SERIES 0-DAY NO CVE HIGH PUBLIC

Jellyfin 12.2, gzip data:image/png leftover of GHSA-cf3c

Jellyfin Server 12.2 still pins Svg.Skia 3.7.0. SvgSecurityValidator allows non-svg data URIs without walking gzip. Rasterize of a profile image still fetches http and file. Leftover of GHSA-cf3c. No CVE yet.

05 SERIES Jellyfin 12.2 leftovers

- 00 The pin
- 00 Jellyfin, EncodingUtils -i leftover of GHSA-wwwm
- 00 Jellyfin, public playlist ItemIds leftover of GHSA-9x85
- 00 Jellyfin 12.2, Suggestions omit-userId hidden-library dump
- 00 Jellyfin 12.2, gzip data:image/png leftover of GHSA-cf3c

← PREVIOUS
Phase 0: Jellyfin 12.2, Suggestions omit-userId hidden-library dump

NAME	SERIES
Jellyfin 12.2, gzip data:image/png leftover of GHSA-cf3c	Jellyfin 12.2 leftovers
PHASE	TYPE
0	0-day analysis
CVE	CVE RISK
n/a	high

DISCLOSURE STATUS	VENDOR
public	Jellyfin
AFFECTED	**PUBLISHED**
Jellyfin Server 12.2 (ca0f16eb), image jellyfin/jellyfin:12.2, Svg.Skia 3.7.0. Any authenticated user POST /UserImage. Rasterize is unauth GET /UserImage.	17 Oct 2026
UPDATED	**TAGS**
17 Oct 2026	0-day, jellyfin, ssrf, svg, gzip, cwe-918, cwe-611, cwe-400, authenticated

TL;DR

Lab files and proof-of-concept here.

01 The validator believed the MIME type. The library believed the magic.

I am @abraxas_null. This is leftover **#2** of the Jellyfin 12.2 leftovers series. Same pin as #1 Suggestions IDOR, playlist ItemIds, and FFmpeg colon-slash. The proof of concept is on GitHub: **abraxas/jellyfin-svg-gzip** (loopback client). The lab stack is `lab/` : Dockerfile, docker-compose.yml, run.sh, poc.py, oracle.py. Authorized lab only. It talks to loopback. AGPLv3. Witness-only. No shells.

This is **Jellyfin Server v12.2** (`ca0f16eb7c195f720a1493ed469a27c4657db0c6`), image `jellyfin/jellyfin:12.2` . Svg.Skia **3.7.0**. No CVE yet. CWE-918 / CWE-611 / CWE-400. 7.7 High (`CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:L`). **Reach:** Remote. **Class:** SSRF + file read + memex. Any authenticated user. Rasterize is unauthenticated.

GHSA-cf3c-8m59-2vhx taught 12.0 a lesson about profile SVGs that fetch `http` and `file` . The patch is a Jellyfin pre-scan plus three `ExternalType.None` assignments.

12.2 still pins the library that ignores those assignments. The pre-scan walks gzip only after it has already decided the payload is SVG. Lie about the media type. Gzip an inner SVG. The library `LoadSvgz` s it. Inner hrefs go to `WebRequest.Create`. That is the original sink, still live, behind a MIME comment that is wrong.

Direct outer `http` hrefs stay blocked. I labbed that as the negative. The leftover is the wrapper the validator does not open.

02 Why Jellyfin

I had just finished a wave of self-hosted media and forge leftovers. Jellyfin is the media server people actually run when they do not want Plex's account, on a LAN they treat as friendly, with a guest user so a cousin can pick a movie. Profile images are not an admin surface. SVG is a valid upload type. Rasterize is an unauthenticated GET. That combination is how GHSA-cf3c happened, and it is how you ask whether 12.2 finished the job.

I only wanted Critical or High. Remote preferred. Medium stays on the floor unless it chains. jellyfin-web-only XSS, Wizarr, Seerr, and jellyfin-ios are other trees. I wrote that on the skip list before I opened a controller.

Jellyfin's intake is security@jellyfin.org. Unpaid. No public GitHub issue on jellyfin/jellyfin.

03 Pinning v12.2

Latest GA on the day I cloned. Tag **v12.2**, commit `ca0f16eb7c195f720a1493ed469a27c4657db0c6`. Image `jellyfin/jellyfin:12.2`. `Directory.Packages.props` says Svg.Skia **3.7.0**. SkiaSharp 3.119.4. That is the pin. Master is not the pin. An RC is not the pin.

