**Union Bank of India**

**Web Application Penetration Test Report**

Version 1.0

August 12, 2024

Statement of Confidentiality

This Confidential Information is being provided to **Union Bank of India** as a deliverable of this consulting engagement. The sole purpose of this document is to provide you with the results of, and recommendations derived from this consulting engagement. Each recipient agrees that, prior to reading this document, it shall not distribute or use the information contained herein and any other information regarding **ControlCase** for any purpose other than those stated.

Table of Contents

[1 Executive Summary 5](#_Toc82229103)

[1.1 Introduction 5](#_Toc82229104)

[1.2 Goals & Objectives 5](#_Toc82229105)

[1.3 Approach & Methodology 6](#_Toc82229106)

[1.4 Project Team – Contact Information 7](#_Toc82229107)

[1.5 Penetration Timeline 7](#_Toc82229108)

[1.6 Target Description 8](#_Toc82229109)

[1.7 Summary of Observations 9](#_Toc82229110)

[1.8 Statement on Compliance 10](#_Toc82229111)

[2 Detailed Observations 11](#_Toc82229112)

[2.1 Overview 11](#_Toc82229113)

[2.2 Vulnerability Table 11](#_Toc82229114)

[2.3 Vulnerability Discovery Phase 13](#_Toc82229115)

[2.3.1 Host Header Injection 14](#_Toc82229116)

# Executive Summary

## Introduction

**Union Bank of India** engaged **ControlCase** to conduct a Penetration Test of their **Union Cards** Web Application.The purpose of the engagement was to identify and prioritize the potential areas of security vulnerability.

The engagement began on **July 25, 2024** and included multiple phases of testing, analysis, and documentation. All testing was conducted from ControlCase Lab.

This document summarizes the analysis, observations and recommendations for the assessment carried out by **ControlCase**.

## Goals & Objectives

The purpose of this assessment was to identify technical as well as logical vulnerabilities in the application and provide recommendations for risk mitigation that may arise on exploiting these vulnerabilities. The idea behind this testing was to discover whether an attacker can leverage flaws in the application to compromise the confidentiality, integrity and availability of the information. **ControlCase** worked with **Union Bank of India** to achieve the following key objectives:

* To determine whether adequate information security controls have been built into the application.
* Perform supplemental research and development activities to support analysis.
* Prioritize vulnerabilities based upon the ease of exploit, level of effort to remedy, and severity of impact if exploited.
* Assess current practice against industry best practices.
* Deliver report which includes **ControlCase's** observations, analysis, and recommendations.
* Transfer knowledge.

## Approach & Methodology

ControlCase Application Security consultants follow the **OWASP** (**O**pen **W**eb **A**pplication **S**ecurity **P**roject) an established guideline in application security methodology. In the course of the assessment, ControlCase consultants use a variety of commercial, open-source tools as well as homegrown scripts & tools.

ControlCase has defined following approaches while doing application security assessment.

Black box testing – This is a technique to attempt to penetrate application where the source code of the application is not available to the tester. ControlCase team will attempt to elicit exception conditions and anomalous behavior from the Web Application by manipulating the identified inputs - using special characters, SQL keywords, maliciously crafted requests, and so forth. Any unexpected reaction from the Web Application is noted and investigated. This may take the form of scripting error messages, server errors or half-loaded pages. The goal of this method is to simulate an attack by an external hacker.

Grey box testing – This approach is similar to black box testing however the attack team is given the same privileges as a 'normal' and 'privileged' user of the application and the goal is to simulate an attack by a malicious insider. The attack team tries to escalate the privileges of a normal user to administrator user.

|  |  |  |
| --- | --- | --- |
| **Types of tests performed** | **Checked** | |
|  |  | |
| 1. Application Security Assessment Test |  | |
| * Automated scanning of possible web application vulnerability |  | 🗸 | |
| * Manual exploit on discovered vulnerability |  | 🗸 | |
| * Compliance Specific checks (e.g. PCI DSS) |  | 🗸 | |
|  |  |  | |
| 1. OWASP Top 10 2021 |  |  | |
| * Broken Access Control |  | 🗸 | |
| * Cryptographic Failures |  | 🗸 | |
| * Injection |  | 🗸 | |
| * Insecure Design |  | 🗸 | |
| * Security Misconfiguration |  | 🗸 | |
| * Vulnerable and Outdated Components |  | 🗸 | |
| * Identification and Authentication Failures |  | 🗸 | |
| * Software and Data Integrity Failures |  | 🗸 | |
| * Security Logging and Monitoring Failures |  | 🗸 | |
| * Server-Side Request Forgery |  | 🗸 | |

## Project Team – Contact Information

The engagement involved contributions from the following team members:

|  |  |
| --- | --- |
| ControlCase Team | Union Bank of India Team |
| Vishal Patil | Aayushi Agrawal |
| Swapnil Kothawade |  |

## Penetration Timeline

The following table outlines key milestones during the penetration test:

| Penetration Timeline | |
| --- | --- |
| Date | Milestone |
| July 25, 2024 | Start of Project |
| August 12, 2024 | Final Deliverable |

## Target Description

The application penetration testing was carried out on one application. The approach conducted was a black box testing followed by grey box testing to login into the application and test it. The application was hosted externally in the **Union Bank of India** environment.

