Open Redirect Vulnerability

EXPLICATION

Nous avons relevé durant la phase d'énumération (voir ENUMERATION – partie 14) que les URLs permettant d'accéder aux réseaux sociaux étaient construites sur le modèle suivant :

```
http://192.168.1.37/index.php?page=redirect&site=facebook
```

Cela suggère que l'application opère une redirection basée sur le contenu du paramètre d'URL "**site**". Ce qui peut être dangereux si l'input utilisateur n'est pas contrôlé ou proprement régulé.

Imaginons que l'application implémente la redirection de cette façon dans sa fonction PHP :

```
$redirect_url = $_GET['site'];
header("Location: " . $redirect_url);
```

Imaginons maintenant que l'input utilisateur sur le paramètre **site** ne soit pas correctement contrôlé. Que se passerait-il si jamais j'entrais l'URL suivante :

```
http://192.168.1.37/index.php?page=redirect&site=http://192.168.1.5:8888
```

L'utilisateur cliquant sur une telle URL serait redirigée, par l'application web (tout à fait légitime), vers l'URL `http://192.168.1.5:8888` (qui s'avère en l'occurence être un serveur web que je contrôle).

Il s'agit d'une classe de vulnérabilités assez connue :
https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html

Ces cas de figure où l'on est capable de forcer l'application à rediriger un utilisateur vers l'URL de notre choix est plus dangereux qu'il n'y paraît. On peut utiliser ce mécanisme pour divers scénario, par exemple :

> Donner plus de crédibilité à un phishing, puisqu'une URL légitime est communiquée (et qui redirige ensuite vers notre phishing).
> Contourner des mécanismes de sécurité implémentés pour restreindre les accès à une application. Par exemple, une application pourrait n'autoriser l'accès à certaines de ses ressources que si l'entité qui essaie d'y accéder provient d'un certain domaine, ce qu'on pourrait simuler ici.

Bref, l'URL suivante illustrait ce concept :
```
http://192.168.1.37/index.php?page=redirect&site=http://192.168.1.5:8888
```

Et nous dévoile donc un **flag**.

RESSOURCES

Un script python permet d'automatiser la démonstration en envoyant une simple requête avec cette URL, et en affichant la page contenant le flag.

MITIGATION

Les recommandations de Portswigger sont assez solides.

> Dans le cas où on peut éviter le user-input libre :

- **Remove** the redirection function from the application, and replace links to it with direct links to the relevant target URLs.
- Maintain a server-side list of all URLs that are permitted for redirection. Instead of passing the target URL as a parameter to the redirector, pass an **index** into this list.

> Dans le cas où on ne peut éviter de fournir de l'input contrôlé par l'utilisateur à la fonction de redirection :

- The application should use **relative** URLs in all of its redirects, and the redirection function should strictly validate that the URL received is a relative URL.
- The application should use URLs relative to the web root for all of its redirects, and the redirection function should **validate** that the URL received starts with a slash character. It should then prepend http://yourdomainname.com to the URL before issuing the redirect.
- The application should use absolute URLs for all of its redirects, and the redirection function should verify that the user-supplied URL begins with http://yourdomainname.com/ before issuing the redirect.