We Test Pens Incorporated

COMP90074 - Web Security Assignment 2

Jiahao Chen

1118749

**PENETRATION TEST REPORT FOR**

**InHR - WEB APPLICATION**

**Report delivered: 13/05/2023**

# Executive Summary

Introduction

This report to aims to present the main discovered vulnerabilities in the new electronic HR system of PleaseHold Pty. Ltd., developed by HRHub. Penetration tests are conducted using Burp Suite and Python, with the request frequency limited to 30 payloads per minute. Main vulnerabilities are listed as follows.

Vulnerabilities

*1. SQLI in Find User*

There is a high risk of system compromise and data leakage due to the SQL injection vulnerability in the “Find User” page. Despite the “User search” function returning only boolean results, attackers can use blind SQL injection techniques to gain unauthorised access to the entire database, including highly sensitive information.

*2. XSS in Anonymous Question*

The “Anonymous Question” page is susceptible to XSS vulnerabilities. Malicious JavaScript code can be inserted in the question and sent to the HR team. If the code is successfully executed, attackers can perform unauthorised operations instead of the HR team.

*3. SSRF in User Profile*

The server is exposed to the risk of data leakage due to the SSRF vulnerability in the “User Profile”. The “VALIDATE WEBSITE” accepts a URL link as a parameter, which makes it possible for attackers to detect other available services and get access to unauthorised data.

*4. SQL Wildcard Attack*

The provided store API is vulnerable to SQL wildcard attack. Attackers can get access to all the data in the store.

Security Posture Assessment:

The above indicate that the new HR system is currently exposed to a couple of security vulnerabilities, which may lead to server disruption to the company’s management. Corresponding mitigations will be discussed in detail. It is highly recommended to implement suitable remediations before the public launch to avoid severe consequences.

Table of Contents

[Executive Summary 2](#_Toc134857087)

[Summary of Findings 4](#_Toc134857088)

[Detailed Findings 5](#_Toc134857089)

[Finding 1 – SQLI in Find User 5](#_Toc134857090)

[**Description** 5](#_Toc134857091)

[**Proof of Concept** 5](#_Toc134857092)

[**Impact** 5](#_Toc134857093)

[**Risk Ratings** 5](#_Toc134857094)

[**Recommendation** 5](#_Toc134857095)

[**References** 5](#_Toc134857096)

[Finding 2 – XSS in Anonymous Question 6](#_Toc134857097)

[**Description** 6](#_Toc134857098)

[**Proof of Concept** 6](#_Toc134857099)

[**Impact** 6](#_Toc134857100)

[**Risk Ratings** 6](#_Toc134857101)

[**Recommendation** 6](#_Toc134857102)

[**References** 6](#_Toc134857103)

[Finding 3 – SSRF in User Profile 7](#_Toc134857104)

[**Description** 7](#_Toc134857105)

[**Proof of Concept** 7](#_Toc134857106)

[**Impact** 7](#_Toc134857107)

[**Risk Ratings** 7](#_Toc134857108)

[**Recommendation** 7](#_Toc134857109)

[**References** 7](#_Toc134857110)

[Finding 4 – SQL Wildcard Attack 8](#_Toc134857111)

[**Description** 8](#_Toc134857112)

[**Proof of Concept** 8](#_Toc134857113)

[**Impact** 8](#_Toc134857114)

[**Risk Ratings** 8](#_Toc134857115)

[**Recommendation** 8](#_Toc134857116)

[**References** 8](#_Toc134857117)

[Appendix I – Risk Matrix 9](#_Toc134857118)

[Appendix II – Additional Information 9](#_Toc134857119)

# Summary of Findings

A brief summary of all findings appears in the table below, sorted by risk rating.

|  |  |  |
| --- | --- | --- |
| **Risk** | **Reference** | **Vulnerability** |
| Extreme | [Finding 1](#_Finding_1_–) | The “Find User” page is vulnerable to SQL injection, thereby posing a risk of data leakage. |
| High | [Finding 2](#_Finding_2_–) | The “Anonymous Question” page may be exploited by non-administrative users using XSS. |
| Medium | [Finding 3](#_Finding_3_–) | Server port’s availability may be disclosed through the “User Profile” page, which could lead to SSRF. |
| Medium | [Finding 4](#_Finding_4_–) | The store’s API is vulnerable to SQL wildcard attack, which could lead to data leakage. |

# Detailed Findings

This section provides detailed descriptions of all the vulnerabilities identified.

