WardenSec

CYBERSECURITY

Findings Report

Demo report - Web

COMMISSIONED BY TEST

EXECUTION DATES : 01-06-2024 - 12-06-2024

REPORT STATUS: Final

VERSION : 7

DATE : 19-06-2024

EXECUTED BY: Pavel Skorepa, Jan Recinsky

Table to Contents

1. MANAGEMENT SUMMARY

1.1. GENERAL INFORMATION

A penetration test is a controlled attack on a computer or network system with the intent of finding security weaknesses and potentially gaining access to the system and its data. The process involves identifying target systems and setting a goal, then obtaining contextual and technical information, followed by vulnerability identification and validation phases. Potential solutions are then shared to assist in helping the organization mitigate any found vulnerabilities.

Ultimately, the organization conducting a pentest can reduce risk in a focused and cost-effective manner. Penetration tests also help in determining where general weaknesses exist in the digital environment, which leads to a more proactive approach in security management processes. For example, when control mechanisms are built into the policies and procedures driving daily operations.

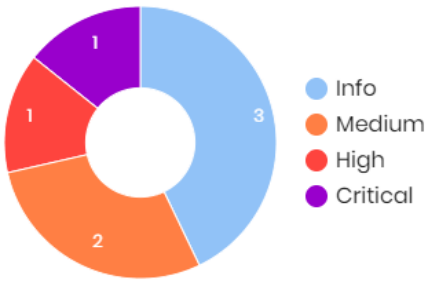
The main objective of the penetrating test is to provide input from a technical and real-world perspective. This input is designed to be valuable in helping the organization to determine and reduce business risk. We obtain that information by identifying system and infrastructure vulnerabilities, which could be used by an attacker to gain unauthorized access to systems or information, while providing sufficient information for the organization to understand associated business risks and potential impact of those vulnerabilities - so that the organization can assess actions and prioritize how to mitigate risks.

This report covers the results of the Demo report - Web security investigation conducted by CYVER at the request of TEST.

This security investigation was conducted from 01-06-2024 - 12-06-2024.
The research methodology used for a Web Security Assessment is in accordance with the OWASP Testing Guide.

1.2. RESULTS

The pentest identified a number of vulnerabilities. Found vulnerabilities are detailed below, organized by risk classification.



1. Name: Code Execution via File Upload
Severity: Critical
Type of Risk: Code Execution
Description: A critical vulnerability was discovered where code can be executed via file upload. When a file is uploaded and requested, the code is executed in the context of the web server. This poses a significant threat to the security of the system and needs immediate attention.

2. Name: Possible Cross-site Scripting
Severity: High
Type of Risk: Cross-site Scripting
Description: A high-risk vulnerability was detected where possible Cross-site Scripting could occur. This allows an attacker to execute a dynamic script in the context of the application, potentially hijacking user sessions or changing the look of the page to steal user credentials. Although not confirmed, it is strongly recommended to investigate this issue manually.

3. Name: Directory Listing (Apache)
Severity: Medium

Type of Risk: Directory Listing
Description: A medium-risk vulnerability was identified where the web server responded with a list of files located in the target directory. This could potentially expose sensitive information to attackers.

4. Name: Apache Server-Status Detected
Severity: Medium
Type of Risk: Information Disclosure
Description: A medium-risk vulnerability was detected where the Apache server-status is enabled. Information disclosed from this page could be used by an attacker to gain additional information about the target system.

5. Name: Missing Content-Type Header
Severity: Info
Type of Risk: MIME-Sniffing Attack
Description: An informational risk was detected where a missing `Content-Type` header was found. This could potentially put the website at risk of MIME-sniffing attacks.

The above vulnerabilities are sorted by severity level, with the most critical at the top. The potential impact of these vulnerabilities on TEST's system security is significant and actions should be prioritized accordingly.

1.3. RECOMMENDATIONS

Based on the findings, WardenSechas the following recommendations:

1. Code Execution via File Upload (Critical Severity): Implement a robust file validation and sanitization process. Ensure that only safe and necessary file types are allowed for upload. Additionally, consider implementing a mechanism to isolate and execute uploaded files in a secure and controlled environment. This will prevent potential malicious code from being executed on your server, thereby enhancing your security posture.
Impact: If not addressed, this vulnerability could allow an attacker to execute arbitrary code on your server, potentially leading to unauthorized access, data breaches, and disruption of services.