12.2 is also the tag that already shipped the 12.0 GHSA wave: VirtualFolders Path.Combine, SVG Skia external refs, session remote-control, playlist Items userId,

HLS traversal, plugin-image traversal, FFmpeg quote-escape on the File branch, and the rest. If you pin the release that shipped the pre-scan, you get to ask which payloads the pre-scan still treats as opaque raster. That is a better question than "does SVG still fetch somewhere."

04 The skip list is half the hunt

Before you look for leftovers you write down what is already public so you do not rediscover a GHSA with a press release. On 12.2 the interesting rows were already closed:

- GHSA-cf3c-8m59-2vhx SVG Skia external refs. Direct `http` / `file` / relative hrefs, `<use>`, CSS `url()`, `@import`, nested `data:image/svg+xml` (plain, base64, gzip), DTD `SYSTEM`, entity expansion over 1 MiB. That is a full external-ref disable at the **pre-scan**. Direct GHSA-cf3c payloads now log "Refusing to render SVG". This post is the media-type mismatch the pre-scan does not walk.
- GHSA-vcmh-9wx9-rfqh SVG stored XSS. `GetImageResult` sets `Content-Disposition: attachment`. Successful rasterize is not `image/svg+xml`.
- GHSA-rgjw-4fwc-9v96 unauth remote-image GET. `GetRemoteImages` is `[Authorize]`. `DownloadRemoteImage` is `RequiresElevation`. Admin artwork fetch of an admin-chosen URL is the feature.
- GHSA-c39p-cgf8-866g plugin-image path. Resolves under `plugin.Path`.
- GHSA-rrr6-mvwg-9pg9 branding splash. Admin-only POST. GET goes through attachment plus rasterize.
- GHSA-6828-c7cx-hvqm VirtualFolders Path.Combine. Closed.

- GHSA-4vx8-xhc9-qg6x session remote-control. Closed.
- GHSA-9×85-gx46-6522 playlist Items userId. Closed on `/Items`. The leftover on `GET /Playlists/{id}` `ItemIds` is a sibling, not this pack.
- GHSA-wwwm-px48-fpvq FFmpeg subtitle path. File-branch quotes escaped. The leftover on non-File `-i` is a sibling.
- GHSA-jh22-fw8w-2v9x StreamOptions. GHSA-8fw7-f233-ffr8 LiveTV M3U. GHSA-j2hf-x4q5-47j3 subtitle `.strm`. GHSA-qcmf-gmhm-rfv9 XFF. GHSA-jg92-mrxq-vv75 ClientLog. Closed.
- Older: GHSA-866x codec argv, GHSA-2c3c leftover of that, GHSA-9p5f ClientLog write, GHSA-rr9h encoder path, GHSA-wg4c unauth file, GHSA-hw73 HLS, GHSA-f47c MKV attachment, GHSA-v2jv SyncPlay, GHSA-fv79 user-management XSS.

I am not re-filing any of that.

05 Twenty surfaces, four leftovers

I split the tree the way you split a media server: first-run wizard, path-combine, SVG and image decode, session remote-control, plugins, LiveTV M3U, subtitle and `.strm`, QuickConnect, backup, DLNA, branding splash, SyncPlay, transcode and trickplay, WebSocket, system and encoder path, library download, ClientLog, metadata SSRF, playlist and HLS, authz and IDOR.

Most of that is Jellyfin doing what the 12.0 / 12.2 changelog said it would do.

Wizard cannot be re-run unauth on a configured server. `FilePathJoinAbs` siblings on VirtualFolders, ClientLog, HLS, plugin image, attachments, subtitles, backup zip: closed or below bar. Session remote-control is GHSA-4vx8. Plugins reject unsafe package names. LiveTV M3U is GHSA-8fw7. QuickConnect, backup, DLNA (plugin-

only), splash, SyncPlay, transcode prefix, WebSocket, encoder path, library download, ClientLog: empty at this bar. `POST /Items/{id}/RemoteImages/Download?imageUrl=` is still a raw `NamedClient.Default.GetAsync` with no private-IP deny. It is `RequiresElevation`. Artwork download of an admin-chosen URL is the feature. I dropped it.

Four leftovers labbed SUCCESS on the pin. #1 is `GET /Items/Suggestions` with no `userId`: the handler sets `user = null` instead of claims, so the query never `AddUserToQuery`. This post is the SVG one. Playlist `ItemIds` and FFmpeg non-File `-i` are the other two. I do not steal their stories here.

