S. Lavirotte
Polytech'Nice – Sophia
PeiP 1
2017-2018

TD séance n° 15
Internet et Sécurité

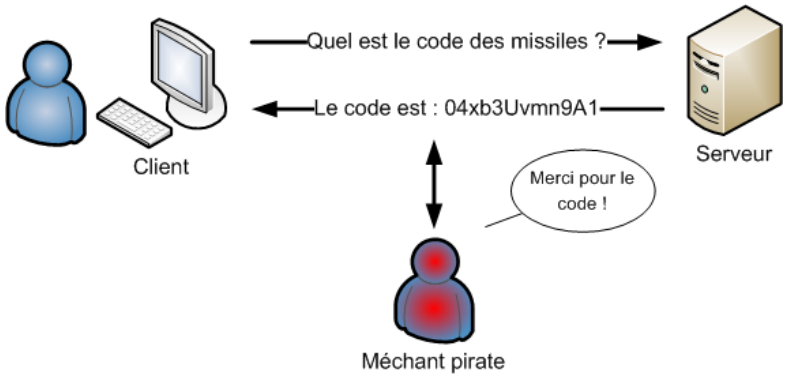
1 Internet et Sécurité : introduction à la cryptographie

Jusqu'ici, vous avez utilisé votre ordinateur avec Linux ou Windows d'une manière classique : vous étiez en face de votre ordinateur (même dans le cas de l'utilisation d'une machine virtuelle). Vous étiez physiquement à côté de votre machine de travail, vous avez appuyé sur le bouton « Power » pour l'allumer et visualiser les données de la machine sur l'écran connecté à la machine. Jusque-là, rien de nouveau.

Pourtant grâce à Internet, vous pouvez communiquer avec de nombreuses machines à distance, contacter des serveurs à l'autre bout de la planète (nous avons vu cela la semaine dernière, y compris en accédant à ces données de manière sécurisée), etc. Alors pourquoi ne pas aussi travailler sur une machine à distance, sans que celle-ci soit physiquement devant vous ? Par exemple, depuis la maison, vous pourriez accéder aux serveurs de Polytech Nice Sophia et ainsi accéder à vos documents comme si vous étiez à l'école.

Mais un serveur, au fait, qu'est-ce que c'est ? Un **serveur** est un ordinateur qui reste allumé 24h/24, 7j/7. Cet ordinateur est semblable au vôtre (quoique souvent plus puissant et parfois plus bruyant) : il possède un processeur, de la mémoire, un ou plusieurs disques durs, bien sûr une connexion réseau, etc. Le principe d'un serveur est de rester allumé et connecté à Internet tout le temps. Il (vous) offre des services. La machine qui se connecte au **serveur** est appelé le **client** (voir figure ci-dessous).

Mais nous avons vu précédemment que les messages échangés entre les machines sont en clair par défaut. Donc quand le client communique avec le serveur, on peut a priori écouter toutes les informations qui sont échangées.



Ça ne vous dérange pas que l'on vous espionne ? Soit. Mais quand vous allez vous connecter à un serveur, vous allez donner votre identifiant et votre mot de passe. Si quelqu'un les intercepte, il pourra avoir accès à votre compte et à toutes les informations qu'il contient, sans oublier qu'il pourra donc se faire passer pour vous. Il ne faut donc pas que l'identifiant et le mot de passe apparaissent en clair sur le réseau ! Il est impératif que les données soient cryptées. Vous ne voulez pas que quelqu'un récupère votre mot de passe tout de même !

