SERIES 0-DAY NO CVE HIGH PUBLIC

Jellyfin, public playlist ItemIds leftover of GHSA-9x85

Jellyfin Server 12.2 GET /Playlists/{id} still returns unfiltered ItemIds. GHSA-9×85 patched the Items list. The playlist object and unauth video stream were not given that filter. A restricted user reads private GUIDs and plays the files with no token. Labbed. 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, EncodingUtils -i leftover of GHSA-wwwm

NEXT →
Phase 0: Jellyfin 12.2, Suggestions omit-userId hidden-library dump

NAME	SERIES
Jellyfin, public playlist ItemIds leftover of GHSA-9×85	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. Public playlist ItemIds leftover of GHSA-9×85. Unauth GET /Videos/{id}/stream.	17 Oct 2026
UPDATED	**TAGS**
17 Oct 2026	0-day, jellyfin, idor, playlist, cwe-639, cwe-862, authenticated

TL;DR

Lab files and proof-of-concept here.

01 They patched the list. They left the object.

I am @abraxas_null. The proof of concept is on GitHub: **abraxas/jellyfin-playlist-itemids** (loopback client). The lab stack is `lab/`: docker-compose.yml, run.sh, poc.py. Authorized lab only. It talks to loopback.

This is leftover **#3** of the Jellyfin 12.2 leftovers series. Leftover #1 on this pin, and the most important, is Suggestions omit-userId. The other two SUCCESS leftovers are gzip `data:image/png` SVG and FFmpeg `-i` quote-break.

This is **Jellyfin Server v12.2** (`ca0f16eb7c195f720a1493ed469a27c4657db0c6`), image `jellyfin/jellyfin:12.2`. No CVE yet. CWE-639 / CWE-862. 6.5 High (`CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N`). **Reach:** Remote. **Class:** IDOR. Any user who can see a public playlist.

GHSA-9×85 was Moderate: `GET /Playlists/{id}/Items?userId=` used the query `userId` as the calling user. 12.0 wrapped that in `RequestHelpers.GetUserId` and

filtered children with `IsVisible`. The sibling `GET /Playlists/{id}` still returns every linked child GUID. `CreatePlaylistDto.IsPublic` defaults true, so a JSON create is OpenAccess unless you remember to set the flag. OpenAccess playlists are visible to other users. `HasParentalAllowedChild` still lets a second account see the playlist even when tags would hide the movie. Then `GET /Videos/{id}/stream` has no `[Authorize]`. `StreamingHelpers` loads `GetItemById` without a user.

12.0 treated playlist-item disclosure as Moderate because GHSA-9×85 leaked metadata of a **private playlist**. This leftover yields the **media bytes** of items the caller is not allowed to list. That is the High.

02 Why Jellyfin

I had just finished a Signal-Server Dropwizard leftover and a Gitea current-user IDOR. Jellyfin is the self-hosted media server people actually run when they do not want Plex's account. Households share one box and split libraries: kids vs adults, roommates, a hidden folder the other profile is not assigned. The security model is folder ACL plus parental tags plus "this playlist is public." Those three sentences are how IDORs hide after a list-scoping fix.

I only wanted Critical or High. Remote preferred. Medium stays on the floor unless it chains. jellyfin-web 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, subject `[Jellyfin Security]`. Unpaid. No public GitHub issues on jellyfin/jellyfin.

03 Pinning 12.2

Latest stable on the day I cloned. Tag object **v12.2**, commit `ca0f16eb7c195f720a1493ed469a27c4657db0c6`. Image `jellyfin/jellyfin:12.2`. 12.0 was the September 2026 GHSA wave: VirtualFolders Path.Combine, SVG Skia,

playlist Items userId, HLS segment prefix, FFmpeg `-i`. I sat on 12.2 anyway. Patches that scope a list are a gift: you read the list handler, then you ask whether GET-by-id got the same check.

