CVE FILE UPLOAD BYPASS CVE FILE UPLOAD BYPASS

CVE-2024-11404: Medium Severity File Upload Vulnerabilities in django-filer 3.2.3

2024-11-20 5 min min read 4767 2

Security Update: Issue Fixed

The fix for this vulnerability has been committed here:

https://github.com/django-cms/django-filer/commit/8f7f96f58a84f224c294a3fdca997cad243d1dd9

Vendor Advisory:

https://www.django-cms.org/en/blog/2024/11/19/security-updates-for-django-filer-and-django-cms-attributes-field/

django Filer introduced file upload validation in version 3. By default, binary or unidentified files could be uploaded and downloaded by a different person and executed by hand on a local machine. To avoid the risk of malware distributed this way, django Filer 3.3 now by default rejects binary files or unknown file types. You can allow them or run them through a virus checker by adjusting your project settings.

We recommend all users of django Filer and django CMS Attributes Field to update to the new versions.

CVE-ID: CVE-2024-11404

CVSS Score:

- CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:L/A:L

Affected Versions:

- django-filer 3.2.3

Impacted CWE Categories:

- **CWE-434**: Unrestricted Upload of File with Dangerous Type
- **CWE-20**: Improper Input Validation
- **CWE-80**: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)

Vulnerability Summary:

The vulnerabilities were identified in **django-filer 3.2.3**, a file management application commonly used with **django CMS**. These issues allow attackers to bypass upload restrictions for HTML and SVG files, potentially uploading malicious files containing scripts that execute on the client side. The following vulnerabilities were observed:

1. **HTML File Upload Bypass**: File extensions like .html are intended to be blocked but can be uploaded by appending a space (**%20**) to the filename.
2. **SVG File Upload Validation Bypass**: The validate_svg function, designed to detect XSS payloads, can be bypassed by appending a space to the filename, allowing the upload of malicious SVG files.

Technical Details:

1. HTML File Upload Bypass

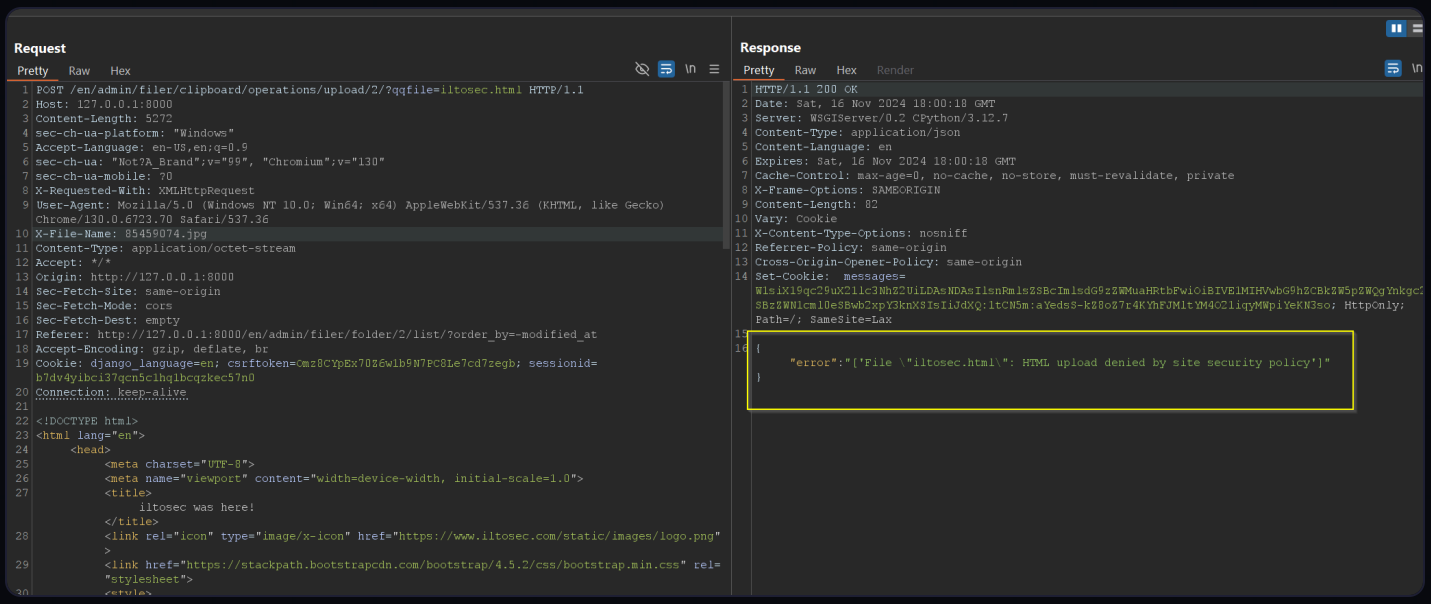
Affected Component:
The issue exists within the media upload functionality at **/admin/filer/folder/** in the application.

