Fiche pratique : Faille LFI/RFI

lundi 5 mai 2025 15:34

Un script PHP de gestion d'upload peut être vulnérable à une faille LFI (et plus rarement RFI) si tu autorises l'utilisateur à manipuler des **chemins de fichiers** après l'upload (ex : pour les lire, les afficher, les supprimer…).

La vulnérabilité ne se situe pas dans l'**upload lui-même**, mais **dans le traitement ultérieur du fichier** — par exemple, lorsque tu passes un nom de fichier dans l'URL pour afficher ou inclure ce fichier.

Voici donc un **TPFx / TPSFx** complet sur LFI/RFI **dans le contexte d'un upload de fichier**.

TPFx – Exploitation d'une faille LFI liée à l'upload de fichier

Contexte pédagogique

L'utilisateur peut téléverser une image, puis consulter son contenu via un script comme view.php?file=nomfichier.jpg. Le script lit ce fichier sans aucun contrôle, ce qui peut être exploité via une attaque LFI.

1. Mise en place du scénario vulnérable

Fichier 1 : upload.php (pour téléverser l'image)

```
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
   $uploadDir = 'Photo/';
   $uploadFile = $uploadDir . basename($_FILES['image']['name']);
   move_uploaded_file($_FILES['image']['tmp_name'], $uploadFile);
   echo "Image uploadée : " . $_FILES['image']['name'];
}
?>
<form method="post" enctype="multipart/form-data">
   <input type="file" name="image">
   <button type="submit">Envoyer</button>
</form>
```

Fichier 2 : view.php (vulnérable à LFI)

```
<?php
$file = $_GET['file']; // ⚠ pas de validation !
include("Photo/" . $file); // ⚠ inclusion directe
?>
```

2. Exploitation simulée (LFI)

Objectif : Lire un fichier système local

Appeller :
view.php?file=../../../../etc/passwd

- Si include() ou file_get_contents() est utilisé sans protection, le fichier est lu et s'affiche dans la

page.

Variante avancée : upload + injection

L'attaquant **upload une image piégée** contenant du PHP, puis appelle view.php?file=mainpage.jpg.

Si l'image est **interprétée comme code PHP** (ex: .php déguisé en .jpg et stocké sans filtrage), alors c'est une **RCE (Remote Code Execution)**, qui est plus grave.

TPSFx – Suppression de la faille (correction)

1. Sécurisation du script view.php

Version corrigée avec whitelist

```
<?php
// Liste blanche des fichiers autorisés
$allowedFiles = scandir('Photo/');
$file = basename($_GET['file']);
if (in_array($file, $allowedFiles)) {
   readfile("Photo/" . $file);
} else {
   echo "Fichier non autorisé.";
}
?>
```

2. Autres contre-mesures à enseigner

- **Ne jamais utiliser include() ou require() pour afficher une image**
- Préférer readfile() ou imagecreatefromjpeg() selon le type de fichier
- **Filtrer les extensions** et types MIME à l'upload (jpg, png, etc.)
- **Renommer les fichiers uploadés avec un UUID** ou hash serveur
- **Stocker les fichiers hors du dossier web** et servir via un contrôleur PHP

Conclusion

Cette fiche permet de faire le lien entre :

- la gestion basique d'un formulaire d'upload,
- les vulnérabilités de type LFI,
- l'importance de filtrer **à la fois à l'upload** et **à la lecture**.