

Penetration Testing Report

Redacted | External BlackBox Pentest

January 01, 2025

Technical Report

CONFIDENTIAL: The contents of this document are privileged and are intended solely for the addressee. This transmission is sent in trust, for the sole purpose of delivery to the intended recipient.

Table of Contents

1. Document Details 3

Tester Details 3

2. Executive Summary 4

2.1 Unrestricted File Upload leading to Command Execution 4

2.2 Multiple SQL Injections Leading Access to Backend Database 4

2.3 User Account Takeover Using Reset Password Functionality 4

2.4 OTP Brute-force to Register with any Mobile Number 4

2.5 Information Disclosure via Error Messages 4

Vulnerability Breakdown 5

Vulnerabilities Severity Ratings 6

3. Scope of Work 7

Activities (Web Application) 7

Activities (Network Infrastructure) 7

4. Findings & Observations 8

4.1 Unrestricted File Upload leading to Command Execution 9

4.2 Multiple SQL Injections Leading Access to Backend Database 11

4.3 User Account Takeover Using Reset Password Functionality 13

4.4 OTP Brute-force to Register with any Mobile Number 14

4.5 Information Disclosure via Error Messages 15

4.6 Enabled Directory Listing in Uploads Directory 16

5. Fix Validation Report 18

5.1 Unrestricted File Upload leading to Command Execution 18

5.2 User Account Takeover Using Reset Password Functionality 18

6. Conclusion 19

7. Attestation 20

1. Document Details

Client Name	Redacted Client
Document Title	Pentest Report – External Infrastructure BlackBox Pentest
Date	January 01st, 2025
Classification	**Confidential**
Assessment Type	Remote

Tester Details

Name	Certifications & Accreditations	Contact
Pen Tester	Member of CREST, OSCP Certified	contact@penvasecurity.com.au

2. Executive Summary

A Black Box Pentest Activity was performed on a redacted web application and infrastructure. The purpose of the engagement was to evaluate the security posture of the Redacted applications and networks against best practice criteria, to validate security mechanisms, and identify vulnerabilities at the infrastructure, application, and code logic level. Methodologies used were OWASP testing guide for web applications pentest and Pentest Execution Standard (PTES) for network pentest.

This report details the scope of tests conducted, all significant findings along with detailed recommendations.

2.1 Unrestricted File Upload leading to Command Execution

An unrestricted file upload vulnerability in edit profile image functionality was found, exploiting which an attacker can upload malicious PHP files to the webserver and execute arbitrary commands to take control of the server.

2.2 Multiple SQL Injections Leading Access to Backend Database

Insecure practices for querying MySQL database were found on multiple parameters of various API endpoints, exploiting which an attacker can dump the whole database and have unauthorized access to sensitive data.

2.3 User Account Takeover Using Reset Password Functionality

A malicious user can takeover any user's account via reset password functionality, enabling them to set a new password for a user account.

2.4 OTP Brute-force to Register with any Mobile Number

A user can bypass OTP verification and register with any mobile number by brute-forcing the OTP, sent to the provided mobile number.

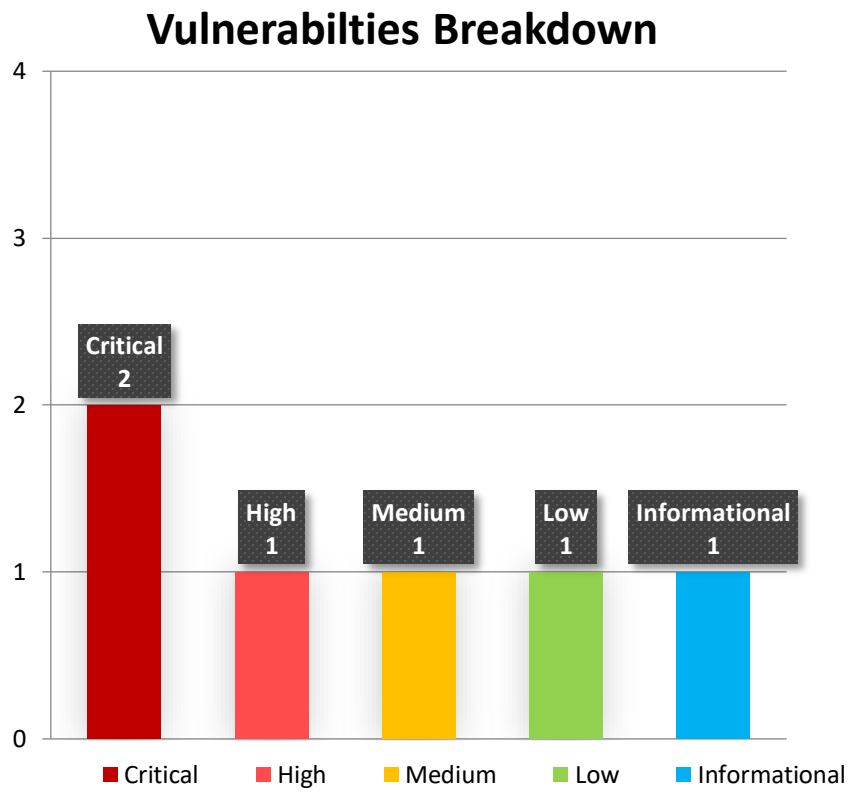
2.5 Information Disclosure via Error Messages