a space at the end of the file name (e.g., **iltosec.html%20**), the system allows the file to be uploaded without triggering the restriction.

Reproduction Steps:

Attempt to upload an .html file, such as iltosec.html. The system will return an error message:

{"error": "['File \"iltosec.html\": HTML upload denied by site security policy']"}



Modify the upload request by appending a space to the filename:

iltosec.html%20

Upload the file again. The file will be uploaded successfully and can be accessed at:

http://127.0.0.1:8000/media/filer_public/.../iltosec.html

2. SVG File Upload Validation Bypass

- **Affected Component:**

  The vulnerability is located in the SVG file validation function (**validate_svg**), specifically in how the application validates potential XSS payloads in uploaded SVG files.

However, appending a space to the filename allows the malicious SVG file to bypass the check and be uploaded.

- **Reproduction Steps:**

1. Upload an SVG file containing an XSS payload, e.g., **bypassxssthread1.svg**. The file will be rejected with the error:
   {"error": "['File \"bypassxssthread1.svg\": Rejected due to potential cross site scripting vulnerability']"}
2. Modify the upload request by adding a space to the filename (**bypassxssthread1.svg%20**).
3. The file is successfully uploaded, bypassing the XSS validation.

Proof of Concept:

1. HTML File Upload Bypass

The issue arises when the system fails to properly handle filenames with appended spaces. This can lead to the upload of malicious HTML files that may contain JavaScript, potentially resulting in Cross-Site Scripting (XSS) or arbitrary code execution.

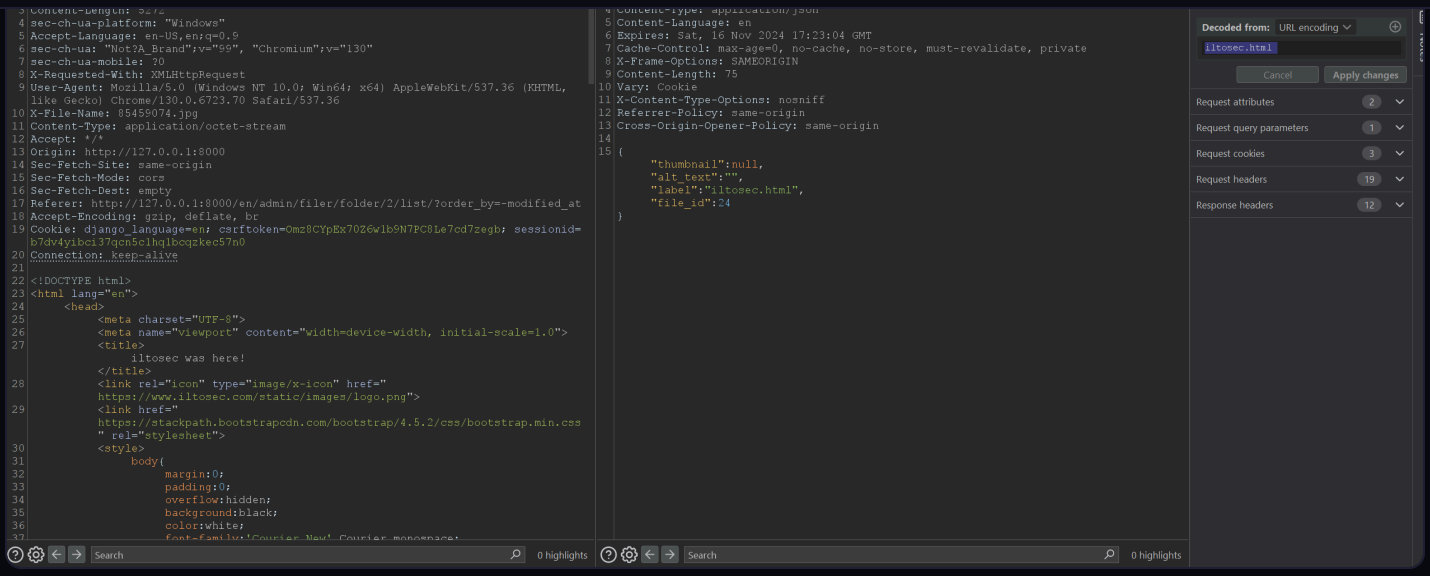
Steps to Reproduce:

Open the admin interface for file uploads at /en/admin/filer/folder/.

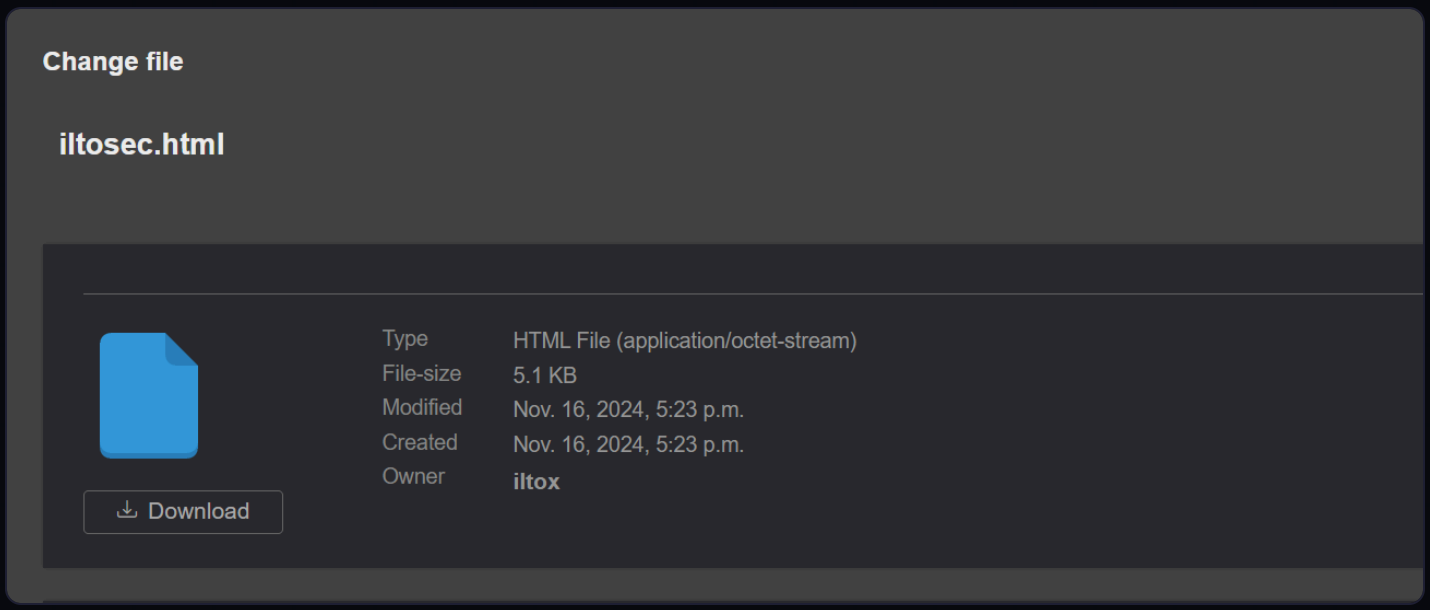
Try to upload a file with the extension .html, such as iltosec.html. The error message will indicate that HTML uploads are denied.