## Finding 1 – SQLI in Find User

|  |  |
| --- | --- |
| **Description** | There is a high risk of system compromise and data leakage due to the SQL injection vulnerability in the “Find User” page. Despite the “User search” function returning only boolean results, attackers can use blind SQL injection techniques to gain unauthorised access to the entire database, including highly sensitive information. |
| **Proof of Concept** | [Check Appendix II – SQLI in Find User](#_SQLI_in_Find) |
| **Impact** | An attacker could get access to the entire database with this vulnerability, including usernames and passwords. With such unauthorised access, the attacker would be able to perform malicious actions, for example, taking control of admin accounts and passing probation for arbitrary users, which could lead to significant disruption within the company’s internal management. |
| **Risk Ratings** | **Extreme**  This vulnerability not only poses a catastrophic risk of sensitive data leakage but also has the potential to disrupt the company's management processes. Meanwhile, exploiting blind SQL injection can be time-consuming, as it requires numerous requests to retrieve sensitive data. Therefore, it may not be highly likely to exploit this vulnerability due to the complexity, but still possible. |
| **Recommendation** | Usernames could be not allowed to include special characters like ' and #, reducing the possibility of SQLI. Whitelist or blacklist could also be applied for further validation and sanitisation of user inputs. Prepared statements or stored procedures may also be useful as the “User search” function should only query on usernames. Besides, database like “Secure” should require higher permission to be accessed to protect sensitive data.  Rate control should be implemented by monitoring number of requests. Abnormal IP addresses or accounts should be blocked to reduce the possibility of blind SQLI as well as DOS. For example, `$\_SERVER['REMOTE\_ADDR']` can get the client’s IP address in PHP. |
| **References** | Lecture Slides |

[Back to Summary of Findings](#_Summary_of_Findings)

## Finding 2 – XSS in Anonymous Question

|  |  |
| --- | --- |
| **Description** | The “Anonymous Question” page is susceptible to XSS vulnerabilities. Malicious JavaScript code can be inserted in the question and sent to the HR team. If the code is successfully executed, attackers can perform unauthorised operations instead of the HR team. |
| **Proof of Concept** | [Check Appendix II – XSS in Anonymous Question](#_XSS_in_Anonymous) |
| **Impact** | An attacker could perform unauthorised operations on behalf of the HR team by inserting malicious payload to the question. For example, the attacker can manipulate the system to pass probation for any user whose username is known. Such unauthorized actions could cause significant confusion and disruption within the company's management processes. |
| **Risk Ratings** | **High**  Although the current consequence of the vulnerability is considered moderate, as it only allows for the modification of a user's probation status, it is crucial to recognize the potential risks. There is a high likelihood that this vulnerability could be discovered by attackers.  Moreover, considering potential future extensions for the system, for instance, the HR team is able to change user passwords, the consequences of the vulnerability could escalate to a more severe level. |
| **Recommendation** | Basic mitigation strategies for XSS should be implemented, including input validation and sanitisation, input encoding and applying filters.  The best solution is to implement a double authentication for operations requiring high-level permission to ensure that they are indeed performed by the HR team. One way is to integrating CAPTCHA to avoid requests made by bots. Another way is to use third-party authenticators, such as Okta, to double check the identity of the requester. |
| **References** | Lecture Slides |

[Back to Summary of Findings](#_Summary_of_Findings)

## Finding 3 – SSRF in User Profile

|  |  |
| --- | --- |
| **Description** | The server is exposed to the risk of data leakage due to the SSRF vulnerability in the “User Profile”. The “VALIDATE WEBSITE” accepts a URL link as a parameter, which makes it possible for attackers to detect other available services and get access to unauthorised data. |
| **Proof of Concept** | [Check Appendix II – SSRF in User Profile](#_SSRF_in_User) |
| **Impact** | An attacker could scan over all the ports of the server and detect active ones, which leads to leakage of private data stored in the server side. Such vulnerability may lead to severe consequences if sensitive data are disclosed. |
| **Risk Ratings** | **Medium**  Currently, the consequence of this vulnerability is moderate as there are no sensitive data discovered. However, it is still of high risk to expose the server to attackers.  Meanwhile, the possibility of SSRF attack is not as likely as other attacks like XSS, as it requires a huge amount of time to discover active ports. Nonetheless, once the active ports are found, the attackers can keep track of them and retrieve data at any time. |
| **Recommendation** | Avoid passing URL link as parameters, if necessary, sensitive links such as localhost should put in the blacklist.  When private ports are accessed, double authentication could be applied to check the source of request. For example, if it is from “VALIDATE WEBSITE”, the access should be denied.  Rate control is necessary as scanning ports requires a huge number of requests. Similar to the suggestions for mitigating SQLI, abnormal activities should be monitored and blocked, which could reduce the possibility of exposing active ports to attackers. |
| **References** | Lecture Slides |