2. Cross-site Scripting (High Severity): Although the pentester could not confirm this vulnerability, it is recommended to manually investigate the issue. If confirmed, implement proper input validation and sanitization to prevent the execution of malicious scripts. Also, consider implementing Content Security Policy (CSP) to prevent Cross-site Scripting attacks.
Impact: If left unaddressed, this vulnerability could allow an attacker to hijack user sessions, steal sensitive information, and potentially gain unauthorized access to the application.

3. Directory Listing (Apache) (Medium Severity): Disable directory listing on your Apache server. This can be done by modifying the server configuration files or through an .htaccess file.
Impact: If not addressed, this vulnerability could allow an attacker to view all the files in a directory, potentially exposing sensitive information.

4. Apache Server-Status Detected (Medium Severity): Disable the 'server-status' page on your Apache server to prevent potential information disclosure. This can be done by modifying the server configuration files.
Impact: If not addressed, this vulnerability could provide an attacker with additional information about your system, potentially aiding in further attacks.

5. Missing Content-Type Header (Info Severity): Ensure that all responses from your server include a 'Content-Type' header. This will prevent potential MIME-sniffing attacks.
Impact: If not addressed, this vulnerability could allow an attacker to perform MIME-sniffing attacks, potentially leading to Cross-site Scripting (XSS) or other types of attacks.

1.4. CONCLUSION

The pentest conducted for TEST as part of the Demo report - Web project took place from 01-06-2024 to 12-06-2024. The objective was to identify potential vulnerabilities in the company's security systems. Several vulnerabilities were discovered during this test, posing varying degrees of risk to the company's security.

The identified vulnerabilities include critical issues such as code execution via file upload, high severity issues like potential cross-site scripting, and medium severity issues like directory listing (Apache) and Apache server-status detection. Additionally, an informational issue related to a missing Content-Type header was also found.

It's important to note that the cross-site scripting vulnerability could not be confirmed and requires further manual investigation. Each vulnerability has been classified based on its severity, impact, and likelihood.

In conclusion, the pentest has revealed several areas of concern in TEST's security systems. Detailed reports containing specific information about each vulnerability and recommendations for mitigation are available for further review. The company is advised to address these vulnerabilities promptly to enhance its security posture.

2. Assignment Details

This pentest is based on the agreed-upon proposal with reference reference which includes the following description of the assignment:

2.2. Scope

The scope of the investigation is shown in the table below. Systems and applications not listed have not been pentested..

Asset	Type
Dummy-Device [10.10.10.11]	Web Application

2.3. Purpose of this Assignment

Independently determining the security level of Demo report - Web, detecting vulnerabilities, and suggesting possible improvements.

To determine that Demo report - Web satisfies

2.4. Research Method

This study was conducted according to a grey-box application study.

When investigating according to the grey-box-method pre-known credentials and limited information about the test objects are available.

The OWASP WSTG 4.2 was used for the research.

2.5. Reporting

The management summaries and technical summaries respectively serve as an overview of the research results for general and technical management. In subsequent chapters, the detailed results are described and supported by reproducible findings. These chapters are intended to guide technical personnel in reproducing and mitigating vulnerability findings.

2.6. Limitations

A Pentest provides valuable insight into the IT security of the target system or application. However, such an investigation is only a snapshot and does not guarantee the security of the IT environment and data. New attack techniques are constantly being developed and discovered. In addition, a small adjustment to the IT environment can easily introduce new vulnerabilities. The role of processes, procedures, and the human factor in information security are at least as important as technology used. This report only provides an overview of vulnerabilities found and is therefore not intended as a guarantee of security.

In addition, it is important to note that a Pentest is performed by people and that during the research period, choices are made in approach and use of tooling. Pentest results also heavily rely on the capabilities of the executive consultant. It is therefore possible that tests with a repetitive character may deviate in results.

3. TECHNICAL SUMMARY

This chapter provides an overview of key findings included in this report. For a detailed description, see the actual findings later in this report.