Modify the upload request by appending a space (%20) to the file name, e.g., iltosec.html%20.

ILTOSEC

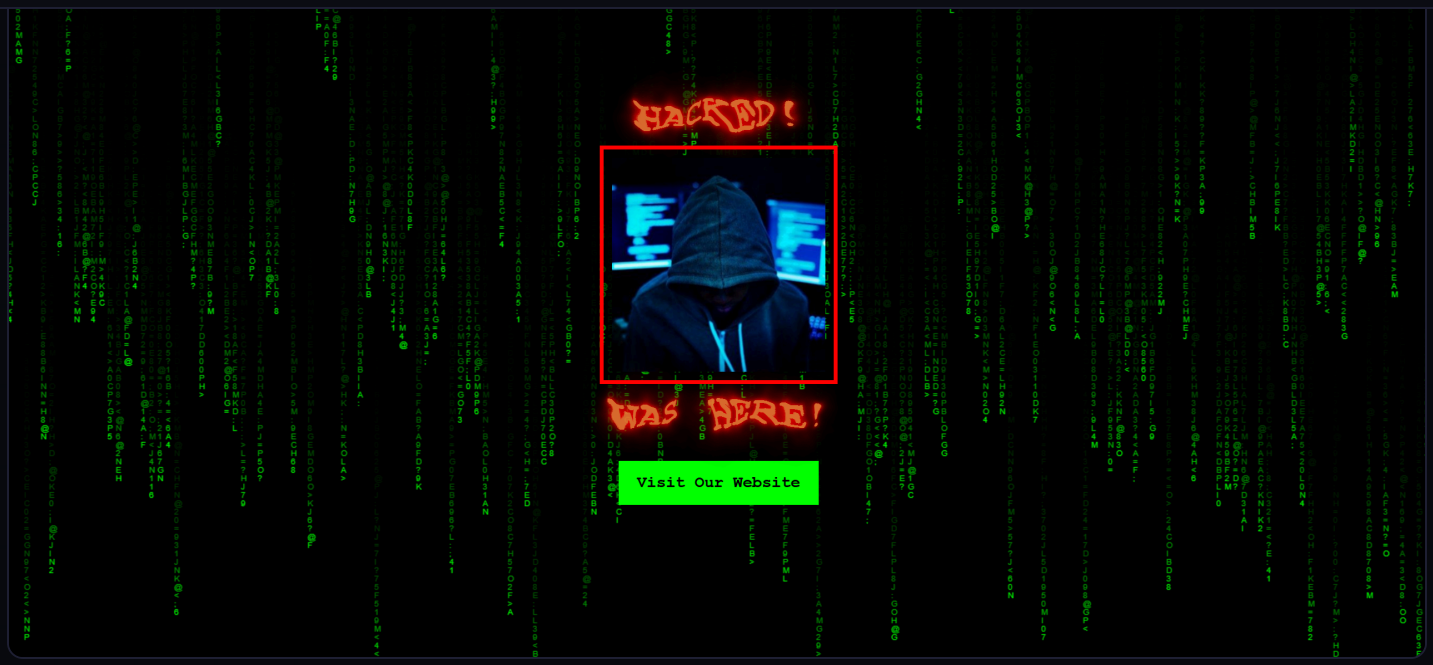


Upload the file with the modified filename. The file will upload successfully.



Access the uploaded file via the URL:

http://127.0.0.1:8000/media/filer_public/.../iltosec.html.



