Incomplete fix: Directory traversal bypass via query string in ReceiveImage downloadURL parameters

Moderate **DanielnetoDotCom** published **GHSA-m63r-m9jh-3vc6** last week

Package

php **wwbn/avideo** (Composer)

Affected versions

<= 29.0

Patched versions

None

Description

Summary

The directory traversal fix introduced in commit `2375eb5` for `objects/aVideoEncoderReceiveImage.json.php` only checks the URL **path component** (via `parse_url($url, PHP_URL_PATH)`) for `..` sequences. However, the downstream function `try_get_contents_from_local()` in `objects/functionsFile.php` uses `explode('/videos/', $url)` on the **full URL string** including the query string. An attacker can place the `/videos/../../` traversal payload in the query string to bypass the security check and read arbitrary files from the server filesystem.

Details

The security fix at commit `2375eb5` added a traversal check at `objects/aVideoEncoderReceiveImage.json.php:49`:

```
$decodedPath = urldecode((string)(parse_url($_REQUEST[$value], PHP_URL_PATH) ?? ''
if (strpos($decodedPath, '..') !== false) {
```

```
    unset($_REQUEST[$value]);
}
```

This only inspects the **path component** of the URL. For a URL like `http://TARGET/x?a=/videos/../../etc/passwd`, `parse_url()` returns `/x` as the path — no `..` is found.

The URL then passes through `isValidURL()` (`objects/functions.php:4203`) which accepts it because `FILTER_VALIDATE_URL` considers `..` in query strings valid per RFC 3986.

It also passes `isSSRFSafeURL()` (`objects/functions.php:4264`) because the host matches `webSiteRootURL`, causing an early return at line 4294.

The URL reaches `url_get_contents()` (`objects/functions.php:1938`) which calls `try_get_contents_from_local()` (`objects/functionsFile.php:214`):

```
function try_get_contents_from_local($url)
{
    // ...
    $parts = explode('/videos/', $url);
    if (!empty($parts[1])) {
        // ...
        $tryFile = "{$global['systemRootPath']}{$encoder}videos/{$parts[1]}";
        if (file_exists($tryFile)) {
            return file_get_contents($tryFile);
        }
    }
    return false;
}
```

`explode('/videos/', $url)` operates on the **entire URL string** including the query string. For the malicious URL, `$parts[1]` becomes `../../../../../../etc/passwd`, constructing a path like `/var/www/html/AVideo/Encoder/videos/../../../../../../etc/passwd` which PHP's filesystem functions resolve to `/etc/passwd`.

The file content is returned to the caller and written to the video's thumbnail path via `_file_put_contents()`. All four `downloadURL_*` parameters (`downloadURL_image`, `downloadURL_gifimage`, `downloadURL_webpimage`, `downloadURL_spectrumimage`) are affected.

PoC

Prerequisites: An authenticated AVideo user account with upload permission and an existing video they own (with known `videos_id`).

1. Identify the AVideo instance's domain (e.g., `https://avideo.example.com`).

2. Send a POST request to the ReceiveImage endpoint with the traversal payload in the query string:

```
curl -s -b "PHPSESSID=<session_cookie>" \
  "https://avideo.example.com/objects/aVideoEncoderReceiveImage.json.php" \
  -d "videos_id=<YOUR_VIDEO_ID>" \
  -d "downloadURL_image=http://avideo.example.com/x?a=/videos/../../../../../../etc/pass
```

3. The response will include `jpgDestSize` indicating the file was read and written (confirming file existence and revealing file size).

4. For files that pass image validation (e.g., other users' uploaded images at known paths), the content persists at the video's thumbnail URL and can be retrieved:

```
curl -s "https://avideo.example.com/videos/<videoFileName>.jpg"
```

5. Non-image files (e.g., `/etc/passwd`, configuration files) are written but then deleted by `deleteInvalidImage()`. However, file existence and size are still leaked, and a race condition exists between the write and the deletion.

Impact

- **Arbitrary file read**: An authenticated user with upload permission can read any file on the server filesystem that the web server process has access to. Files that pass image validation (PNG/JPEG/GIF) are fully exfiltrable via the video thumbnail URL.
- **Information disclosure**: For non-image files, file existence and size are leaked through the `jpgDestSize` response field.
- **Configuration exposure**: Server configuration files, database credentials (`videos/configuration.php`), and other sensitive data can be targeted. While PHP files would be deleted by `deleteInvalidImage`, there is a race window between write and deletion.
- **Bypass of security fix**: This directly bypasses the path traversal mitigation added in commit `2375eb5`.

Recommended Fix

The `..` check in `aVideoEncoderReceiveImage.json.php` should inspect the **full URL** (after URL-decoding), not just the path component. Additionally, `try_get_contents_from_local()` should validate its derived path.

Fix 1 — Check the full URL in ReceiveImage (objects/aVideoEncoderReceiveImage.json.php:49):

```
// Replace:
$decodedPath = urldecode((string)(parse_url($_REQUEST[$value], PHP_URL_PATH) ?? ''));
if (strpos($decodedPath, '..') !== false) {

// With:
```

```
$decodedFull = urldecode((string)$_REQUEST[$value]);
if (strpos($decodedFull, '..') !== false) {
```

Fix 2 — Add path validation in try_get_contents_from_local (objects/functionsFile.php:229):

```
$tryFile = "{$global['systemRootPath']}{$encoder}videos/{$parts[1]}";
// Add traversal check:
$realTryFile = realpath($tryFile);
$videosDir = realpath("{$global['systemRootPath']}{$encoder}videos/");
if ($realTryFile === false || !str_starts_with($realTryFile, $videosDir)) {
    return false;
}
if (file_exists($tryFile)) {
    return file_get_contents($tryFile);
}
```

Severity

Moderate 6.5 / 10

CVSS v3 base metrics

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

Learn more about base metrics

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

CVE ID

CVE-2026-41062

Weaknesses

▶ CWE-22