Twenty parallel source hunts. Four independent loopback labs. All four SUCCESS. Stacks torn down.

04 The skip list is half the hunt

Before you look for leftovers, you write down what is already public so you do not "discover" a CVE with a press release. On 12.2 the interesting ones were already closed:

- GHSA-6828-c7cx-hvqm VirtualFolders `Path.Combine` rooted second arg. Unauth during FirstTimeSetup. Critical. Closed.
- GHSA-cf3c-8m59-2vhx SVG Skia external refs. High. Direct `http` / `file` hrefs on the outer SVG are closed. The gzip `data:image/png` wrapper is a sibling leftover.
- GHSA-9×85-gx46-6522 playlist Items `userId` IDOR. Moderate. **This post is the leftover of that sanitizer**, not a re-file of the advisory.
- GHSA-hw73-62vp-9gxf HlsSegment path traversal. `ValidateTranscodePath` keeps legacy segments inside the transcode dir.
- GHSA-wwwm-px48-fpvq FFmpeg subtitle path argument injection. File-branch `-i` is escaped. Non-File `-i` is a sibling leftover.
- GHSA-4vx8-xhc9-qg6x session remote-control. Closed.
- GHSA-c39p-cgf8-866g plugin image traversal. Closed.

- GHSA-jg92-mrxq-vv75 / GHSA-9p5f-5×8v-x65m ClientLog path traversal. Closed.
- GHSA-f47c-m7gr-q92j MKV attachment filename. Closed.
- GHSA-jh22-fw8w-2v9x unauth StreamOptions ffmpeg injection. Closed.
- GHSA-v2jv-54xj-h76w SyncPlay group-name DoS. Closed.
- GHSA-8fw7-f233-ffr8 LiveTV M3U tuner SSRF. Closed.
- GHSA-j2hf-x4q5-47j3 subtitle upload `.strm` RCE. Closed.
- GHSA-rrr6-mvwg-9pg9 branding splash. Closed.
- GHSA-qcmf-gmhm-rfv9 X-Forwarded-For LAN policy. Closed.
- GHSA-2c3c-r7gp-q32m / GHSA-866x-wj5j-2vf4 FFmpeg codec-parameter injection. Closed.
- GHSA-vcmh-9wx9-rfqh SVG inline XSS. Attachment plus rasterize. Closed.
- GHSA-rr9h-w522-cvmr custom FFmpeg binary path. Admin. Closed.
- GHSA-rgjw-4fwc-9v96 unauth remote-image GET. Closed.
- GHSA-wg4c-c9g9-rxhx unauth file access. Closed.

I am not re-filing any of that.

05 Surfaces that returned nothing

I split the tree the way you split a media server: first-run wizard, path combine, session remote-control, plugins, LiveTV M3U, subtitle/`.strm`, QuickConnect, backup, DLNA, branding splash, SyncPlay, transcode/trickplay, WebSocket, system/config/encoder path, library download, ClientLog.

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

Wizard cannot be re-run unauth on a configured server. `FilePathJoinAbs` siblings stay under their roots. Session commands check the calling session. Plugin image names are sanitized. LiveTV M3U is gated. Subtitle upload does not write `.strm`. QuickConnect is a short-lived code, not a library dump. Backup zip is admin. DLNA is plugin-only on this pin. Splash is bounded. SyncPlay group names are length-checked. Transcode prefix is `ValidateTranscodePath`. WebSocket does not skip auth. Encoder path is admin. Library download is user-scoped. ClientLog is a stub that will not take `../`.

Dropped on purpose: `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. GHSA-rgjw closed the unauth siblings. I am not dressing that up as this bug.

The temperature changed on `GetPlaylist` sitting next to `GetPlaylistItems`.

