Module Sécurité
TP5 : Attaque ARP

François Lesueur

Lors de ce TP, vous aurez besoin de plusieurs machines pour réaliser l'attaque. Regroupez-vous donc en trinômes.

1 Introduction

Durant ce TP, nous allons mettre en place un serveur HTTPS, attaquer la connexion d'un client vers ce serveur puis voir comment protéger le client de ces attaques. Le TP est proposé dans le cas du protocole HTTPS, c'est-à-dire HTTP encapsulé dans une couche SSL, mais les problèmes soulevés s'appliquent à toute connexion SSL.

Pour ce TP, vous aurez besoin de 3 machines :

- une machine honnête serveur nommée *serveur*. Cette machine sert un site web HTTPS en utilisant le serveur Apache httpd ;
- une machine client nommée *client*. Cette machine accède au site web HTTPS ;
- une machine attaquante nommée *attaquant*. Cette machine va se faire passer pour le serveur honnête.

2 Mise en place du service honnête

Dans un premier temps, vous devez installer et configurer un serveur web Apache avec le module SSL. Il faut installer `httpd` et `mod_ssl`. La configuration d'Apache est localisée dans `/etc/httpd` et l'installation du paquet `mod_ssl` active le SSL avec un certificat auto-signé généré à l'installation. Lancez le service et vérifiez son fonctionnement depuis la machine cliente.

Créez un fichier `index.html` dans le dossier `/var/www/html` de la machine serveur ; ce fichier sera envoyé par le serveur honnête lors de la visite du site et doit donc contenir un texte permettant au client de constater qu'il est bien connecté à ce serveur.

3 Attaque

Nous allons réaliser une attaque de type *ARP spoofing*. Sur un réseau local Ethernet, les messages IP sont encapsulés dans des trames Ethernet. Lorsqu'une machine A envoie un message vers une adresse IP donnée, ce message est encapsulé dans une trame Ethernet et la machine A doit obtenir l'adresse MAC de la machine destination : le protocole ARP permet cette résolution d'adresse.

L'attaque *ARP spoofing* permet de transmettre de faux messages ARP à la cible afin de lier une adresse IP légitime du réseau local à l'adresse MAC de l'attaquant : bien qu'adressé à la bonne adresse IP, ce message sera reçu par l'attaquant qui possède l'adresse MAC associée. Un attaquant peut ainsi se faire passer pour un serveur local (cas que nous utilisons dans ce TP), pour la passerelle de réseau ou encore pour un serveur DNS.

3.1 Corruption de la table ARP de la victime

Vous devez installer le programme `arpspoof` contenu dans le paquet `dsniff`. Ce programme est lancé de la manière suivante (en root) :

```
# arpspoof -t target host
```

où :

- *target* est la machine client que nous souhaitons attaquer
- *host* est la machine serveur pour laquelle nous souhaitons nous faire passer

Ce programme est actif jusqu'à sa fermeture. Attaquez le client en prenant l'adresse IP du serveur honnête.

3.2 Réception des messages adressés au serveur honnête

À ce moment-là, le client ne devrait plus pouvoir se connecter au serveur honnête. Nous allons configurer le serveur attaquant pour recevoir les messages du client et y répondre.

Dans un premier temps, le serveur attaquant doit accepter les messages reçus à la place du serveur honnête. Par défaut, les messages sont reçus au niveau Ethernet mais sont refusés par la couche IP de l'attaquant : en effet, cette couche refuse les messages dont elle n'est pas la destination. Il faut configurer une interface virtuelle pour recevoir ces messages :

```
# ifconfig eth0:0 <IP usurpée>
```

Dans un second temps, il faut configurer un serveur HTTPS sur le serveur attaquant, avec un fichier index.html différent du serveur honnête afin de bien distinguer les deux cas.

Connectez-vous enfin avec le client sur la machine serveur honnête, en `https` : quel serveur répond ? Que pouvez-vous déduire sur la sécurité du protocole HTTPS et sur les exceptions de sécurité des navigateurs ?

4 Sécurisation de la connexion HTTPS

Dans la première partie de ce TP, nous avons utilisé des certificats auto-signés, c'est-à-dire non validés par une autorité de certification reconnue. Dans ce cas, le client ne peut pas vérifier le certificat du serveur et il est possible pour un attaquant de se faire passer pour le serveur, bien que le protocole HTTPS en lui-même soit sûr.

Les autorités de certification permettent de résoudre ce problème, en n'autorisant que le possesseur du domaine `acme.fr` à présenter un certificat pour la machine `www.acme.fr`. Nous allons instantier une telle autorité, la faire reconnaître par la machine client et signer le certificat du serveur avec la clé de cette autorité.

4.1 Création de l'autorité de certification

Toutes les opérations suivantes peuvent être réalisées à l'aide de la commande `openssl`. Néanmoins, pour des raisons pratiques, nous allons utiliser une interface graphique, `tinyca2`.

Installez ce programme et exécutez-le. Laissez-vous guider pour créer une nouvelle autorité de certification puis, dans la fenêtre principale, exportez le certificat de l'autorité au format `pem`.

4.2 Reconnaissance de cette autorité par le client

Copiez le certificat sur la machine client. Dans Firefox, allez dans Édition / Préférences / Avancé / Chiffrement / Afficher les certificats. Importez le certificat précédemment copié.

À partir de ce moment, Firefox reconnaîtra les certificats signés par cette autorité sans avoir à ajouter une exception de sécurité.

4.3 Création du certificat pour le serveur honnête

Dans `tinyca2`, créez un certificat signé par l'autorité. Dans le champ CN (Common Name) du certificat généré, vous devez entrer le nom exact de la machine serveur, tel qu'il sera tapé dans le navigateur (si le client accède à `https://www.acme.fr`, le champ CN du certificat présenté doit valoir `www.acme.fr`).

Exportez la clé du certificat, copiez-la sur le serveur honnête et mettez à jour la configuration d'Apache pour utiliser ce nouveau certificat. Connectez-vous avec le client et constatez que le certificat est accepté automatiquement (pensez bien à stopper l'attaque ARP et à éventuellement vider la table ARP corrompue du client avec la commande `arp`).

5 Bilan

Dans ce TP, nous avons constaté que le protocole HTTPS (SSL) n'était sûr que si le certificat obtenu lors de la connexion était bien le certificat du serveur honnête. Plus généralement, nous avons été confronté au problème de la distribution des clés : les protocoles cryptographiques ne sont sûrs que si le client a su obtenir la clé du serveur de manière fiable.

Les autorités de certification sont une solution à ce problème. Le client doit reconnaître un faible nombre d'autorités (un certain nombre est présent par défaut dans les navigateurs web) et ces autorités délivrent des certificats aux serveurs.

Dans le cadre d'un déploiement de services SSL en entreprise (serveurs web, LDAP, VPN...), l'entreprise devrait soit obtenir des certificats d'une autorité reconnue par défaut, soit créer sa propre autorité. Dans le second cas, le certificat de l'autorité locale peut par exemple être installé lors de la configuration initiale des postes de travail.

Retenez bien qu'il existe un risque réel de sécurité lors de l'acceptation de certificats auto-signés, non reconnus automatiquement par le navigateur web.

6 Bonus

La même attaque peut-être réalisée par l'un des routeurs présents entre le client et le serveur, sans que ni le client ni le serveur ne s'en aperçoivent : on parle alors d'attaque *Man-in-the-Middle*. Comment cette attaque est-elle mise en œuvre précisément ?