3.1. Findings Summary

Vulnerability	Severity
F-2024-0100 - Code Execution via File Upload	Critical
F-2024-0095 - Cross-site Scripting	High
F-2024-0097 - Directory Listing (Apache)	Medium
F-2024-0099 - Apache Server-Status Detected	Medium
F-2024-0096 - [Possible] SQL Injection	Info
F-2024-0098 - I'm a Teapot	Info
F-2024-0101 - Missing Content-Type Header	Info

3.2. OWASP Reference

Control	Findings
A01:2021 | Broken Access Control	
A01:2021 - Broken Access Control	0 Critical 0 High 0 Medium 0 Low 0 Info
A02:2021 | Cryptographic Failures	
A02:2021 - Cryptographic Failures	0 Critical 0 High 0 Medium 0 Low 0 Info
A03:2021 | Injection	
A03:2021 - Injection	0 Critical 1 High 0 Medium 0 Low 1 Info
A04:2021 | Insecure Design	
A04:2021 - Insecure Design	0 Critical 0 High 0 Medium 0 Low 0 Info
A05:2021 | Security Misconfiguration	
A05:2021 - Security Misconfiguration	0 Critical 0 High 2 Medium 0 Low 1 Info
A06:2021 | Vulnerable and Outdated Components	
A06:2021 - Vulnerable and Outdated Components	1 Critical 0 High 0 Medium 0 Low 0 Info
A07:2021 | Identification and Authentication Failures	
A07:2021 - Identification and Authentication Failures	0 Critical 0 High 0 Medium 0 Low 0 Info
A08:2021 | Software and Data Integrity Failures	
A08:2021 - Software and Data Integrity Failures	0 Critical 0 High 0 Medium 0 Low 0 Info
A09:2021 | Security Logging and Monitoring Failures	
A09:2021 - Security Logging and Monitoring Failures	0 Critical 0 High 0 Medium 0 Low 0 Info
A10:2021 | Server Side Request Forgery (SSRF)	
A10:2021 - Server Side Request Forgery (SSRF)	0 Critical 0 High 0 Medium 0 Low 0 Info

3.3. Main Findings

- Code Execution via File Upload
- Cross-site Scripting

3.4. Observations

4. Methodology

Control objectives covered:
OWASP Top 10 2021

Network port scanning and protocol identification: We inventoried all open ports in order to detect every service provided by the servers in order to plan the vulnerability assessment.

Software fingerprinting and identification: Once services and applications were identified, we proceeded to identify known software and their versions in order to look for components with known vulnerabilities from public databases.

Manual reconnaissance: In order to identify information that could be subject to be used for the testing.

Manual penetration testing: Intrusion attempts based on OSSTTM and OWASP methodology.

IP ranges listed below were in scope for this assessment. Ranges were scanned using various automated tools to identify known vulnerabilities and service misconfigurations, such as those tracked by CVE and listed in the CIS Critical Security Controls. Targets were then assessed manually in order to validate scan results, determine their risk, and to improve coverage and quality of the test.

The test was done according to penetration testing best practices. The flow from start to finish is listed below.

Pre Engagement:

- Scoping
- Customer
- Documentation
- Information
- Discovery

Penetration Testing:

- Tool assisted assessment
- Manual assessment
- Exploitation
- Risk analysis
- Reporting

Post Engagement

- Prioritized remediation
- Best practice support
- Re-testing

Risk Factors

Each finding is assigned two factors to measure its risk. Factors are measured on a scale of 1 (very low) through 5 (very high).

Impact

This indicates the finding's effect on technical and business operations. It covers aspects such as the confidentiality, integrity, and availability of data or systems; and financial or reputational loss.

Likelihood

This indicates the finding's potential for exploitation. It takes into account aspects such as skill level required of an attacker and relative ease of exploitation.

Criticality Definitions

Findings are grouped into three criticality levels based on their risk as calculated by their business impact and likelihood of occurrence, `risk = impact * likelihood`. This follows the OWASP Risk Rating Methodology.

High

Vulnerabilities with a high or greater business impact and high or greater likelihood are considered High severity. Risk score minimum 16.

Medium

Vulnerabilities with a medium business impact and likelihood are considered Medium severity. This also includes vulnerabilities that have either very high business impact combined with a low likelihood or have a low business impact combined with a very high likelihood. Risk score between 5 and 15.

Low

Vulnerabilities that have either a very low business impact, maximum high likelihood, or very low likelihood, maximum high business impact, are considered Low severity. Also, vulnerabilities where both business impact and likelihood are low are considered Low severity. Risk score 1 through 4.

