**1 Piece Rubiks Cube**

Web Vulnerability Assessment and Penetration Testing

**Prepared for:** Wei Heng Ptd Ltd

**Version:** 0.1

# Version Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Description** | **Author** | **Reviewer** |
| 0.1 | 7 June 2022 | Initial Draft Created | XXX,  *Security Consultants* | XXX,  *Security Consultant* |

# Disclaimer

A security test is considered a snapshot in time. The findings and recommendations reflect the information gathered during the assessment and not any changes or modifications made outside of that period.

A time-bound assessment approach does not allow for a full evaluation of all security controls. Ensign InfoSecurity (SmartTech) Pte. Ltd. prioritized the assessment to identify the weakest security controls an attacker could exploit. Ensign InfoSecurity (SmartTech) Pte. Ltd. recommends conducting similar assessments on an annual basis by internal or third-party assessors to ensure the continued success of the controls.

Recommendations and references enclosed are not meant to be comprehensive or exhaustive in nature. The client should evaluate the suitability of the recommendations against their own policies, standards, processes and risk management posture. The technical implementation of recommendations should be tested before deployment in the affected environment (especially production environment). Ensign InfoSecurity (SmartTech) Pte. Ltd. shall not be liable for the implementation of the recommendations adopted.

References provided neither indicate endorsement nor support for the approaches to meet the recommendations but are provided to allow the client to understand viable solutions to the recommendations.

Table of Contents

[Version Control 2](#_Toc158211347)

[Disclaimer 3](#_Toc158211348)

[1. Executive Summary 5](#_Toc158211349)

[1.1 Scope 5](#_Toc158211350)

[1.2 Limitations 5](#_Toc158211351)

[1.3 List of Findings 6](#_Toc158211352)

[1.4 Finding Status Description 6](#_Toc158211353)

[2. Security Test Findings 7](#_Toc158211354)

[2.1 WVAPT-1: Use of Hardcoded Credentials 7](#_Toc158211356)

[2.2 WVAPT-2: SQL Injection 10](#_Toc158211357)

[APPENDIX 1 - Methodology (WVAPT) 12](#_Toc158211358)

[APPENDIX 2 - Severity Classification & Rating 14](#_Toc158211359)

# Executive Summary

The client has engaged Ensign to conduct security testing with the following information:

|  |  |
| --- | --- |
| **Testing Details** | **Description** |
| Type | Web Vulnerability Assessment and Penetration Testing (“WVAPT”) |
| System Name | 1 Rubiks Cube |
| Testing Environment | Ensign Penetration Testing Laptop, with static IP address range of XXXXX to XXXXX |
| Testing Period | 6 June 2022 to 7 June 2022 |
| System Environment | UAT (Internet) |
| Approach | Non-Authenticated |
| No. of User Role(s) | 0 |

Table 1: Details of the security testing.

The purpose of security testing is to identify and mitigate potential security vulnerabilities and risks specific to an organization or system. By conducting security testing based on the client’s risk profile within a specified time frame, the organization can prioritize and implement mitigation measures promptly, reducing the window of opportunity for potential attackers. The results of the security testing shall be used to guide the remediation efforts and support the strengthening of the client’s overall security posture.

The classification of the findings identified are summarized in the table below, according to its severity ratings.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Severity Rating** | | | | | |
| Critical | High | Medium | Low | None | **Total** |
| 0 | 2 | 0 | 0 | 0 | **2** |

Table 2: Summary of Findings.

## Scope

The scope for the security test is as follows:

|  |  |  |
| --- | --- | --- |
| **SN.** | **URL** | **Remarks** |
|  | https://43a7-122-11-192-165.ap.ngrok.io/ctf/login.php | UAT |

Table 3: List of URLs in scope of the security testing.

The list of test accounts provided for this security test is as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN.** | **Role** | **User ID** | **Password** | **Remarks** |
|  | NIL | NIL | NIL | NIL |

Table 4: List of test accounts.

## Limitations

The list of limitations for this security test is as follows:

|  |  |  |
| --- | --- | --- |
| **SN.** | **Functionality** | **Remarks** |
|  | NIL | NIL |

Table 5: List of functions omitted in the security testing.

## List of Findings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN.** | **Findings** | **Severity Rating** | **CVSS Score** | **Finding Status** |
|  | WVAPT- Use of Hardcoded Credentials | High | 7.5 | Open |
|  | WVAPT- SQL Injection | High | 8.1 | Open |

Table 6: List of findings.