Dans les exercices de ce TD, vous pourrez constater que, par défaut, l'échange des informations est bien réalisé en clair et que l'on peut tout lire si l'on sait écouter. Comme on ne peut pas complètement empêcher quelqu'un d'intercepter les données qui transitent sur Internet, il faut trouver un moyen pour que le client et le serveur communiquent de manière sécurisée. Le cryptage sert précisément à ça : si le pirate récupère le message crypté, il ne peut rien en faire. Nous avons évoqué la semaine dernière le protocole SSL qui permet de sécuriser les échanges

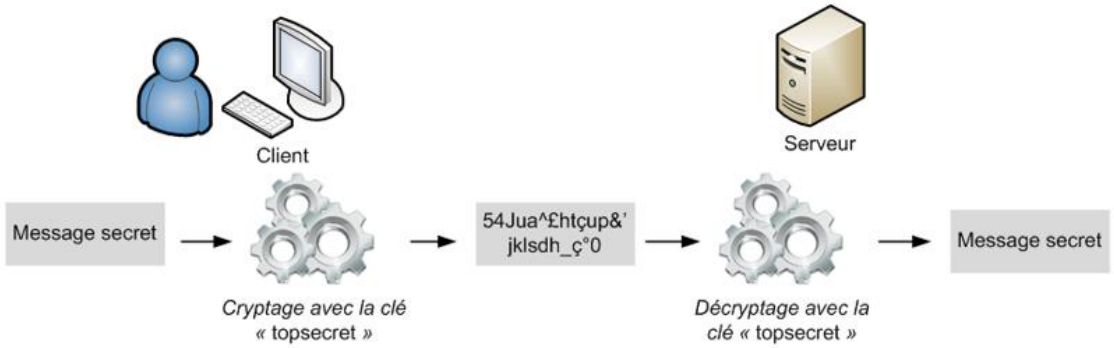
entre le navigateur Web (client) et le serveur Web et ainsi éviter de se faire voler ses informations de carte bancaire par exemple.

Pour comprendre comment cela fonctionne en détail, nous allons tout d'abord nous intéresser aux méthodes de cryptage des communications pour pouvoir échanger des informations en toute confidentialité. La **confidentialité** a été définie par l'Organisation Internationale de Normalisation (ISO) comme : « *le fait de s'assurer que l'information n'est seulement accessible qu'à ceux dont l'accès est autorisé* ».

1.1 Méthodes de cryptage de l'information

1.1.1 Le cryptage symétrique

C'est la méthode de cryptage la plus simple. Cela ne veut pas dire qu'elle n'est pas robuste (il existe des cryptages symétriques très sûrs). Cela veut plutôt dire que le fonctionnement est simple à comprendre. Avec cette méthode, on utilise une clé (un mot de passe secret) pour crypter un message. Par exemple, imaginons que cette clé soit « *topsecret* ». On encrypte alors le message que l'on souhaite envoyer avec la clé, le message encrypté sera celui communiqué sur Internet et à la réception, le message encrypté sera décrypté grâce à la même clé « *topsecret* ». Si un pirate intercepte un message crypté, il ne peut rien en faire s'il ne connaît pas la clé secrète.



Mais il faut que la personne qui crypte et celle qui décrypte connaissent toutes deux cette clé qui sert à crypter et décrypter. Ils doivent donc s'échanger la clé secrète « *topsecret* » ; pour cela, le client doit d'abord envoyer la clé secrète pour que le serveur puisse décrypter ses futurs messages (ou l'inverse). Mais lorsque le client envoie la clé secrète, si un pirate l'intercepte, il pourra alors très facilement décrypter tous les messages cryptés qui seront échangés entre le client et le serveur. À moins de… crypter la clé de cryptage lors de son envoi ! Et pourquoi pas ?

Pour crypter la clé de cryptage symétrique, on doit utiliser une autre méthode : le cryptage asymétrique. Avec cette autre méthode, on ne risque pas de connaître à nouveau le problème que l'on vient de rencontrer.

1.1.2 Le cryptage asymétrique

Le cryptage asymétrique consiste à utiliser deux clés au lieu d'une seule :

- Une clé dite **publique**
- Une clé dite **privée**

Dans le cas du cryptage de l'information, la clé publique sert à crypter le message et la clé privée sert à décrypter celui-ci. Comme son nom l'indique, la clé publique est publique, c'est-à-dire qu'elle peut être largement diffusée, voire être mise à disposition sur un serveur de clés. