**Exploit Title: CMS Made Simple Version: 2.2.21 - Arbitrary File Upload**

**Date: 2025-07-04**

**Exploit Author: feixuezhi**

**Vendor Homepage: https://www.cmsmadesimple.org/**

**Version: 2.2.21**

**Tested on: https://softaculous.com/demos/cms\_made\_simple**

**Description:** CMS Made Simple Version 2.2.21 upload-photo.php file contains a critical arbitrary file upload vulnerability. There is a lack of proper validation for the extracted files from compressed archives. Attackers can bypass restrictions to upload compressed files containing malicious PHP scripts and extract them after upload. The uploaded files are stored in a publicly accessible directory, and these scripts can then be accessed and executed via a web browser, enabling the attacker to fully compromise the server environment.

**Steps to Reproduce:**

Log in as admin and navigate to Content > File Manager .

Compress the backdoor PHP script containing <?php echo shell\_exec($\_GET['cmd']); ?> into .tar.gz format.

Click on the Browse Button and upload the .tar.gz file.

Select the uploaded tar.gz file.

Click on "Unpack".

Click on the decompressed PHP file, and add ?cmd=ifconfig after the URL path.

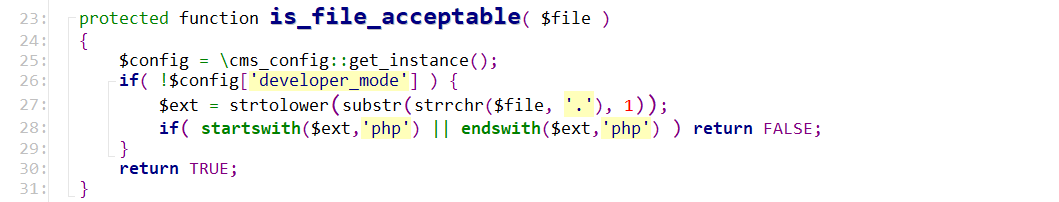
After pressing Enter, any command specified by cmd will be executed, and the result will be displayed on the page.

**Vulnerability Details and POC**

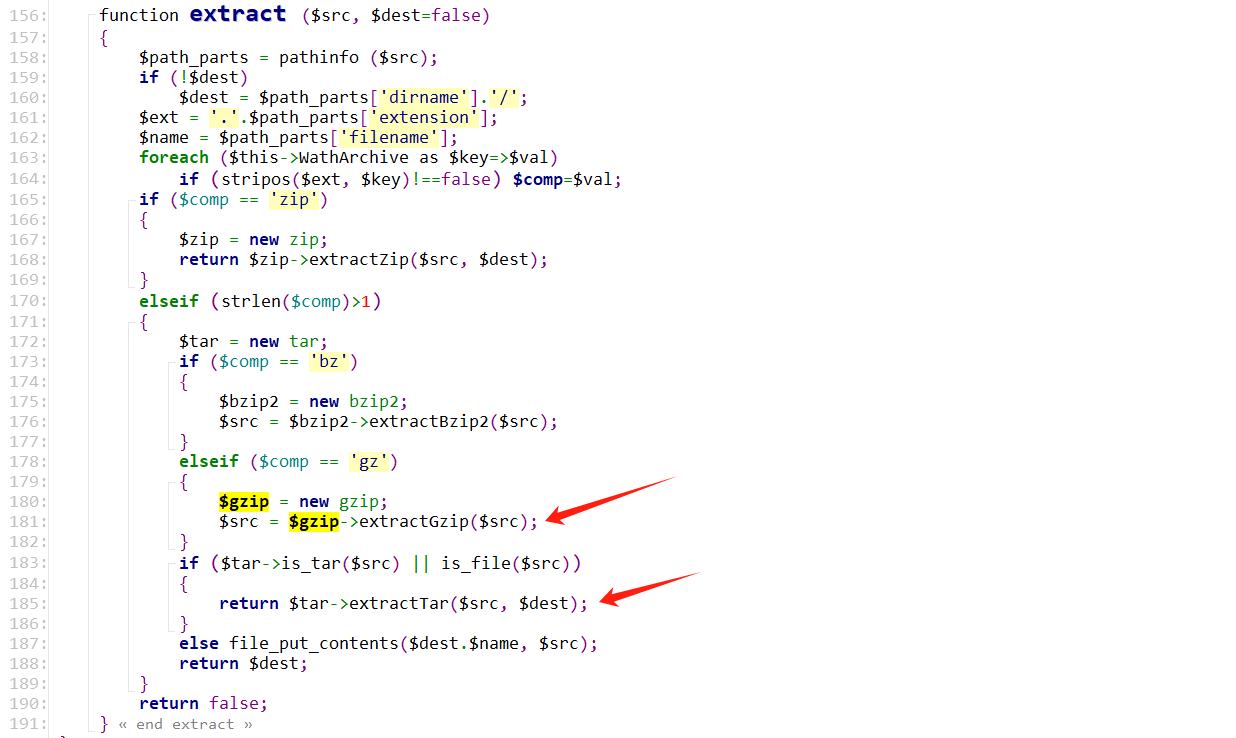
Vulnerable File：

/cmsmadesimple-2.2.21/modules/FileManager/action.upload.php

1、When uploading files, action.upload.php file checks if the file extension is .php; files with the .php extension cannot be uploaded. However, compressing malicious PHP files into a .tar.gz file can bypass this check and result in a successful upload.

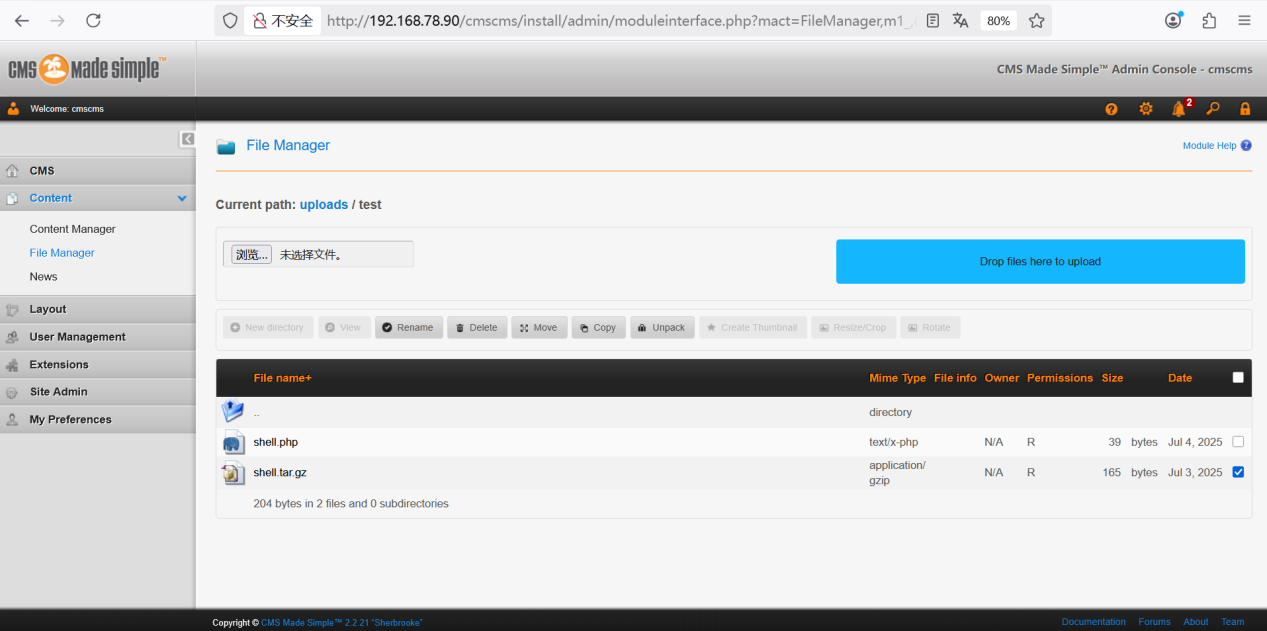


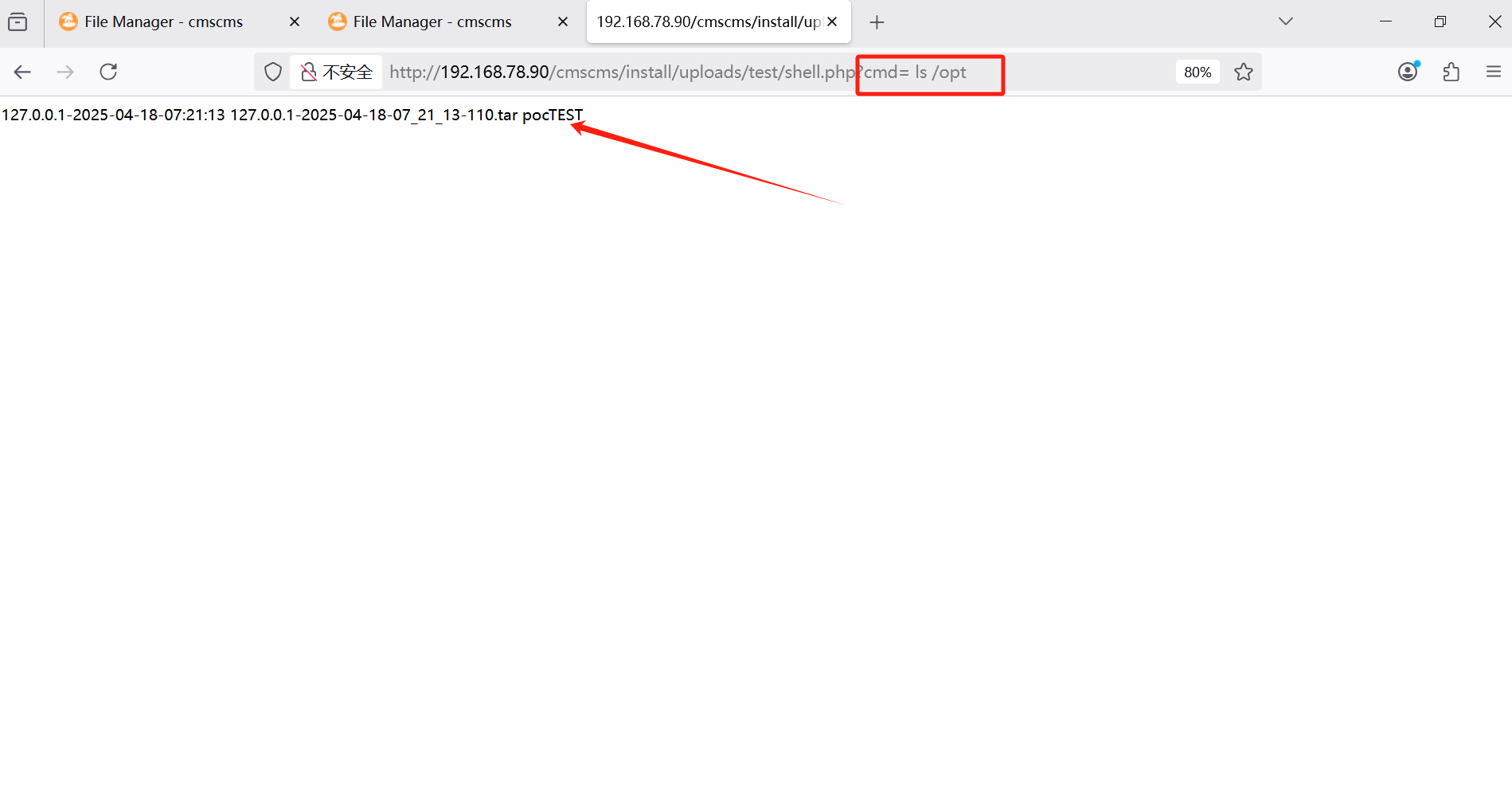
2、During file decompression in EasyArchive.class.php, the suffix of the decompressed files is not checked, leading to arbitrary file uploads.



**Exploitation Verification:**

After uploading, the malicious file can be accessed and executed in the web path under the /uploads/test/ directory.





**Suggested Repair:**

1. Enforce strict MIME type and file extension validation:

Ensure that uploaded files are strictly verified both by extension and MIME type. Reject any file that is not an expected image format (e.g., .jpg, .png, .gif), and inspect the actual file content (magic bytes) to confirm type consistency.

2. Disable execution in upload directories:

Configure the web server to disallow execution of any script within the upload directory (e.g., using .htaccess, web.config, or server configuration). This ensures uploaded files are treated as static content only.

3. Rename and relocate uploaded files:

Store uploaded files using randomized names and move them to directories outside the web root if execution is not needed. This prevents direct web access and arbitrary script execution.

4. Implement Content Security Policies (CSP):

Apply strong CSP headers to reduce the risk of client-side script execution even if a malicious file is served.

5. Sanitize all file metadata and form input:

Validate associated form fields like img\_title or description to prevent metadata-based injection or path manipulation.

6. Monitor and alert on suspicious uploads:

Integrate file upload logging with anomaly detection systems to identify unusual upload patterns or executable content.

7. Conduct periodic security testing and code review:

Regularly perform static code analysis, dynamic testing, and manual audits to discover and mitigate hidden upload logic flaws.