2. SVG File Upload Validation Bypass

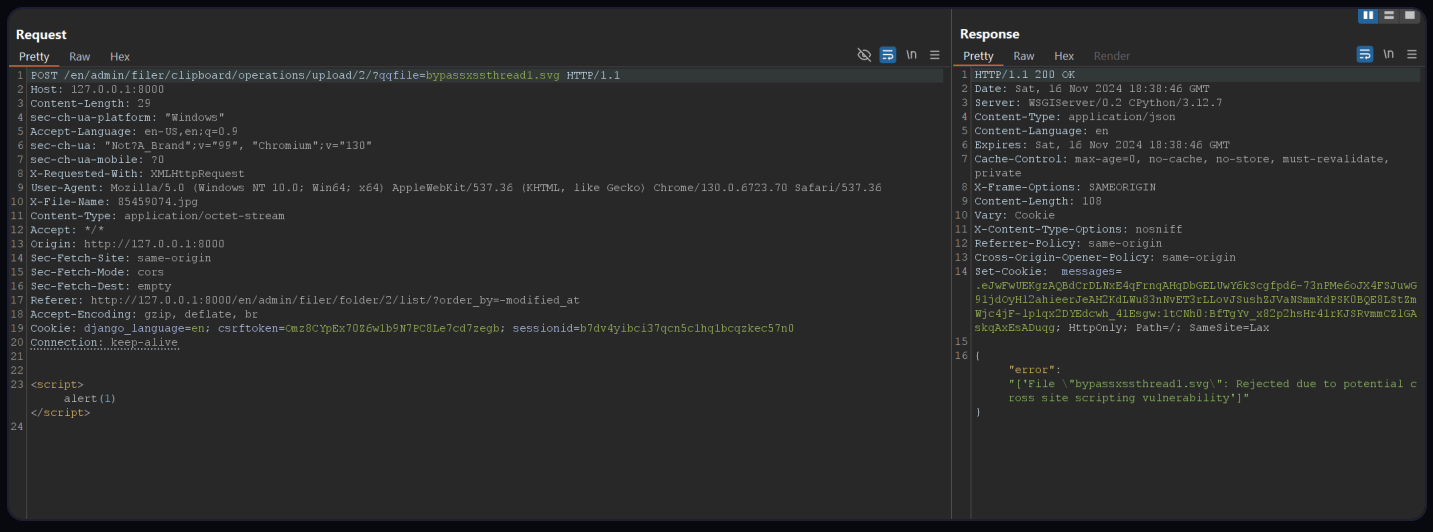
Steps to Reproduce:

Open the admin interface for SVG file uploads.

Attempt to upload an SVG file containing an XSS payload (e.g., **bypassxssthread1.svg**).

The file will be rejected, showing the error:

{"error": "['File \"bypassxssthread1.svg\": Rejected due to potential cross site scripting vulnerability']"}.



Modify the filename by appending a space (**%20**), resulting in **bypassxssthread1.svg%20**.