06 The leftover is a MIME lie plus gzip magic

`SkiaEncoder` still sets the Svg.NET flags:

PLAIN TEXT Copy

```
static SkiaEncoder()
{
    SvgDocument.ResolveExternalElements = ExternalType.None;
    SvgDocument.ResolveExternalImages = ExternalType.None;
    SvgDocument.ResolveExternalXmlEntites = ExternalType.None;
}
```

Both rasterize entry points pre-scan, then invoke Svg.Skia. `EncodeImage` always takes the SVG path when the extension is `.svg`. 12.2 still pins Svg.Skia **3.7.0**. Upstream wieslawsoltes/Svg.Skia#448 (opened 2025-12-02, closed 2026-05-11 in #501): `ImageDrawable` calls `SvgService.GetImage` without checking `ResolveExternalImages`. 3.7.0 `ImageDrawable.Create` matches that. Jellyfin's open renovate PR #17076 (Svg.Skia 5.x) is **not** in v12.2.

So `ExternalType.None` does not stop `<image href="http://...">` / `file://` on this pin. The pre-scan is the only gate.

`ValidateReference` allows only empty, same-document `#fragment`, and `data:` URIs. Everything else (`http`, `https`, `file`, relative `../../etc/hosts`, `ftp`, `javascript`) is rejected. Tests cover those plus `<use>`, CSS `url()`, `@import`, nested `data:image/svg+xml` (base64 / URL-encoded / gzip), XXE `SYSTEM`, and billion-laughs. That is a real patch. Direct GHSA-cf3c payloads fail closed.

Nested SVG is re-parsed **only** when the data URI media type is exactly `image/svg+xml`. Any other type is treated as opaque raster and skipped, including gzip:

PLAIN TEXT Copy

```
// Only "image/svg+xml" is re-parsed as SVG by the renderer; any other type is tr
if (!mediaType.Contains('/') || !mediaType.Equals("image/svg+xml", StringComparis
{
    return null;
}
```

`return null` here is allow. The gzip walk (`0x1F 0x8B`, 16 MiB cap) runs **after** that return. The comment is a bet that Svg.Skia agrees.

Svg.Skia 3.7.0 `GetImageFromDataUri` does not agree:

- `mimeType == image/svg+xml` (ignore-case) → `LoadSvg` / `LoadSvgz`.
- `mimeType.StartsWith("image/")` or `"img/"` → if the decoded bytes are gzip, `LoadSvgz` **anyway**, else `assetLoader.LoadImage`.

So `<gzip(inner.svg)>` is:

01 Validator: media type `image/png` → allow, do not inspect payload.

02 `ImageDrawable`: `GetImage` → gzip magic → `LoadSvgz` of the inner SVG (no Jellyfin re-scan).

03 Inner `<image href="http://...">` / `file://...` → `GetImageFromWeb` → `WebRequest.Create(uri).GetResponse()` with **no** `ExternalType` check.

`GetImageFromWeb` uses obsolete `WebRequest.Create`, which still speaks `http`, `https`, and `file`. That is the original GHSA-cf3c sink.

Direct `<gzip(evil.svg)>` **is** re-scanned. The leftover is the wrong media type, not gzip itself. Nested `data:image/svg+xml` wrapping the same trick is also re-scanned. The hole is "I am a PNG" plus gzip magic.

07 Reach: any user uploads, nobody needs to be signed in to fire it

`POST /UserImage` is `[Authorize]`, `[AcceptsImageFile]`. `image/svg+xml` is a valid `TryGetImageExtensionFromContentType` type. The body is BASE64 of the image bytes (`FromBase64Transform`). Content-Type maps to `.svg`. Any signed-in user may set their own profile image. Admin is not required.

`GET /UserImage` has **no** `[Authorize]`. Comment in the controller: "UserId is required if unauthenticated." `GetOutputFormats` never includes `ImageFormat.Svg` (webp/jpg/png only), so `ImageProcessingOptions.HasDefaultOptions` is false for a `.svg` source and `EncodeImage` / `GetBitmapFromSvg` always runs. A rejected SVG falls through `ImageProcessor.ProcessImage` catch to the original file. A validator-pass SVG is rasterized. That is why the negative stayed `image/svg+xml` and the gzip-png wrapper came back `image/png`.

`GET /Items/{itemId}/Images/{imageType}` is likewise unauthenticated and passes `maxWidth` into `EncodeImage`. Local artwork still includes `.svg`. A library `poster.svg` is the same sink after an admin import. The lab used the profile path because any user can write it.

