked sanitization on key areas of user input that resulted in the disclosure of sensitive information.

- *SQL Injection* – The application did not properly validate user supplied data included within a dynamic SQL query. Consequently, attackers could extract and modify sensitive information stored within the backend SQL database, including information from other applications.

- *Stored Cross-Site Scripting* – HIC identified multiple instances of persistent cross-site scripting (XSS), allowing attackers to execute arbitrary JavaScript within the context of GIPM user browsers. A common target is the session identifier, allowing attackers to compromise authenticated sessions and impersonate victims within the application.

- *Weak Encryption* - Unencrypted Communications in the application. Consultants found that passwords were not encoded/encrypted. By having plain text passwords, consultants were able to view the clear text sent passwords. By examining the get request of the login page, the password was seen. If an attacker was examining sent packets, such as in a man in the middle attack, an attacker could steal user credentials allowing them to compromise user accounts.

- *Weak Authentication* - Authentication requirements for the application are weak and easily vulnerable to brute forcing techniques. Due to the length and lack of complexity required for user accounts, cracking user ids and passwords is considerably simple.

Index of Findings

Finding Title	**Severity Rating**	**CVSS Score**
1.SQL Injection	HIGH	8.8
2. Reflected Cross-Site Scripting	HIGH	7.2
3. Unencrypted Communications	HIGH	7
4. Weak Authentication	HIGH	7

1. SQL Injection			
Location	https://www.requesterinc.com/SQLexamplepage.php?		
CWE	CWE-89: Improper Neutralization of Special Elements used in an SQL Command (SQL Injection)		
Impact	**HIGH**	CVSS	**8.8**

Details

HIC identified SQL injection (SQLi) within the web application. The application did not properly validate user supplied data included within a dynamic SQL query. Consequently, attackers could inject arbitrary statements into the generated SQL query. HIC was able to exploit the SQL injection vulnerability to extract sensitive information from the backend database server. The following screenshots demonstrate exploitation of the SQL injection vulnerability.

Figure 1. Page with SQL injection and Burp parameter

General Vulnerability Information

SQL injection vulnerabilities occur when applications construct SQL queries dynamically using user-provided input without appropriate sanitization of those inputs. This allows malicious actors to manipulate the SQL statement's, potentially jeopardizing the confidentiality and integrity of the data stored in the database.

Impact

Attackers could extract and modify sensitive information stored within the backend SQL database, including information from other applications. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system.

Recommendation

Implement either prepared statements with parameterized queries or securely coded stored procedures to prevent variables from being interpreted within SQL queries.

In addition, implement thorough input validation in order to remove dangerous characters from user supplied data. It is essential to remove or escape the single quote character (').

Please refer to the following resources for more information regarding SQL injection prevention:

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

2. Stored Cross-Site Scripting			
Location	https://www.requesterinc.com/crossexamplepage.asp?		
CWE	CWE-79: Improper Neutralization of Input During Web Page Generation (Cross-site Scripting)		
Impact	**HIGH**	CVSS	**8.8**

CONTINUED