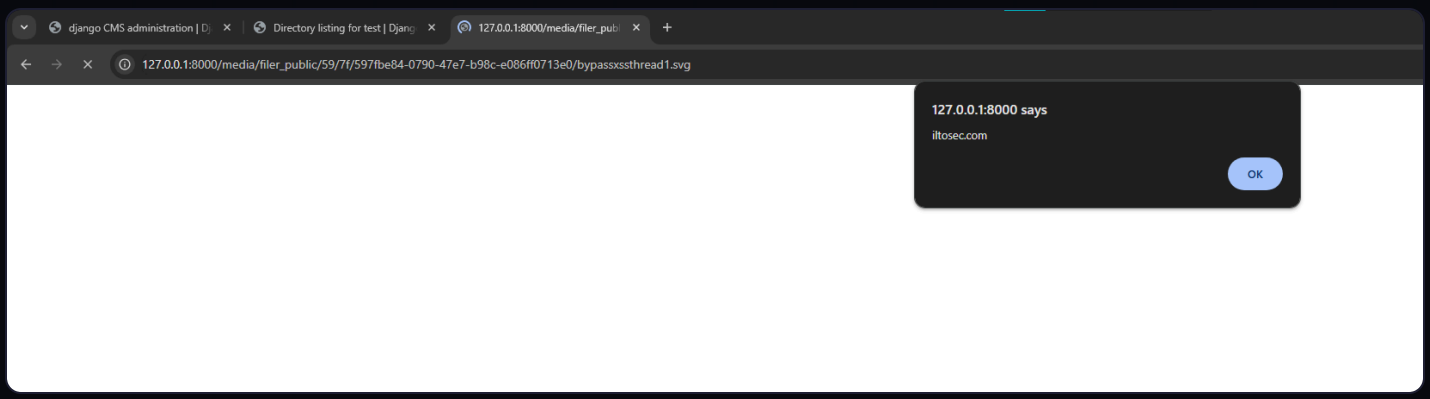
ILTOSEC

```
Content-Length: 354
sec-ch-ua-platform: "Windows"
Accept-Language: en-US,en;q=0.9
sec-ch-ua: "Not?A_Brand";v="99", "Chromium";v="130"
sec-ch-ua-mobile: ?0
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.6723.70 Safari/537.36
X-File-Name: 85459074.jpg
Content-Type: application/octet-stream
Accept: */*
Origin: http://127.0.0.1:8000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://127.0.0.1:8000/en/admin/filer/folder/2/list/?order_by=-modified_at
Accept-Encoding: gzip, deflate, br
Cookie: django_language=en; csrftoken=Cmz8CYpEx70Z6wlb9N7PC8Le7cd7zegb; sessionid=b7dv4yibci37qcn5clhqlbcqzkec57n0
Connection: keep-alive


<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">
    <polygon id="triangle" points="0,0 0,50 50,0" fill="#009900" stroke="#004400"/>
    <script type="text/javascript">
        alert('iltosec.com');
        </script>
    </svg>
```

```
Server: WSGIServer/0.2 CPython/3.12.7
Content-Type: application/json
Content-Language: en
Expires: Sat, 16 Nov 2024 18:43:15 GMT
Cache-Control: max-age=0, no-cache, no-store, must-revalidate, private
X-Frame-Options: SAMEORIGIN
Content-Length: 83
Vary: Cookie
X-Content-Type-Options: nosniff
Referrer-Policy: same-origin
Cross-Origin-Opener-Policy: same-origin

{
    "thumbnail":null,
    "alt_text":"",
    "label":"bypassxssthread1.svg",
    "file_id":49
}
```

Upload the modified file. The file will successfully bypass the validation and be uploaded.

The malicious SVG file can now be executed to carry out potential XSS attacks.




```python
def validate_svg(file_name: str, file: typing.IO, owner: User, mime_type: str) -> None:
    """SVG files must not contain script tags or javascript hrefs.
    This might be too strict but avoids parsing the xml"""
    content = file.read().lower()
    if any(map(lambda x: x in content, TRIGGER_XSS_THREAD)):
        # If any element of TRIGGER_XSS_THREAD is found in file, raise FileValidationError
        raise FileValidationError(
            _('File "{file_name}": Rejected due to potential cross site scripting vulnerability')
            .format(file_name=file_name)
        )
```