Every later unauth image GET re-renders. You upload once. The fetch repeats.

08 How the lab actually runs

I wanted a catcher Skia would fetch and a control the validator would refuse. The image is `jellyfin/jellyfin:12.2`. Compose puts Jellyfin on `127.0.0.1:18780` and an oracle on `127.0.0.1:18781`, hostname `oracle` on the overlay so Skia talks to `http://oracle:8080`. The oracle returns a 1×1 PNG and appends the request path to a bind-mounted log. Witness string `JELLYFIN-SVG-SVGSKIA-WITNESS`. No IMDS addresses in the pack. Loopback.

First-run wizard: `POST /Startup/User`, remote access, complete, then `POST /Users/AuthenticateByName`. Then:

01 Reset the oracle log.

02 `POST /UserImage` with an outer SVG whose `<image href>` is `http://oracle:8080/negative.png`. Unauth `GET /UserImage?userId=`. Content-Type stays `image/svg+xml`. Oracle 0. That is GHSA-cf3c still holding the outer document.

03 Gzip an inner SVG whose href is `http://oracle:8080/JELLYFIN-SVG-SVGSKIA-WITNESS.png`. Wrap as `data:image/png;base64` on a new outer SVG. Same POST, same GET. Content-Type is `image/png`. Oracle 1. Path contains the witness.

PLAIN TEXT Copy

```
1 SUCCESS jellyfin-svg-gzip gzip-png-ssrf hits=1 negative=0 JELLYFIN-SVG-SVGSKIA-WI
```

Replay: `python3 jellyfin-svg-gzip-Abraxas-Labs.py` or `cd lab && ./run.sh`. Loopback. Witness-only. I am not printing an SVG you can paste at someone else's `/UserImage`. The MIME disagreement is the useful part.

The POST body is BASE64 of the SVG bytes. Send raw XML with `Content-Type: image/svg+xml` and the server's `FromBase64Transform` will not like you. That cost a minute and looked, for that minute, like 12.2 had finished the job. It had not. It wanted the body encoded.

09 Wrong turns, in the order I made them

Treating GHSA-cf3c as closed because the negative control stays `image/svg+xml`. That is the patch working on the outer document. The leftover is the inner document the validator never opens. A FAIL on the direct href is the control, not the finding.

Nested `data:image/svg+xml` gzip. Re-scanned. Tests cover it. The leftover is the declared type, not compression.

DTD `SYSTEM` on the outer SVG. `XmlResolver = null`, internal-subset `SYSTEM` reject, `ResolveExternalXmlEntites = None`. Closed on the outer document. I am not dressing XXE up as this bug.

GHSA-vcmh inline XSS. Attachment disposition plus rasterize. Admin-viewed inline script is the old bug. Not High, and the attachment patch is present.

Plugin-image traversal. GHSA-c39p. Resolves under `plugin.Path`. Closed.

Magick / PDF / ICO as a second SSRF parser. No Magick.NET on this pin. ICO is `SKCodec`. PDF is not a supported image type. Not this leftover.

Admin `RemoteImages/Download`. Intended artwork fetch. `RequiresElevation`. No private-IP deny, still admin-only. Dropped.

Pixel-bomb DoS. `GetBitmapFromSvg` allocates `Drawable.Bounds` with no cap. `POST /UserImage` has no `[RequestSizeLimit]`. Generic image-bomb. GHSA-cf3c's High memex was unbounded **external** reads (`file:///dev/urandom`) that this leftover still reaches through the gzip wrapper. I did not file the pixel bomb as a second High.

Wanting `ExternalType.None` to save it. It would, on a library that consults the flag. 3.7.0 `ImageDrawable` does not. The three assignments in `SkiaEncoder` are a comment that does not run.

A reverse shell. Theatre. The witness is one GET on the oracle with the witness in the path.

10 What an attacker can do

Who: any authenticated Jellyfin user. Profile image is not admin-only. Rasterize is unauthenticated `GET /UserImage?userId=`. This is the privilege set. It is not RCE, not session XSS, and not a plugin install.