5. Findings

5.1. Summary

5.1.1. Vulnerability Summary

Vulnerability	Severity
F-2024-0100 - Code Execution via File Upload	Critical
F-2024-0095 - Cross-site Scripting	High
F-2024-0097 - Directory Listing (Apache)	Medium
F-2024-0099 - Apache Server-Status Detected	Medium
F-2024-0096 - [Possible] SQL Injection	Info
F-2024-0098 - I'm a Teapot	Info
F-2024-0101 - Missing Content-Type Header	Info

5.1.2. Observations Summary

5.2. Vulnerability Details

F-2024-0100 - Code Execution via File Upload

Description: Pentester detected a code execution via file upload. Pentester successfully uploaded a file and when requesting the uploaded file, code is executed in the context of the web server.

Assets:

- Dummy-Device [10.10.10.11]

Classification

Severity: Critical

Recommendations

Background Information: The web server can be compromised by uploading and executing a web-shell which can run commands, browse system files, browse local resources, attack other servers, and exploit the local vulnerabilities, and so forth.

Remediation:

- Never accept a filename and its extension directly without having a white-list filter.
- Uploaded directory should not have any "execute" permission.

Compliance

OWASP Top 10 2021:

- A06:2021 - Vulnerable and Outdated Components

F-2024-0095 - Cross-site Scripting

Description:

Pentester detected Possible Cross-site Scripting, which allows an attacker to execute a dynamic script (*JavaScript, VBScript*) in the context of the application.

This allows several different attack opportunities, mostly hijacking the current session of the user or changing the look of the page by changing the HTML on the fly to steal the user's credentials. This happens because the input entered by a user has been interpreted as HTML/JavaScript/VBScript by the browser. Cross-site scripting targets the users of the application instead of the server. Although this is a limitation, since it allows attackers to hijack other users' sessions, an attacker might attack an administrator to gain full control over the application.

Although pentester believes there is a cross-site scripting in here, it could **not confirm it**. We strongly recommend investigating the issue manually to ensure it is cross-site scripting and needs to be addressed.

Assets:

- Dummy-Device [10.10.10.11]

Classification

Severity: High

CWEs:

- CWE-712 | OWASP Top Ten 2007 Category A1 - Cross Site Scripting (XSS)
- CWE-1033 | OWASP Top Ten 2017 Category A7 - Cross-Site Scripting (XSS)

Recommendations

Background Information:

There are many different attacks that can be leveraged through the use of XSS, including:

- Hijacking user's active session.
- Changing the look of the page within the victim's browser.
- Mounting a successful phishing attack.
- Intercepting data and performing man-in-the-middle attacks.

Remediation:

This issue occurs because the browser interprets the input as active HTML, JavaScript or VBScript. To avoid this, all input and output from the application should be filtered / encoded. Output should be filtered / encoded according to the output format and location.

There are a number of pre-defined, well structured whitelist libraries available for many different environments. Good examples of these include OWASP Reform and Microsoft Anti-Cross-site Scripting libraries.

Additionally, you should implement a strong Content Security Policy (CSP) as a defense-in-depth measure if an XSS vulnerability is mistakenly introduced. Due to the complexity of XSS-Prevention and the lack of secure standard behavior in programming languages and frameworks, XSS vulnerabilities are still common in web applications.

CSP will act as a safeguard that can prevent an attacker from successfully exploiting Cross-site Scripting vulnerabilities in your website and is advised in any kind of application.

Please make sure to scan your application again with Content Security Policy checks enabled after implementing CSP, in order to avoid common mistakes that can impact the effectiveness of your policy. There are a few pitfalls that can render your CSP policy useless and we highly recommend reading the resources linked in the reference section before you start to implement one.

Compliance

OWASP Top 10 2021:

- A03:2021 - Injection

F-2024-0097 - Directory Listing (Apache)

Description: Pentester identified a Directory Listing (Apache).

The web server responded with a list of files located in the target directory.

Assets:

- Dummy-Device [10.10.10.11]

Classification

Severity: Medium

Recommendations

Background Information: An attacker can see the files located in the directory and could potentially access files which disclose sensitive information.