[Back to Summary of Findings](#_Summary_of_Findings)

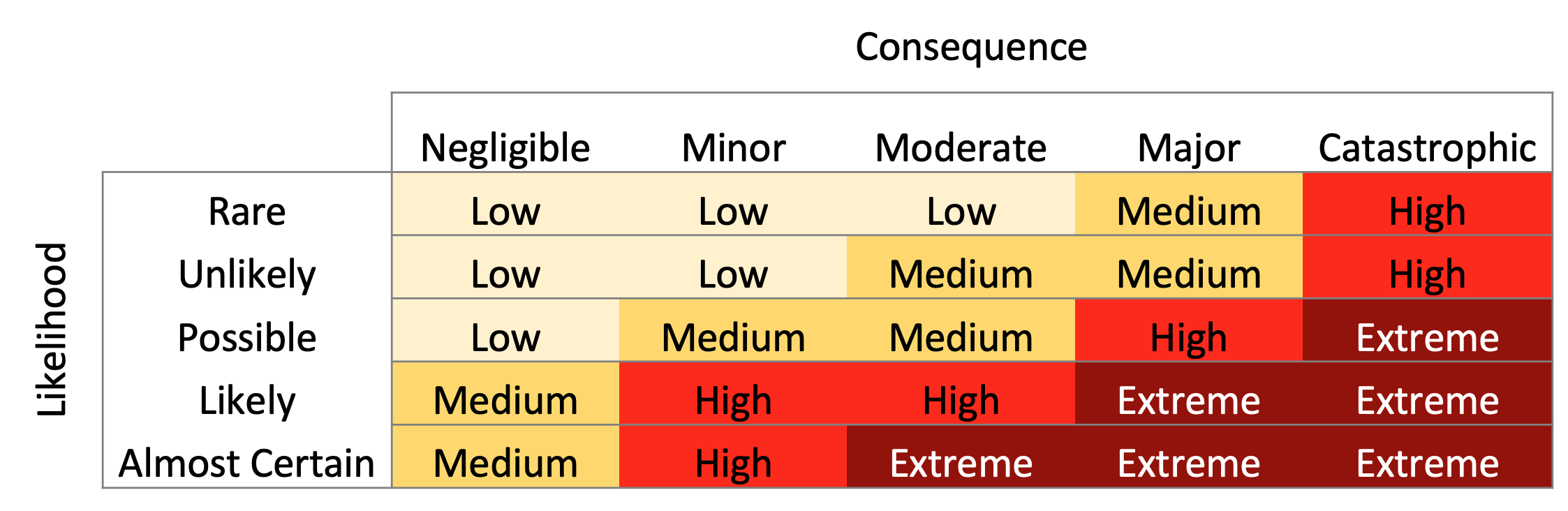
## Finding 4 – SQL Wildcard Attack

|  |  |
| --- | --- |
| **Description** | The provided store API is vulnerable to SQL wildcard attack. Attackers can get access to all the data in the store. |
| **Proof of Concept** | [Check Appendix II – SQL Wildcard Attack](#_SQL_Wildcard_Attack) |
| **Impact** | An attacker could retrieve all the data in the store by simply passing ‘%’ as the parameter. Sensitive data could be easily leaked if stored there. |
| **Risk Ratings** | **Medium**  The endpoint stores different types of training with no sensitive data. Therefore, the consequence is currently negligible. However, the wildcard attack is almost certain to be discovered and performed by attackers. |
| **Recommendation** | Wildcard operators like ‘%’ are necessary to be banned from user inputs. Note that the “User search” function does have this mechanism to check such operators.  Moreover, abnormal accounts should be monitored. For example, if an account performs such attack, it could be banned from using the API. |
| **References** | Lecture Slides |

[Back to Summary of Findings](#_Summary_of_Findings)

# Appendix I – Risk Matrix

All risks assessed in this report are in line with the ISO31000 Risk Matrix detailed below:

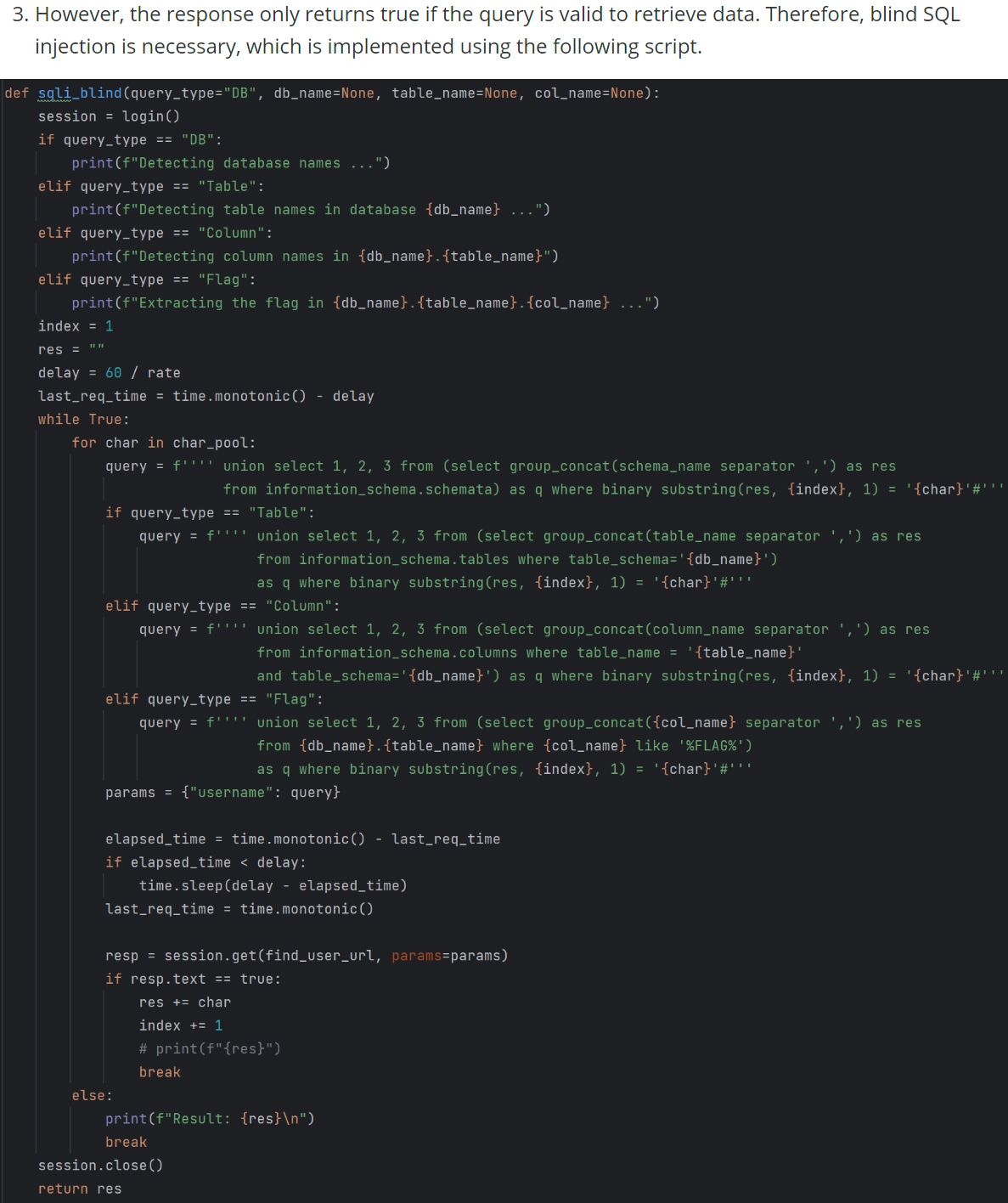


# Appendix II – Additional Information

# SQLI in Find User

A screenshot of a computer program

Description automatically generated with medium confidence



A screenshot of a computer screen

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer screen

Description automatically generated with low confidence

[Back to Finding 1](#_Finding_1_–)

# XSS in Anonymous Question

A screenshot of a computer

Description automatically generated with medium confidence

A screen shot of a computer program

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated

[Back to Finding 2](#_Finding_2_–)

# SSRF in User Profile

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer screen

Description automatically generated with medium confidence

A screenshot of a computer screen

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with medium confidence

[Back to Finding 3](#_Finding_3_–)

# SQL Wildcard Attack

A screenshot of a computer program

Description automatically generated with medium confidence

[Back to Finding 4](#_Finding_4_–)

[Back to Summary of Findings](#_Summary_of_Findings)