06 The leftover

`GetPlaylistItems` is the GHSA-9×85 sanitizer. It still takes `userId`. It now forces that id through `GetUserId`. Then it filters children:

PLAIN TEXT Copy

```
var callingUserId = RequestHelpers.GetUserId(User, userId);
var playlist = _playlistManager.GetPlaylistForUser(playlistId, callingUserId);
// OpenAccess / owner / share ...
var user = _userManager.GetUserById(callingUserId);
var items = playlist.GetManageableItems().Where(i ⇒ i.Item2.IsVisible(user)).ToA
```

`GetPlaylist` is the sibling. Same controller. Same `GetPlaylistForUser`. No `Where`:

PLAIN TEXT Copy

```
[HttpGet("{playlistId}")]
public ActionResult<PlaylistDto> GetPlaylist(
    [FromRoute, Required] Guid playlistId)
{
    var userId = User.GetUserId();
    var playlist = _playlistManager.GetPlaylistForUser(playlistId, userId);
    if (playlist is null)
    {
        return NotFound("Playlist not found");
    }

    return new PlaylistDto()
    {
        Shares = playlist.Shares,
        OpenAccess = playlist.OpenAccess,
        ItemIds = playlist.GetManageableItems().Select(t ⇒ t.Item2.Id).ToList()
    };
}
```

`GetManageableItems` is `GetLinkedChildrenInfos()`. Every resolved linked child. No user. `GetPlaylistForUser` only requires the **playlist** to be visible.

Body create defaults the playlist **public**:

PLAIN TEXT Copy

```
public bool IsPublic { get; set; } = true;
```

JSON that omits `isPublic` is OpenAccess. Query-only create leaves `Public` null and `OpenAccess` false. The web UI posts JSON. Households that click Create get a public playlist.

OpenAccess `IsVisible` is `HasParentalAllowedChild`. That helper returns true when the user has no max parental rating. When it does walk children, it calls

`IsParentalAllowed(user, skipAllowedTagsCheck: true)`. Tags that hide a movie on `/Items` do not hide the playlist. The attacker still sees the object. The object still has the GUIDs.

Add is filtered. I am not inflating this into a write:

```
PLAIN TEXT
Copy
1  inputItems = inputItems.Where(i ⇒ i.IsVisible(user));
```

`POST /Playlists/{id}/Items` and create `Ids` go through that. Private GUIDs are dropped, not linked. InstantMix uses `GetItemById(id, user)`. Collections need `EnableCollectionManagement` (default false).

Then the stream. `VideosController.GetVideoStream` has no `[Authorize]`. No FallbackPolicy. DefaultPolicy only applies when the attribute is present. Media players historically put `ApiKey=` on the query string. A request with a known item GUID and no token still reaches `GetStreamingState`.

`StreamingHelpers` loads the item without a user:

```
PLAIN TEXT
Copy
1  var item = libraryManager.GetItemById<BaseItem>(streamingRequest.Id)
2      ?? throw new ResourceNotFoundException();
```

The user-aware overload that Items and InstantMix use is `GetItemById<T>(id, user)`, which calls `ItemIsVisible`. This call does not. `state.User` is set from the caller when a token is present, and is used later to force stream-copy when the user lacks transcode permission. It is not an ACL for the item. Audio `/Audio/{id}/stream` is the same helper. Authenticated HLS `master.m3u8` is the same lookup with class-level `[Authorize]` on `DynamicHlsController`.

Item GUIDs are `MD5(type.FullName + path)`, not random. The playlist leftover is the ID oracle. Guessing a default `/media/...` path is a second, lower-confidence

unauth path. It is not required for this High.

07 The lab

Source of truth is `lab/`. Image `jellyfin/jellyfin:12.2`. Port **18800**. Bind it to loopback.

`run.sh` tears the compose project down, seeds two tiny mp4s with `ffmpeg` (PrivateA filename is the witness, PublicB is a dummy), brings the image up, waits for `/Startup/User`, then `poc.py` runs the wizard.

