Petite introduction pratique à OpenSSL

On donne un serveur Ubuntu 20.04 (VPN/IP campus seulement) `192.168.74.36` appelé par la suite *le serveur*. On donne également une autre machine similaire `192.168.74.58` appelé par la suite *le serveur annexe* si votre machine personnelle n'est pas sous Linux.

Attention quand vous cherchez la documentation de `openssl` ou `john` sur Google : de nombreuses documentation sont périmées (les options ont changées) *prenez toujours les références et faites attention à la version considérée !*.

- https://www.openssl.org/docs/man1.1.1/
- https://www.openwall.com/john/doc/
- https://www.feistyduck.com/library/openssl-cookbook/online/
- https://pki-tutorial.readthedocs.io/en/latest/#

Génération de clef publique pour authentification SSH

Ce petit exercice fait utiliser un peu de cryptographie et a aussi pour objectif de s'assurer que tous peuvent utiliser l'environnement https://cloud-info.univ-lyon1.fr (VM avec IPs privées):

- Identifiez vous sur le serveur avec vos identifiants UCBL
- Avec `ssh-keygen` générer une clef RSA 2048 bits *sans mot de passe de protection* et *précisez le format PEM* (`-m PEM`).
- Déposer la clef publique avec `ssh-copy-id` sur le serveur
- Authentifiez vous sur le serveur sans mot de passe (`-i clef`)

Cette procédure est très pratique et très commune. **Conseil** : mettez systématiquement un commentaire avec `-C 'comment'` quand vous générez les clefs.

- Maintenant, répéter la procédure précédente (générer et déposer) mais lors de la création de la clef **donnez un mot de passe de protection**. Utilisez un mauvais mot de passe (pour un exercice plus tard)
- Logguez-vous avec cette nouvelle clef. Le serveur a-t-il besoin de connaitre le mot de passe ?
- Que contient maintenant `~/.ssh/authorized_keys` sur le serveur ?

A chaque fois que vous allez vouloir vous connecter, vous devrez taper le mot de passe. Ceci rendra impossible certaines automatisations.

- Vérifiez que `ssh-agent` a bien un *pid*.
- Avec `ssh-add` ajoutez la deuxième clef à l'agent.
- Authentifiez-vous maintenant sur le serveur sans mot de passe.
- Avec `ssh-add -l`, vérifiez que la clef est présente dans la liste et notez l'empreinte SHA256.
- Avec `ssh-keygen -lf clef`, vérifiez l'empreinte précédente.

Attaque au dictionnaire

On vous donne le fichier `sample-1.passwd` qui contient la chaîne suivante (sans retour chariot ni indentation):

```
$6$bTJI.5QPcTHHY$DMZ.pUFM7OPtQ2/fy0pYwHTHpRlMfV8ROF2
  iYiyKdlGH5wk2J8CqjhDDs0OYQIlnUpkDiC3MimkfNhDVZgeaa.
```

- Quelle est le codage utilisé ? Quel est le sel ?
- Avec https://www.openwall.com/john/ (installé sur le serveur via snap) tentez de retrouve le mot de passe utilisé `sample.passwd`. Allez-vous y arriver ?
- Que contient ce fichier https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt (installé dans `/usr/share/wordlist`) ? Combien-at-il de lignes ?
- Utilisez le fichier précédent pour aider `john`. Attention, à cause de l'isolation de snap, vous ne pourrez pas utiliser l'argument `--wordlist=`, utilisez à la place `cat /usr/share/wordlist/rockyou.txt | john --stdin`. Quel est le mot de passe ?
- Maintenant que vous connaissez le secret, retrouvez la chaîne produite avec la commande `openssl passwd`.
- Si on vous avez dit qu'un mot de passe fait 6 chiffres, estimez le temps pour retrouver le mot de passe en *mode incremental*. Testez sur le fichier `sample-2.passwd` qui contient la chaine ci-après.
- Vue la forme du mot de passe, que pourriez vous conclure ?
- Vous avez utilisé un mot de passe faible lors de la généreration de vos clefs RSA, vous pouvez les convertir avec `/snap/john-the-ripper/current/run/ssh2john.py`. Tenter de retrouver votre mot de passe.

```
$6$TdM/jAOFtjv.F8$NDBFnbU7u4aXcw7xuE5XlqQqInIDhS7eTR
  /PF.l3O4bTmTm4YyLrEvVjJMs.m6OjNBYNfdrJvRsTbW/XORf3k0
```

Intro à HTTPS/TLS

- Visitez les services HTTPS des deux serveurs. Quel sont les messages d'erreur et les problèmes relevés ?
- Pour **chacun des deux serveurs**, utilisez les options avancées de votre navigateur et déterminer :
  - qui a émit le certificat (*issuer*)
  - qui est le bénéficiaire du certificat (*subject*)
  - la clef publique et son type (RSA ou autre)
  - Regardez la configuration de nginx dans `/etc/nginx/` et retrouvez le certificat servi par le serveur. Regardez son contenu avec `openssl x509 -text -noout -in le_fichier_trouve`
- Dans le gestionnaire certificats de votre navigateur ajoutez le certificat MIF13 à la liste des autorités. Expliquez pourquoi cela résoud le problème d'un des deux sites.