## Finding Status Description

|  |  |
| --- | --- |
| **Finding Status** | **Description** |
| Open | The issue or vulnerability has been identified but has not yet been addressed or remediated. It poses a risk to the organization's security and requires further action to be taken. |
| Resolved | The risk finding has been remediated in orderly fashion; or  Mitigation measures have been implemented and minimal residual risk is observed. |
| Partially Resolved | Remediation measures have been implemented for the issue or vulnerability, but further action is required to fully address the issue. |
| Risk Accepted | Management has acknowledged and accepted the inherent risk or residual risk for this finding, determining that the cost of mitigation outweighs the potential impact of the risk. |
| Closed | Management has evaluated and determined the risk finding to be irrelevant, not applicable, or a false positive and poses no threat to the organization. No further action is required. |
| NA | Not applicable. The risk finding has a severity rating of None. |

Table 7: List of finding status and its meaning.

# Security Test Findings



## WVAPT-1: Use of Hardcoded Credentials

|  |  |  |
| --- | --- | --- |
| **Severity Level** | **High** | |
| **CVSS Vector and Score** | AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N | 7.5 |

### Observations

It was observed that the website contained hardcoded and encoded credentials in HTML page’s hidden input type field. It was possible to decode the credentials to login into the web application.

### Proof of Concept (Hardcoded values before login)

Navigate to the following URL and view the page source. Observe the hardcoded and encoded value in the hidden input tag.

|  |  |
| --- | --- |
| URL | <https://43a7-122-11-192-165.ap.ngrok.io/ctf/login.php> |
| Hardcoded Value Found | 616d396c4f6d4e68626e6c7664575978626d52744d773d3d |

Text

Description automatically generated with low confidence

*Figure 1: Hardcoded Credentials*

Copy the value from above and convert it from hex and base64 respectively. Observe the username and password found, “joe” and “canyouf1ndm3”

A screenshot of a computer

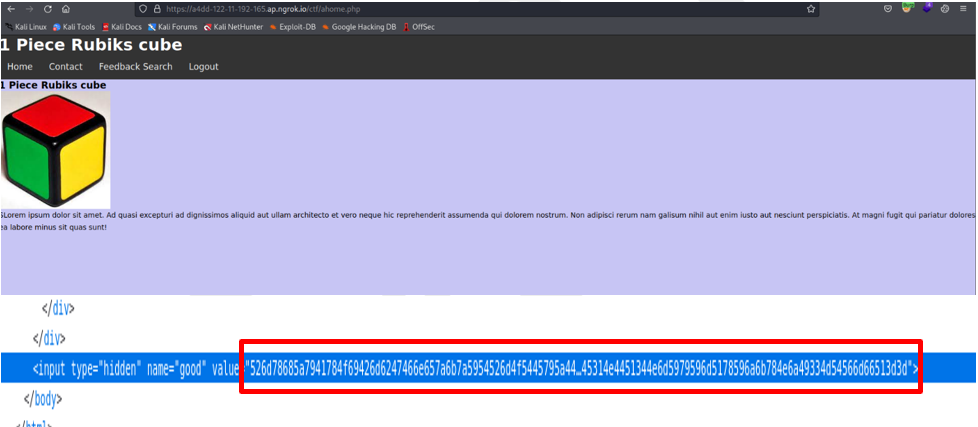
Description automatically generated

*Figure 2: Credentials found*

### Proof of Concept (Hardcoded values after login)

Login using the credentials “joe” and “canyouf1ndm3” and view the page source, Observe that there is another set of a hardcoded and encoded string.

|  |  |
| --- | --- |
| Affected Endpoint | [https://43a7-122-11-192-165.ap.ngrok.io/ctf/ahome.php](https://43a7-122-11-192-165.ap.ngrok.io/ctf/ahome.php%20login.php) |
| Hardcoded Value Found | 526d78685a7941784f69426d6247466e657a6b7a5954526d4f5445795a4459314f4445314e4451344e6d5979596d5178596a6b784e6a49334d54566d66513d3d |



*Figure 3: Another set of hardcoded string once logged in*

Copy the value from above and convert it from hex and base64 respectively. The output shows the flag has been obtained.