Remediation: Change your server configuration file. A recommended configuration for the requested directory should be in the following format:

Remove the *Indexes* option from configuration. Do not forget to remove *MultiViews* as well.
2. Configure the web server to disallow directory listing requests.
3. Ensure that the latest security patches have been applied to the web server and the current stable version of the software is in use.

```
<Directory /{YOUR DIRECTORY}>
Options FollowSymLinks
</Directory>
```

Compliance

OWASP Top 10 2021:

- A05:2021 - Security Misconfiguration

F-2024-0099 - Apache Server-Status Detected

Description:

Pentester detected that `Apache server-status` is enabled.

Information disclosed from this page can be used to gain additional information about the target system.

Assets:

- Dummy-Device [10.10.10.11]

Classification

Severity: Medium

Recommendations

Background Information:

An attacker can gather reconnaissance information about the internals of the target web server, such as:

- Server uptime
- Individual request-response statistics and CPU usage of the working processes
- Current HTTP requests, client IP addresses, requested paths, and processed virtual hosts

This type of information can help the attacker gain a greater understanding of the system in use and the other potential avenues of attack available.

Remediation:

We recommend disabling this functionality. Comment out the `Location/server-info` section from Apache configuration file `httpd.conf` (for Redhat, Centos, Fedora) or `apache2.conf` (for Debian, Ubuntu).

Compliance

OWASP Top 10 2021:

- A05:2021 - Security Misconfiguration

F-2024-0096 - [Possible] SQL Injection

Description:

Pentester identified a Possible SQL Injection, which occurs when data input by a user is interpreted as a SQL command, rather than as normal data by the backend database.

However, this issue **could not be confirmed** by pentester. Pentester believes this was not an SQL injection; however, there were some indications of a possible SQL injection. There can be numerous reasons for not being able to confirm it.

We strongly recommend investigating the issue manually. You can also consider sending the details of this issue to us so we can address this issue for the next time and give you a more precise result.

Assets:

- Dummy-Device [10.10.10.11]

Classification

Severity: Info

Recommendations

Background Information:

Depending on the backend database, the database connection settings and the operating system, an attacker can mount one or more of the following types of attacks successfully:

- Reading, updating and deleting arbitrary data/tables from the database
- Executing commands on the underlying operating system

There are numerous freely available tools to exploit SQL injection vulnerabilities. This is a complex area with many dependencies; however, it should be noted that the numerous resources available in this area have raised both attacker awareness of the issues and their ability to discover and leverage them. SQL injection is one of the most common web application vulnerabilities.

Remediation:

A robust method for mitigating the threat of SQL injection-based vulnerabilities is to use parameterized queries (*prepared statements*). Almost all modern languages provide built-in libraries for this. Wherever possible, do not create dynamic SQL queries or SQL queries with string concatenation.

Compliance

OWASP Top 10 2021:

- A03:2021 - Injection

F-2024-0098 - I'm a Teapot

Description: identified a... uhm... teapot(!?). Go and get a cup of tea while is scanning the site.

Assets:

- Dummy-Device [10.10.10.11]

Classification

Severity: Info

F-2024-0101 - Missing Content-Type Header

Description: Pentester detected a missing `Content-Type` header which means that this website could be at risk of a MIME-sniffing attacks.

Assets:

- Dummy-Device [10.10.10.11]

Classification

Severity: Info

Recommendations

Background Information: MIME type sniffing is a standard functionality in browsers to find an appropriate way to render data where the HTTP headers sent by the server are either inconclusive or missing.

This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the intended content type.

The problem arises once a website allows users to upload content which is then published on the web server. If an attacker can carry out XSS (Cross-site Scripting) attack by manipulating the content in a way to be accepted by the web application and rendered as HTML by the browser, it is possible to inject code in e.g. an image file and make the victim execute it by viewing the image.

Remediation: When serving resources, make sure you send the content-type header to appropriately match the type of the resource being served. For example, if you are serving an HTML page, you should send the HTTP header:

Add the X-Content-Type-Options header with a value of "nosniff" to inform the browser to trust what the site has sent is the appropriate content-type, and to not attempt "sniffing" the real content-type.

```
X-Content-Type-Options: nosniff
```

```
Content-Type: text/html
```

Compliance

OWASP Top 10 2021:

- A05:2021 - Security Misconfiguration

5.3. Observations Details