AXIS	METHOD + PATH	GRANTS	PROVENANCE
Recon	`POST /UserImage` (`Content-Type: image/svg+xml`, BASE64 body) then unauth `GET /UserImage?userId=`	Server GET of attacker `http` / `https` (loopback, RFC1918, link-local/IMDS) while Skia rasterizes	**Lab-proven.** Oracle hostname `oracle:8080`, hits=1, witness in path
Recon	same	`file://` bytes painted into the PNG/WebP/JPEG the requester downloads, if they decode as an image	Inferred. Same `WebRequest.Create` sink as GHSA-cf3c
Availability	same, inner `file:///dev/urandom`	Unbounded read retained on the managed heap. Repeats on every later image GET without re-upload	Inferred. Measured in GHSA-cf3c

AXIS	METHOD + PATH	GRANTS	PROVENANCE
Integrity	none	No user-policy write, no plugin install, no session steal	Closed

Stays closed: direct `http` / `file` hrefs on the outer SVG (lab: GET stayed `image/svg+xml`, oracle 0); nested `data:image/svg+xml` (re-scanned, including gzip); DTD `SYSTEM`; GHSA-vcmh inline XSS (`Content-Disposition: attachment` plus rasterize); plugin-image traversal; Magick/PDF. Not RCE. Not session XSS. SSRF body is what Skia paints, not the raw HTTP response. You do not get a shell from this leftover. You get the server to fetch, and you get whatever of that fetch Skia is willing to draw.

Changed-scope is the IMDS / RFC1918 / loopback fetch from a process that was not supposed to talk to those hosts while rendering a cousin's avatar. That is why it is High and why `S:C` is in the vector.

11 The lab, again, as a recipe

PLAIN TEXT Copy

```
git clone https://github.com/abraxas/jellyfin-svg-gzip
cd jellyfin-svg-gzip
python3 jellyfin-svg-gzip-Abraxas-Labs.py
```

Loopback: Jellyfin `http://127.0.0.1:18780`, oracle `http://127.0.0.1:18781`. Image `jellyfin/jellyfin:12.2`. Last line:

PLAIN TEXT Copy

```
SUCCESS jellyfin-svg-gzip gzip-png-ssrf hits=1 negative=0 JELLYFIN-SVG-SVGSKIA-WI
```

12 The fix

Walk gzip and nested data URIs in `SvgSecurityValidator` even when the declared media type is not `image/svg+xml`. The comment at lines 169-173 is the bug. Treat unknown `image/*` as opaque only after you have checked that the bytes are not gzip-compressed SVG.

And bump Svg.Skia past the `ImageDrawable` `ExternalType.None` ignore. Upstream #448 / #501 already did that work. Renovate #17076 is not in this tag. The library flag is the intended kill switch. The validator should not lie about what the library will parse. Either fix alone is a race with the other parser. Both.

13 References

- Proof of concept: abraxas/jellyfin-svg-gzip · jellyfin-svg-gzip-Abraxas-Labs.py
- Lab: `lab/` · Dockerfile · docker-compose.yml · run.sh · poc.py · oracle.py
- @abraxas_null · github.com/abraxas · abraxaslabs.tech · abraxas.null@proton.me
- Series: Jellyfin 12.2 leftovers · leftover #1 Suggestions IDOR · playlist ItemIds · FFmpeg colon-slash
- Tree: jellyfin/jellyfin v12.2 (`ca0f16eb7c195f720a1493ed469a27c4657db0c6`) · `SvgSecurityValidator.cs` · `SkiaEncoder.cs` · `ImageController.cs`
- Closed, not this leftover: GHSA-cf3c-8m59-2vhx · GHSA-vcmh-9wx9-rfqh
- Upstream: Svg.Skia#448 · #501. Renovate not in tag: jellyfin#17076

CWE-918 · CWE-611 · CWE-400

05 SERIES Jellyfin 12.2 leftovers

- 00 The pin
- 00 Jellyfin, EncodingUtils -i leftover of GHSA-wwwm
- 00 Jellyfin, public playlist ItemIds leftover of GHSA-9x85
- 00 Jellyfin 12.2, Suggestions omit-userId hidden-library dump
- 00 Jellyfin 12.2, gzip data:image/png leftover of GHSA-cf3c

← PREVIOUS
Phase 0: Jellyfin 12.2, Suggestions omit-userId hidden-library dump

A X Abraxas Labs

Independent security research by abraxas.
Analysis of 0-days, n-days, malware, and the
leftovers that still matter.

GitHub @abraxas_null abraxas.null@proton.me abraxaslabs.tech

Independent research. Private analysis by arrangement. No ads. abraxas.null@proton.me · For defensive awareness and authorized testing only.

© 2026 Abraxas Labs