|  |  |
| --- | --- |
| Flag 1 | flag{93a4f912d658154486f2bd1b9162715f} |

A screenshot of a computer

Description automatically generated

*Figure 4: Flag Obtained*

### Affected Hosts/Services

|  |  |  |
| --- | --- | --- |
| **SN.** | **Affected Host/Service** | **Status** |
|  | <https://43a7-122-11-192-165.ap.ngrok.io/ctf/login.php> | Open |
|  | [https://43a7-122-11-192-165.ap.ngrok.io/ctf/ahome.php](https://43a7-122-11-192-165.ap.ngrok.io/ctf/ahome.php%20login.php) | Open |

### Implications

An attacker can perform unauthorized login to the web application. As a result, an attacker can gain access over the entire website, which can lead to an attacker access and steal sensitive user data.

### Recommendations

Remove the hardcoded values from the page. Instead use configuration files to store sensitive information.

### References

* [https://cwe.mitre.org/data/definitions/547.html](https://cwe.mitre.org/data/definitions/547.html%20)
* <https://wiki.sei.cmu.edu/confluence/display/seccode/Top+10+Secure+Coding+Practices>

### Remarks

|  |  |
| --- | --- |
| **Management Comments *(To be filled in by the client)*** | |
| **Date** | **Comment** |
| Please select the date. | Enter your comments and screenshots here. |

|  |  |
| --- | --- |
| **Post Review Observations *(To be filled in by Ensign)*** | |
| **Date** | **Comment** |
| Please select the date. | Enter your comments and screenshots here. |

## -2: SQL Injection

|  |  |  |
| --- | --- | --- |
| **Severity Level** | **High** | |
| **CVSS Vector and Score** | [AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N](https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?vector=AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N&version=3.1) | 8.1 |

### Observations

It was found that the search bar in the web application is vulnerable to SQL injection.

### Proof of Concept

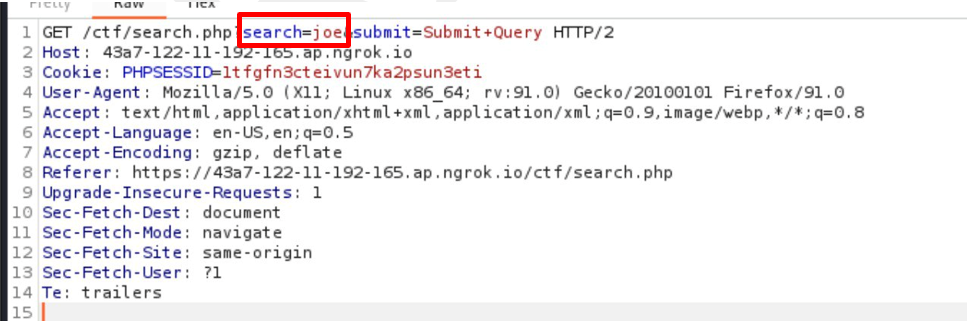
Once logged in using the credentials found in “WVAPT-1 Use of Hardcoded Credentials”, navigate to the search tab and enter arbitrary values in the search function. Intercept that request and save that request to a file.

|  |  |
| --- | --- |
| Affected Endpoint | https://43a7-122-11-192-165.ap.ngrok.io/ctf/search.php |

A screenshot of a computer

Description automatically generated

*Figure 5: Search function*



*Figure 6: Intercepted GET request.*

Run the following command to retrieve the flag.

|  |  |
| --- | --- |
| Command | sqlmap -r <filename> -D dcsc -T secret --dump --threads=8 |
| Flag | flag{08c9d38d21b3039591823dd1f04768cd} |

Graphical user interface

Description automatically generated with medium confidence

*Figure 7: Flag Obtained*

### Affected Hosts/Services

|  |  |  |  |
| --- | --- | --- | --- |
| **SN.** | **Affected Host/Service** | **Vulnerable Parameter** | **Status** |
|  | https://43a7-122-11-192-165.ap.ngrok.io/ctf/search.php | Search | Open |

### Implications

An SQL injection attack can result in unauthorized access to sensitive data. This might include data belonging to other users, or any other data that the application itself is able to access.

### Recommendations

Use parameterized queries for untrusted input. In addition validate input sent to the server.