ILTOSEC

```
        _('File "{file_name}": HTML upload denied by site security policy').format(file_name=file_name)
    )


TRIGGER_XSS_THREAD = (
    # Part 1: Event attributes that take js code as values
    # See https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/Events
    b"onbegin=", b"onend=", b"onrepeat=",
    b"onabort=", b"onerror=", b"onresize=", b"onscroll=", b"onunload=",
    b"oncopy=", b"oncut=", b"onpaste=",
    b"oncancel=", b"oncanplay=", b"oncanplaythrough=", b"onchange=", b"onclick=", b"onclose=", b"oncuechange=", b"ondblclick=",
    b"ondrag=", b"ondragend=", b"ondragenter=", b"ondragleave=", b"ondragover=", b"ondragstart=", b"ondrop=",
    b"ondurationchange=", b"onemptied=", b"onended=", b"onerror=", b"onfocus=", b"oninput=", b"oninvalid=",
    b"onkeydown=", b"onkeypress=", b"onkeyup=", b"onload=", b"onloadeddata=", b"onloadedmetadata=", b"onloadstart=",
    b"onmousedown=", b"onmouseenter=", b"onmouseleave=", b"onmousemove=", b"onmouseout=", b"onmouseover=", b"onmouseup=",
    b"onmousewheel=", b"onpause=", b"onplay=", b"onplaying=", b"onprogress=", b"onratechange=", b"onreset=", b"onresize=",
    b"onscroll=", b"onseeked=", b"onseeking=", b"onselect=", b"onshow=", b"onstalled=", b"onsubmit=", b"onsuspend=",
    b"ontimeupdate=", b"ontoggle==", b"onvolumechange==", b"onwaiting=",
    b"onactivate=", b"onfocusin=", b"onfocusout=",
) + (
    # Part 2:
    # Reject base64 obfuscated content
    b";base64,",
) + (
    # Part 3: Obvious scripts
    # Reject direct <scrpit> tags or javascript: uri
    b"<script",
    b"javascript:",
)
```

Impact

Both vulnerabilities pose significant security risks:

1. **Vulnerability 1 (Media File Upload Bypass):**
   Allows the upload of HTML files that can contain malicious scripts, which may lead to Cross-Site Scripting (XSS) attacks or the execution of arbitrary code when viewed by users.

2. **Vulnerability 2 (SVG File Upload Bypass):**
   Allows attackers to upload SVG files with embedded XSS payloads, potentially compromising user sessions, stealing cookies, or performing unauthorized actions on behalf of the user.

Both vulnerabilities can weaken the overall trust and security of the web application.

Remediation:

- **Vulnerability 1 (Media File Upload Bypass):**

- Ensure file names are sanitized and reject names with appended spaces or special characters.

- **Vulnerability 2 (SVG File Upload Bypass):**
  - Strengthen the **validate_svg** function to handle edge cases, such as filenames with appended spaces.
  - Implement more comprehensive content inspection to prevent the upload of malicious scripts.

Timeline:

Date	Status
16-NOV-2024	Reported to vendor
18-NOV-2024	Vendor acknowledgement
20-NOV-2024	Vulnerability fixed
20-NOV-2024	Patch available
20-NOV-2024	Public Disclosure

RELATED POSTS

CVE-2026-48493: Privilege Escalation via Permission Bypass in Snipe-IT

Technical breakdown of CVE-2026-48493: Users with users.edit permission escalate to near-full system access via PreserveUnauthorizedPrivilegedPermissionsAction bypass. Detailed PoC and impact analysis.

2026-05-28 61

Technical breakdown of CVE-2026-48492: A missing authorization flaw in Snipe-IT allowing authenticated users to enumerate accounts via the API.

2026-05-27 77

RCE

FacturaScripts <= 2026 Authenticated RCE via Malicious Plugin Upload

Detailed vulnerability analysis of an Authenticated Remote Code Execution (RCE) in FacturaScripts (<= 2026). Explore the PoC via malicious plugin upload and learn about server hardening mitigations.

2026-05-01 186

RCE · CMS · FILE UPLOAD BYPASS

EspoCRM v9.3.4 Authenticated Remote Code Execution via Malicious Extension Upload

Explore the technical analysis of the Authenticated Remote Code Execution (RCE) vulnerability in EspoCRM <= v9.3.4. Learn how malicious extension uploads can lead to full OS command execution and find mitigation strategies. Official PoC and exploit details included.

2026-04-13 276