The wizard creates `labadmin`, adds both folders, finishes setup, creates `labuser`. Policy: `EnableAllFolders=false`, `EnabledFolders=[PublicB]`, `AllowedTags=["lab-public"]`. Admin waits until the private movie is in the library, then `POST /Playlists` with `IsPublic: true` and that item id. `labuser` logs in.

PLAIN TEXT Copy

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

Target **only** `http://127.0.0.1:18800`. Do not publish the port off loopback. Witness-only. No shells.

Last lab run, trimmed:

PLAIN TEXT Copy

```
IOC playlist-items-b count=0 has-private=0
IOC playlist-get-b itemIds=['a2bf95146e44897be6fb09020a82eb52'] leak=1 openAccess
IOC user-b-library http=200 has-private=0
IOC unauth-stream static=true http=200 len=1142 witness-in-body=1
SUCCESS JELLYFIN-PLAYLIST-ITEMIDS items-filtered=yes itemids-leak=yes unauth-stre
```

B's `GET /Playlists/{id}/Items` is empty. Same id's `GET /Playlists/{id}` lists the private GUID. Unauth `GET /Videos/{id}/stream?static=true` is 200. The witness string is in the mp4 body because `run.sh` appends it after mux. That is the whole argument.

08 Wrong turns, in the order I made them

Treating `EnabledFolders` as the Items hide. Folder ACL alone did not hide the child on `/Items` in this lab. Movie `IsVisible` is parental/tags, not `EnabledFolders`. GHSA-9×85's Items filter is `Where(i ⇒ i.Item2.IsVisible(user))`. I had to set `AllowedTags=["lab-public"]` so the patched route would go empty. That is the control. `ItemIds` and unauth stream do not consult that filter. `HasParentalAllowedChild` on OpenAccess still lets B see the playlist because it skips the allowed-tag check.

Expecting add-to-playlist to be the High. It is closed. Create and `POST Items` run `IsVisible`. Planting a private GUID onto your own playlist fails. The leftover is read of the victim's public playlist, not write of yours.

Walking `GET /Playlists/{id}/Items?userId=` as the leak. That is GHSA-9×85. `GetUserId` 403s a foreign id. Wrong route. The leftover is the sibling GET with no `userId` at all.

Treating playlist-title enum as in bar. Titles and sharee GUIDs are Moderate. Streaming the private file is High. I kept that split.

HLS as the only sink. `master.m3u8` is authenticated and uses the same unscoped `GetItemById`. The progressive stream is worse: no `[Authorize]`. The lab hit `static=true` and got the file bytes. I did not need FFmpeg on the stream path.

Collections and InstantMix. Collections need a permission that defaults false. InstantMix is user-scoped. Neither is this leftover.

A reverse shell. Theatre. The oracle is a witness string in JSON and in the stream body.

09 What an attacker can do

Victim and attacker are two users on one server with disjoint libraries. Victim puts a private movie on a public playlist (default). Attacker has any account that can see that playlist. They do not need to be assigned the folder. They do not need a share.

Recon (lab-proven)

GET `/Playlists/{playlistId}`**.** Child GUIDs of a visible playlist. `ItemIds` is `GetManageableItems()` with no `IsVisible` on children. Also `Shares[].UserId` / `CanEdit`. Lab: B's Items list empty; same GET listed `a2bf95146e44897be6fb09020a82eb52`. Playlist id `e4798134497195ff4733dde62428e8d8`. `OpenAccess=true`. Witness name in the playlist JSON because the playlist was named after the witness.

Confidentiality (lab-proven)

GET `/Videos/{itemId}/stream?static=true`**.** Private media bytes. No `Authorization` header. Lab: HTTP 200, 1142 bytes, witness in the body.

GET `/Audio/{itemId}/stream`**.** Same helper, same unscoped lookup. Audio sibling. Same class.

Authenticated HLS `GET /Videos/{itemId}/master.m3u8` is the same `GetItemById` path. Lab exercised the unauth progressive stream, not HLS.