During the pentest, it was observed that the web application is displaying error messages in the response which could aid an attacker to perform further attacks.

Vulnerability Breakdown

During the penetration testing, the assessment uncovered two critical, one high, one medium-severity and one low-severity threat.

Category	Description				
Threat Level	Critical				
Total No. of Security Issues Identified	6				
Total No of Issues Fixed	2				
Vulnerability Breakdown Based on Severity	Critical	High	Medium	Low	Informational
	2	1	1	1	1



The following sections detail how vulnerabilities are classified and their potential impact when exploited. Additionally, CVSS 3.1 is used to assess the severity of detected vulnerabilities using a standardized scoring system.

Vulnerabilities Severity Ratings

The table below outlines the severity levels and corresponding CVSS score ranges used to evaluate vulnerabilities and their potential impact.

Severity	Score	Criteria
Critical	9.0 – 10.0	The vulnerability is extremely severe, often trivial to exploit, and can result in complete system compromise. Attackers may gain full control, execute arbitrary code remotely, or cause widespread service outages.
High	7.0 – 8.9	The vulnerability is significant and can be exploited easily in most environments. Attackers can gain substantial privileges, expose sensitive data, or cause major service disruptions.
Medium	4.0 – 6.9	The vulnerability can cause some damage, but it is not easily exploited or requires some level of user interaction. The impact may include data exposure, limited system access, or partial denial of service.
Low	0.1 – 3.9	The vulnerability has minimal impact and is difficult to exploit. Attackers may need special conditions, such as local access, user interaction, or uncommon configurations, to exploit the flaw.
Informational	N/A	The vulnerability poses no security risk. It does not affect confidentiality, integrity, or availability. Either it is purely informational or has no practical exploitability.

3. Scope of Work

The following domains and IPs were included in the scope:

#	Hosts	#	Hosts
1.	https://www.redacted.com	2.	http://internal.redacted.com
3.	https://test.redacted.com	4.	https://app.redacted.com
5.	http://mail.redacted.com	6.	209.28.xx.xx

Activities (Web Application)

1. Information Gathering
2. Identity Management Testing
3. Authentication Testing
4. Authorization Testing
5. Session Management Testing
6. Configuration and Deployment Management Testing
7. Input Validation Testing
8. Testing for Error Handling
9. Client-Side Testing
10. Business Logic Testing
11. Tech Specific Testing

Activities (Network Infrastructure)

1. Information Gathering
2. Network Reconnaissance
3. Network Enumeration
4. Vulnerability Scanning
5. Configuration Testing
6. Testing for default credentials
7. Active Directory Exploitation
8. Lateral Movement & Clean Up

4. Findings & Observations

The following table lists identified vulnerabilities, categorized by severity according to industry-standard CVSS 3.1, along with their fix status.