### References

* <https://portswigger.net/web-security/sql-injection>
* <https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html>
* <https://cheatsheetseries.owasp.org/cheatsheets/Query_Parameterization_Cheat_Sheet.html>

### Remarks

|  |  |
| --- | --- |
| **Management Comments *(To be filled in by the client)*** | |
| **Date** | **Comment** |
| Please select the date. | Enter your comments and screenshots here. |

|  |  |
| --- | --- |
| **Post Review Observations *(To be filled in by Ensign)*** | |
| **Date** | **Comment** |
| Please select the date. | Enter your comments and screenshots here. |

1. Methodology (WVAPT)

**Planning and Preparation**

**Information**

**Gathering, Mapping and Profiling**

**Execution and Testing**

**Analysis and Reporting**

**Follow-up (Optional)**

Ensign’s Web Vulnerability Assessment and Penetration Test is based on OWASP Top 10 (2021) standards and is backed up by the company’s years of information security experience. It is also designed to be adaptable towards the application environment and security requirements.

Our methodology can be summarised as a five-step approach consisting of following phases:

* **Planning and Preparation**
  + - Conduct project kick-off meeting.
    - Confirm project scope and objectives.
    - Identify parties involved and key contact persons.
    - Establish project timelines and duration.
* **Information Gathering, Mapping, and Profiling**
  + - Determine the Web Application(s) be reviewed.
    - Gather information on the Web Application and its operating environment.
    - Identify the criteria to review the application against, such as Application development guidelines, Security policies/ requirements, regulatory mandates, among others.
    - Design test procedures that will determine compliance with identified review criteria.
* **Execution and Testing**
  + - Using a hybrid approach that involves automated web application vulnerability scan(s), manual validation of the automated scan(s) results and manual penetration test to uncover vulnerabilities in the following categories:

1. Broken Access Control

Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users’ data, change access rights, etc.

1. Cryptographic Failures

Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

1. Injection

Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker’s hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

1. Insecure Design

Focuses on risks related to design and architectural flaws, with a call for more use of threat modelling, secure design patterns, and reference architectures. An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks. One of the factors that contribute to insecure design is the lack of business risk profiling inherent in the software or system being developed, and thus the failure to determine what level of security design is required.

1. Security Misconfiguration

Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched and upgraded in a timely fashion.

1. Vulnerable and Outdated Components

Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defences and enable various attacks and impacts.

1. Identification and Authentication Failures

Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users’ identities temporarily or permanently.

1. Software and Data Integrity Failures

Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs). An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code, or system compromise. Lastly, many applications now include auto-update functionality, where updates are downloaded without sufficient integrity verification and applied to the previously trusted application. Attackers could potentially upload their own updates to be distributed and run on all installations. Another example is where objects or data are encoded or serialized into a structure that an attacker can see and modify is vulnerable to insecure deserialization.

1. Security Logging and Monitoring Failures

Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

1. Server-Side Request Forgery (SSRF)

SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL).

* **Analysis and Reporting**
  + - Summarise and analyse all security weaknesses identified in a report that includes:
      * Summary of applications reviewed, and the corresponding security weaknesses identified.
      * A detailed account of all security weaknesses found.
      * Suggestions and techniques to resolve vulnerabilities found.
* **Follow-up (Optional)**
  + - Obtain remediation plan/documentation.
    - Determine post review schedule and timelines.
    - Conduct security review to validate the effectiveness of remediation efforts.
    - Summarise and report findings.

1. Severity Classification & Rating

Analysis on the criticality severity of the findings was performed based on the Common Vulnerability Scoring System 3.1 (CVSS v3.1), and the confidentiality, integrity and availability impact were also analysed wherever applicable.

Refer to the following CVSS resource website below for the explanation of each severity component metric and how findings are rated:

<https://www.first.org/cvss/v3.1/specification-document>

The final rating scale is categorised into 5 severity-level according to CVSS scoring as follows:

|  |  |  |
| --- | --- | --- |
| Rating | CVSS Score | Description |
| None | 0.0 | No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation. |
| Low | 0.1-3.9 | Vulnerabilities are non-exploitable but would increase an organization’s attack surface. It is advised to form a plan of action and patch during the next maintenance window. |
| Medium | 4.0-6.9 | Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved. |
| High | 7.0-8.9 | Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible. |
| Critical | 9.0-10.0 | Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately. |

Table A- 1: The qualitative severity rating scale according to CVSS 3.1 The calculation is performed via an online CVSSv3.1 Calculator (https://www.first.org/cvss/calculator/3.1).