Integrity (closed)

POST `/Playlists/{id}/Items` **and create** `Ids`**.** `IsVisible` drops private GUIDs. You cannot add someone else's hidden movie to your playlist. Source, not a lab FAIL.

Adjacent surface that stays closed

- InstantMix (`GetItemById(id, user)`).
- Collections (`EnableCollectionManagement` default false; `FilterLinkedChildrenPerUser`).
- GHSA-hw73 transcode prefix.
- GHSA-9×85 `userId` impersonation on Items (`GetUserId` 403s).
- `GET /Playlists/{id}/Items` listing of private children (the patched route; lab `items-filtered=yes`).
- Password/policy write, plugin install, wizard re-run, ClientLog traversal.

No shell. No RCE. Confidentiality of media the caller is not assigned: High. That is why it is in bar.

PRIVILEGE	METHOD + PATH
Read child GUIDs (and sharee userIds) of a visible playlist	`GET /Playlists/{playlistId}`
Play those files with no token	`GET /Videos/{itemId}/stream` or `/Audio/{id}/stream`
HLS of the same GUID	`GET /Videos/{itemId}/master.m3u8`

PRIVILEGE	METHOD + PATH
Add private GUIDs to your own playlist	`POST /Playlists/{id}/Items`
Dump a private playlist via `?userId=`	`GET /Playlists/{id}/Items?userId=`
InstantMix / collections / transcode prefix	mix routes, BoxSet, HlsSegment

10 What this is not

It is not "GHSA-9×85 never shipped." Items is filtered. The playlist object is not. It is not unauthenticated recon: you need an account that can see the playlist. It is not a write. It is not RCE. It is not playlist-title enum.

Unauth stream-by-GUID is longstanding. This leftover is what enumerates the GUIDs for a restricted user. Leftover #1, Suggestions, enumerates them without a playlist. This one needs the public playlist. Both play the file the same way.

11 The fix

Filter `ItemIds` with `IsVisible(user)`, matching `GetPlaylistItems`. Same `Where`. Same user. Do not return a GUID the Items list would drop.

Scope stream `GetItemById` to a user (`GetItemById<T>(id, user)`), or put `[Authorize]` on `VideosController.GetVideoStream` and the audio sibling. Media players that relied on query-string `ApiKey=` can keep sending a token. A GUID is not a capability.

Default `IsPublic` to false if the product wants "public" to be a choice rather than a JSON default.

Re-run the loopback client against a patched build: B's `GET /Playlists/{id}` must not list the private GUID, and unauth stream of that GUID must not be 200 with the file.

Until then, do not treat "I scoped the Items list" as "I scoped the playlist." Public playlists plus unauth stream is a household library bypass.

I am not going to print a walk of `/Playlists/{id}` aimed at a live host. The missing `IsVisible` is the useful part. If you own the box, run the proof of concept against loopback.

Same product, other unpublished labs on this tag: Suggestions omit-userId · gzip SVG · FFmpeg colonslash.

12 References

- Proof of concept: abraxas/jellyfin-playlist-itemids · jellyfin-playlist-itemids-Abraxas-Labs.py
- Lab: `lab/` · docker-compose.yml · run.sh · poc.py
- @abraxas_null · github.com/abraxas · abraxaslabs.tech · abraxas.null@proton.me
- CWE-639 · CWE-862

- Tree: jellyfin v12.2 (`ca0f16eb`) · `GetPlaylist` · `StreamingHelpers`
- Nearby patched (not this GET): GHSA-9×85-gx46-6522
- Same pin: jellyfin-suggestions-idor · jellyfin-svg-gzip · jellyfin-ffmpeg-colonslash
- Series: Jellyfin 12.2 leftovers
- Product: Jellyfin

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, EncodingUtils -i leftover of GHSA-wwwm

NEXT →

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

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**

Home **Research** **Open Source** **Reading Room** **About** **GitHub** **@abraxas_null** **Console**

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

© 2026 Abraxas Labs