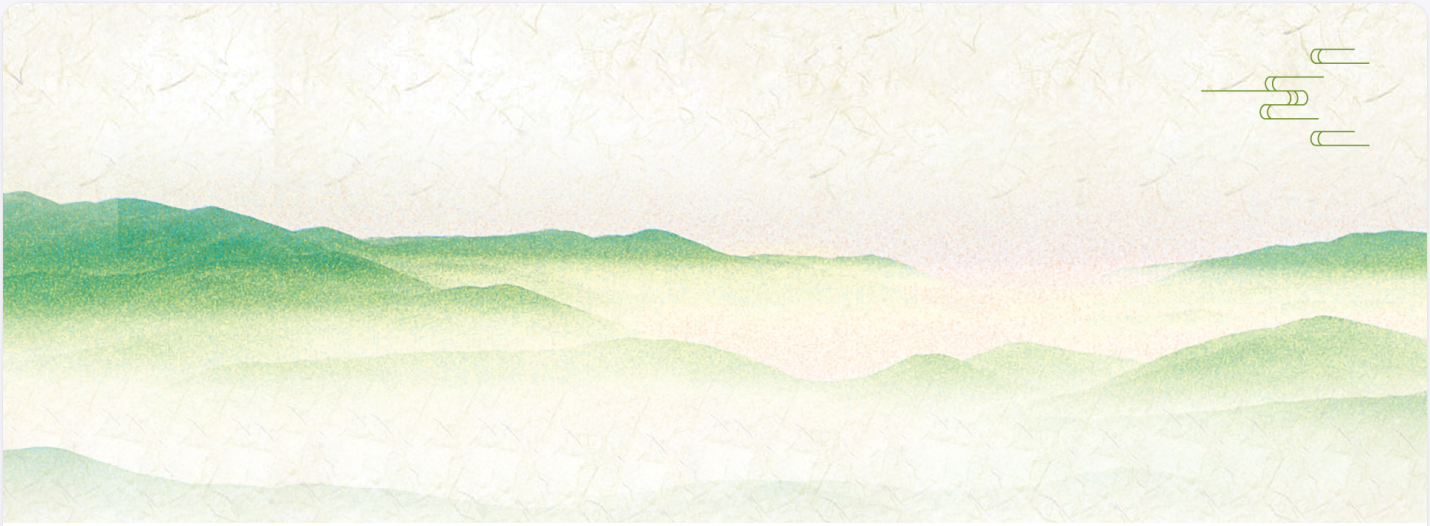
Byebyedoggy



POC

1231 PHPEMS CSRF POC

Dec 31, 2025 阅读时长: 3 分钟

Vulnerability Information

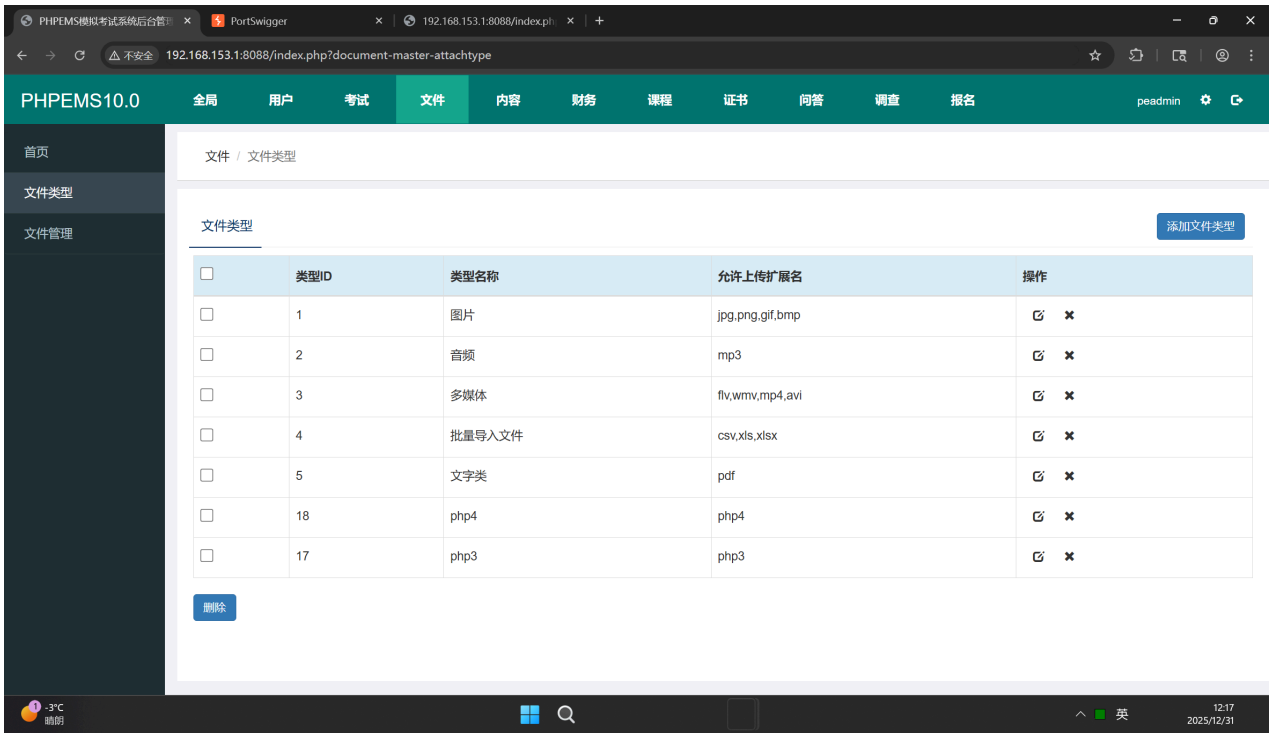
Item	Details
Vulnerability Name	PHPEMS Points Concurrent Usage Race Condition Vulnerability
Affected Versions	PHPEMS 11.0 and earlier
Type	Cross-Site Request Forgery (CSRF)
Severity	Medium

Reproduction Environment

1. Test Site: Local deployment
2. Source Code Setup: Download any version from the official website: https://www.phpems.net/
3. Tool: Burp Suite

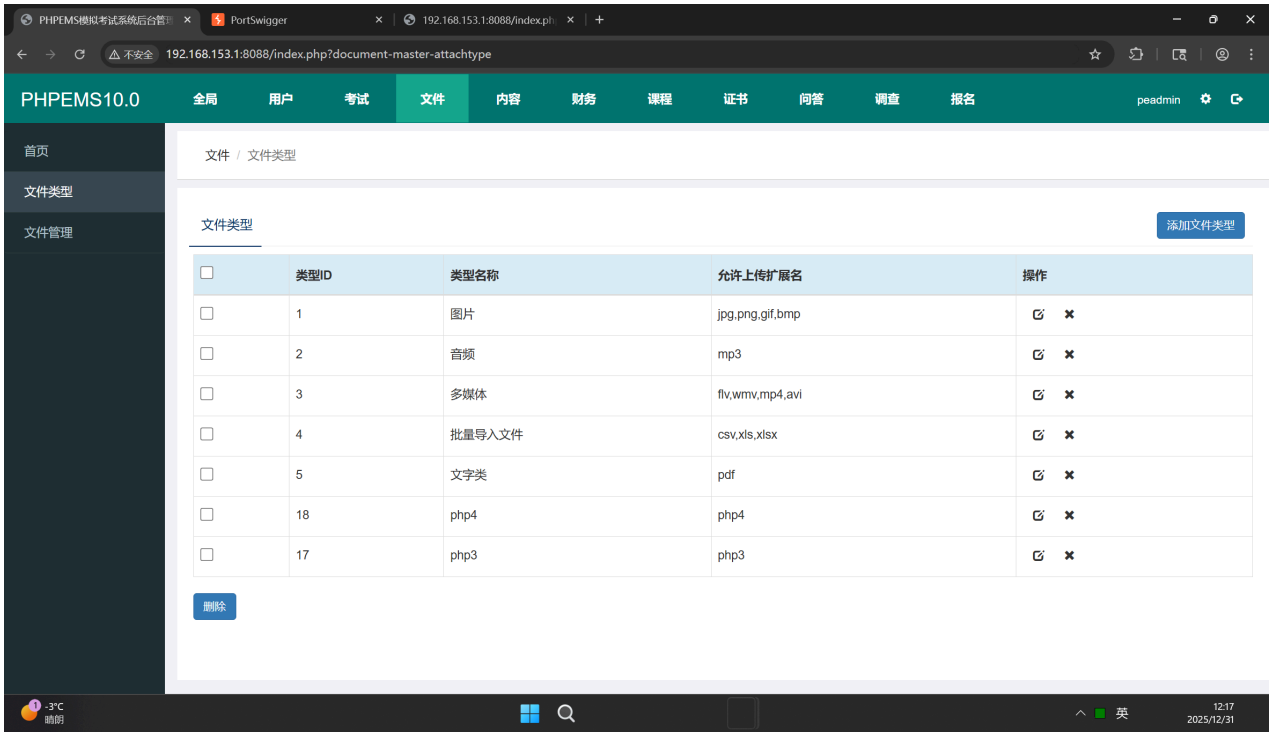
Reproduction Steps

1. Prepare the test account: Log in to the PHPEMS system with a valid ordinary user account, and confirm that the account has the permission to manage personal/file attachment settings (or system-level file type configuration permission that is not protected by CSRF).
2. Capture the vulnerable request: Use Burp Suite to intercept the HTTP request when the user modifies the allowed file name extensions (e.g., adding/editing the list of uploadable file suffixes). The core request typically contains parameters such as allowed_extensions (or similar keys) for submitting the updated suffix list.



3. Construct the CSRF payload: Create a malicious static HTML page that automatically sends the intercepted request (modified to add dangerous suffixes like phar). The payload will

forge the request using the victim's valid session (without the victim's knowledge).



4. Trigger the CSRF attack: Ask the logged-in ordinary user (victim) to access the malicious HTML page (e.g., via phishing link, embedded in a malicious website). The page will automatically submit the forged request in the background.

Impact

- Ordinary users (without proper authorization awareness) can be tricked into modifying the system's allowed file name extensions, adding dangerous suffixes such as phar that can execute malicious code.
- Attackers can further upload malicious PHAR files (or PHP scripts) after the suffix restriction is lifted, leading to server-side code execution, data theft, or server takeover.
- Compromises the system's file upload security mechanism, which is a critical defense line against malicious file attacks. May cause the leakage of sensitive data (e.g., user information, exam data) in the PHPEMS system, or damage to the system's normal operation.

Mitigation Recommendations

1. Implement CSRF token verification: Add a unique, random CSRF token to all sensitive operation requests (e.g., file extension modification, points operation). The token should be stored in the user's session and verified on the server side before processing the request.
2. Example implementation: Generate a token in the page and carry it in the form, then validate it on the backend. Strengthen permission control: Restrict the permission to modify file name extensions to only high-privilege backend administrators (prohibit ordinary users from accessing this function), and add secondary authentication (e.g., password verification) for sensitive configuration modifications.
3. Validate and whitelist file extensions: Even if the configuration is modified, implement a hard-coded whitelist of safe file extensions on the server side for file upload operations. Reject any file whose suffix is not in the whitelist (ignore the user-modifiable configuration for critical security checks).
4. Check the Referer/Origin header: On the server side, verify that the Referer or Origin header of the request comes from the official domain of the PHPEMS system to block cross-domain forged requests (as an auxiliary defense measure).
5. Disable the execution permission of uploaded files: Store uploaded files in a directory that is not parsed by the web server (e.g., disable PHP parsing for the upload directory), and prohibit the execution of PHAR/PHP files in the upload directory even if they are uploaded.

POC

相关文章

使用 Hugo 构建

主题 **Stack** 由 Jimmy 设计