#	Vulnerability Identification	Severity	Status
1.	**Unrestricted File Upload leading to Command Execution**	Critical	Fixed
2.	**Multiple SQL Injections Leading Access to Backend Database**	Critical	Not Fixed
3.	**User Account Takeover Using Reset Password Functionality**	High	Fixed
4.	**OTP Brute-force to Register with any Mobile Number**	Medium	Not Fixed
5.	**Information Disclosure via Error Messages**	Low	Not Fixed
6.	**Enabled Directory Listing in Uploads Directory**	Informational	Not Fixed

4.1 Unrestricted File Upload leading to Command Execution

Severity	**Critical**
Status	Fixed
CVSS	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
Affected Component(s)	• https://redacted.com/fileupload.php (Edit Profile Image) Parameter: uploaded_image

Description

A critical unrestricted file upload vulnerability was identified during the penetration test, allowing arbitrary file extensions and content types to be uploaded to the server. Attackers could exploit this to upload malicious PHP scripts, enabling remote code execution (RCE), as demonstrated by executing system commands.

Proof Of Concept:

https://redacted.com/media/shell.php?command=uname -a (Command execution via uploaded malicious PHP file).

Impact

An attacker can upload malicious PHP files to the server, which may allow them to execute arbitrary commands. This could lead to full server compromise, potentially allowing attackers to gain unauthorized access to sensitive data, disrupt operations, or escalate their privileges within the environment.

Remediation

To mitigate this vulnerability, it is strongly recommended to implement strict file validation controls. Specifically, file uploads should be filtered using an allowlist to ensure only trusted file types are accepted. Additionally, the server should check both the file's extension and its MIME type to prevent malicious files from being uploaded. Finally, avoid relying solely on client-side validation, and ensure server-side checks are robust enough to handle any bypass attempts.

Reference(s):

- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- https://www.php.net/manual/en/features.file-upload.php

Evidence Image(s):



Figure 4.1: Create PHP File and executing phpInfo() function.

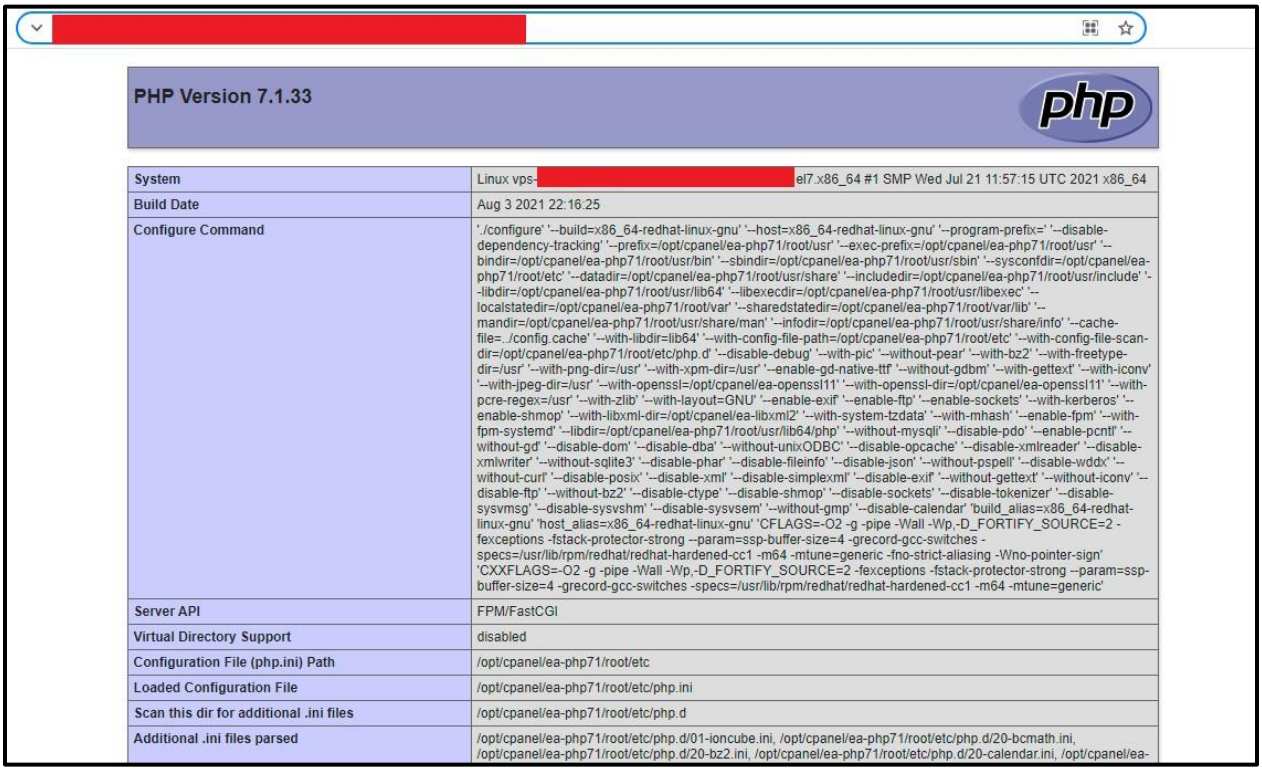


Figure 4.2: Accessing Phpinfo Page

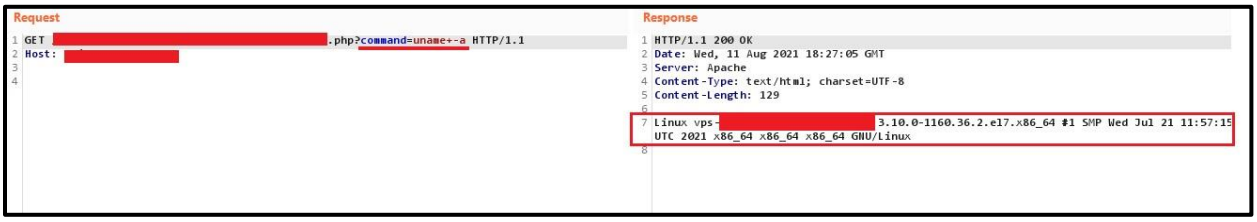


Figure 4.3: Executing Arbitrary Command on Server

4.2 Multiple SQL Injections Leading Access to Backend Database

Severity	**Critical**
Status	Not Fixed
CVSS	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:L
Affected Component(s)	• https://redacted.com/v1/company/get-companies Parameter: name • https://redacted.com/v1/user/getuser Parameter: username, uid

Description

SQL injection is a critical web security vulnerability that enables attackers to manipulate the queries an application makes to its database. This can result in unauthorized access to sensitive data, including information from other users or data the application itself can access.

During the penetration test, multiple instances of SQL injection were found across various input parameters, indicating a lack of proper input sanitization and insecure querying practices. These vulnerabilities allow attackers to access the backend database, potentially dump its contents, and gain unauthorized access to sensitive data.

The following payload was used to extract information from database:

```
https://test.redacted.com/v1/company/get-companies?name=205%22%20+Union+Select+1,+(CONCAT_WS(%27%20::%20%27,USER(),DATABASE(),VERSION()))+,3,4,5,6,7,8,9,10,11,12,13,14,15+%20--%20
```

Impact

An unauthenticated attacker could exploit these vulnerabilities to dump the entire database, compromising sensitive data and gaining unauthorized access to critical information stored within the system.

Remediation

To mitigate SQL injection risks, it is essential to sanitize all input parameters before making database queries. Employing secure querying practices, such as using parameterized queries, is crucial to preventing SQL injection. Additionally, ensure that database error messages are not exposed in production environments, as they could provide attackers with valuable insights into the database structure. Finally, the web application should operate with the minimum necessary database privileges to limit the potential impact of a successful attack.

Reference(s):

- https://www.w3schools.com/sql/sql_injection.asp
- https://www.netsparker.com/blog/web-security/sql-injection-vulnerability/

Evidence Image(s):