**Technical Details of the Target**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Application URL** | **Tested Environment** |
|  | https://unioncards.unionbankofindia.co.in/ | Production |

## Summary of Observations

The tested application has one (1) low-risk vulnerabilities that an attacker can target or exploit. It is important to periodically check, review and modify the application logic if any kind of change is being applied to the application.

The graph below gives the status of severity of the vulnerabilities found during the Application Security Assessment.

|  |  |
| --- | --- |
| **Risk severity level** | **No of observations** |
| **High** | 00 |
| **Medium** | 00 |
| **Low** | 01 |
| **Total** | 01 |

***Given below is the Summary of the Observations:***

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Observations** | **Risk Level** |
|  | Host Header Injection | Low |

## Statement on Compliance

**ControlCase** has determined that **Union Bank of India's Union Cards** web application is **Compliant** with **ControlCase** validation requirement as mentioned in [section 1.3](#_Approach_&_Methodology).

# Detailed Observations

## Overview

The following format shows a typical vulnerability representation and provides in detail information of vulnerabilities discovered during Application Vulnerability Test.

## Vulnerability Table

|  |  |
| --- | --- |
| 1. **Vulnerability Title** | |
| **Risk Level** |  |
| **OWASP Category** |  |
| **Abstract** |  |
| **Ease of Exploitation** |  |
| **Impact** |  |
| **Recommendations** |  |
| **Substantiated Assessment** |  |
| **Affected URL** |  |
| **Note** |  |
| **Reference** |  |
| **CWE** |  |

* **Vulnerability Title – A short title that describes the vulnerability.**

The title bar for each vulnerability table is color coded for a quick identification of the risk level. Title bar color codes are as follows:

|  |  |
| --- | --- |
| **Risk Level** | **Description** |
|  | **High risk** vulnerability can be exploited by an attacker to gain full administrative access to the application or its underlying operating system. |
|  | **Medium risk** vulnerability reveals information about the application and its underlying infrastructure that can be used by an attacker in conjunction with another vulnerability to gain administrative control of the application or its underlying operating system. |
|  | **Low risk** vulnerability can result in enumeration of vital information held by or about the Application or its underlying operating system. |

* **OWASP Category –** Refers to OWASP top 10-2021 vulnerability category.
* **Abstract –** Describes the flaw or bugs that cause the vulnerability.
* **Ease of Exploitation –** Provides a metric for the skill level required to exploit the vulnerability. The categories are:

|  |  |
| --- | --- |
| **Metric** | **Skill-level** |
| Easy | Casual user |
| Medium | Computer-savvy individual |
| Hard | Determined hacker |

* **Impact –** Describes the possible business impact if this vulnerability is successfully exploited.
* **Recommendation –** Provides solutions or workarounds to mitigate the risk arising from this vulnerability.
* **Substantiated Assessment –** The evidence of the vulnerability being present, wherever possible, is provided in the form of screenshots.
* **Affected URL –** Provides URLs and respective parameters which are affected with that specific vulnerability
* **Note –** A brief description of how the vulnerability can be exploited by internal/external attacker or limitations for exploitation which may result in minimizing the risk of the reported vulnerability.
* **Reference –** It provides reference to outside resource such as OWASP, SANS etc.
* **CWE –** Provides Common Weakness Enumeration ID

## Vulnerability Discovery Phase

This phase has been completed successfully and below are vulnerabilities observed during the application penetration testing:

|  |  |
| --- | --- |
| Host Header Injection | |
| **Risk Level** | **Low** |
| **OWASP Category** | A03:2021-Injection |
| **Abstract** | Assessor observed that it is possible to inject host header and redirect user to any malicious website |
| **Ease of Exploitation** | Hard |
| **Impact** | An attacker can inject any host header pointing to any malicious website and redirects users to a malicious site that performs phishing and installs malware. |
| **Recommendations** | It is recommended that the web application should use the SERVER\_NAME instead of the Host header. It should also create a dummy vhost that catches all requests with unrecognised Host headers. This can also be done under Nginx by specifying a non-wildcard SERVER\_NAME and under Apache by using a non-wildcard serverName and turning the UseCanonicalName directive on. You can also match a whitelist of domains. |
| **Substantiated Assessment** |  |
| **Affected URL** | https://unioncards.unionbankofindia.co.in/ |
| **Reference** | https://owasp.org/Top10/A03\_2021-Injection/ |
| **CWE** | 644 |