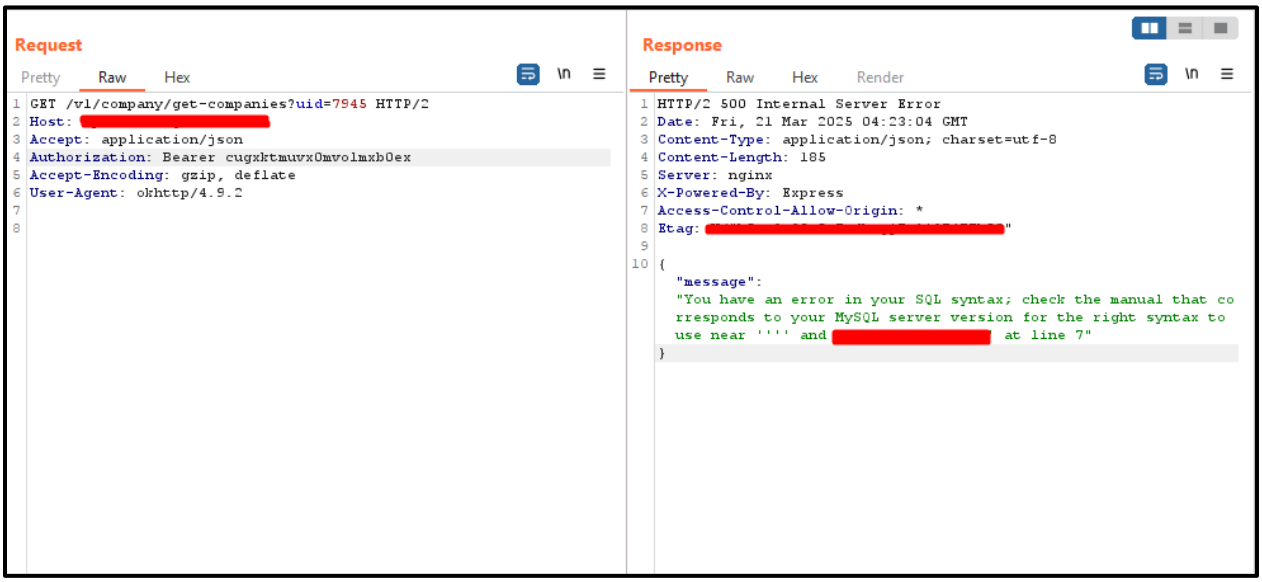


Figure 4.2.1: SQL Injection identified in uid parameter of updating user's data.

4.3 User Account Takeover Using Reset Password Functionality

Severity	High
Status	Fixed
CVSS	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N
Affected Component	• https://redacted.com/forget-password/ Parameter: email

Description

During the penetration test of the password reset functionality, it was discovered that the reset password token was being exposed in the server's response. This allows an attacker to capture the token and reset the user's password to their own choice. When a user requests a password reset for any email, the server includes the reset token in the response, which can be intercepted. An attacker could then use the token to reset the user's password, gaining unauthorized access to the account.

Impact

An attacker who obtains the reset token can reset any user's password, leading to a full account takeover and unauthorized access to sensitive information.

Remediation

To mitigate this vulnerability, sensitive information, such as the reset password token, should only be sent to the user via a secure email, not included in the server's response. This ensures that tokens are only accessible by the intended user, preventing unauthorized access.

Reference(s):

- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- https://www.php.net/manual/en/features.file-upload.php

Evidence Image(s):



Figure 4.3.1: Reset Password Token of a User Account Leaked in the Response.

4.4 OTP Brute-force to Register with any Mobile Number

Severity	**Medium**
Status	Not Fixed
CVSS	CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:N/A:N
Affected Component	• https://redacted.com/register Parameter: otp

Description

During the penetration test of the registration functionality, it was found that users are required to enter a valid mobile number and an OTP sent to that number. However, the system was vulnerable to brute-force attacks on the 5-digit numeric OTP. This allows an attacker to attempt multiple combinations and bypass OTP verification, registering with any mobile number of their choice.

Impact

A malicious user could bypass the OTP verification process and successfully register with any mobile number, potentially leading to unauthorized account creation and abuse of the registration system.

Remediation

To mitigate this issue, it is recommended to use alphanumeric characters for OTP generation instead of relying solely on numeric 5-digit OTPs. Additionally, implementing rate limiting on OTP submission attempts can help prevent brute-force attacks. Furthermore, enforcing a maximum number of retries (typically 3 or 5 attempts) for entering the OTP will limit the attack surface and reduce the chances of bypassing the verification process.

Reference(s):

- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- https://www.php.net/manual/en/features.file-upload.php

Evidence Image(s):



Figure 4.4.1: Brute-force OTP to Register with any Mobile Number.

4.5 Information Disclosure via Error Messages

Severity	Low
Status	Not Fixed
CVSS	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
Affected Component	• https://redacted.com/

Description

During the penetration test, it was discovered that the web portal is revealing detailed error messages in its responses. These messages expose sensitive information, such as source code or stack traces, which can be valuable to attackers. Such information should never be exposed to users as it can provide critical insights into potential vulnerabilities in the system.

Impact

The displayed error messages disclose sensitive information, such as source code or stack traces, which can be leveraged by attackers to craft more sophisticated attacks, thereby expanding the attack surface and increasing the likelihood of a successful compromise.

Remediation

Error messages should not be sent to users in production environments. Instead, detailed error information should be logged securely on the backend (e.g., in log files or a database) with a reference number. The user should only see a generic, user-friendly error message along with a reference number, ensuring sensitive information is kept private while still allowing for troubleshooting and support.

Reference(s):

- https://owasp.org/www-community/Improper_Error_Handling
- https://www.c-sharpcorner.com/blogs/handling-verbose-error-messageimproper-errorhandling-in-webconfig

Evidence Image(s):

Screenshot can't be added to Sample report for Confidential purposes.

4.6 Enabled Directory Listing in Uploads Directory

Severity	Informational
Status	Not Fixed
CVSS	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:N
Affected Component	• https://redacted.com/

Description

Cross-Origin Resource Sharing (CORS) is a mechanism that allows resources, such as fonts or images, to be requested from domains other than the one from which the resource originated. The Access-Control-Allow-Origin header controls whether a resource can be shared across domains. If the header is set to *, it allows resources to be shared with any origin. This means any external website can make XMLHttpRequest (XHR) requests to the site and access sensitive data, potentially leading to unauthorized actions.

Impact

An attacker could exploit this misconfiguration by making unauthorized requests from a different domain, potentially exposing sensitive data or allowing malicious actions, such as executing commands on the server and gaining control over it.

Remediation

To mitigate this risk, configure the Access-Control-Allow-Origin header to only permit specific, trusted domains to access resources. Avoid using the wildcard *, as it allows all origins to access your resources.

Reference(s):

- https://portswigger.net/web-security/cors/access-control-allow-origin
- https://www.moesif.com/blog/technical/cors/Authoritative-Guide-to-CORS-Cross-OriginResource-Sharing-for-REST-APIs/

Evidence Image(s):



Figure 4.6.1: Allow-Control-Allow-Origin set to wildcard (*).

5. Fix Validation Status

The security issues identified during the penetration engagement have been remediated, and further details for each security issue can be found below.

5.1 Unrestricted File Upload leading to Command Execution

The application previously allowed unrestricted file uploads, posing a risk of remote code execution. To mitigate this, the development team implemented strict file type validation, size restrictions, and secure upload directories. Upon testing, we confirmed that the implemented controls effectively prevent unauthorized file uploads and execution, resolving the vulnerability.

5.2 User Account Takeover Using Reset Password Functionality

During the penetration test, it was discovered that the password reset token was exposed in the server's response, allowing an attacker to intercept and misuse it to gain unauthorized access. To mitigate this, the development team modified the implementation to ensure that the reset token is now sent only via a secure email channel and never exposed in server responses. Upon testing, we confirmed that the issue is resolved, and the token is no longer accessible through the application's responses.



Reset Token not being sent to the user enabling the issue to be fixed

6. Conclusion

A Black Box Penetration Test was conducted on the redacted web application. During the assessment, we identified a total of 2 critical, 1 high, 1 medium, and 1 low severity vulnerabilities. Notably, a critical unrestricted file upload vulnerability was discovered, which allows attackers to upload malicious PHP files to the webserver and execute arbitrary commands, potentially taking full control of the server. Additionally, insecure querying practices in multiple parameters of various endpoints were found, which could allow attackers to dump the entire database and gain unauthorized access to sensitive information.

Further, we found that an attacker could exploit the password reset functionality to take over any user's account by resetting the password to their own choice. The OTP verification process was also bypassable, allowing a malicious user to brute-force the OTP and register with any mobile number. Additionally, error messages were found to be exposed in the response, which could assist attackers in identifying vulnerabilities and mounting further attacks.

To improve the security posture of the application, it is strongly recommended that the vulnerabilities be addressed in order of the severity levels outlined above. Fixing these issues will significantly reduce the risk of exploitation and enhance the overall security of the application.

7. Attestation

This assessment was conducted by a CREST-certified security professional from Penva Security following established penetration testing standards. All tests were performed within the agreed-upon scope, and every effort was made to avoid service disruptions.

Prepared by:

Tester Name

Penetration Tester, Penva Security

Member of CREST and OSCP Certified



Date: January 01, 2025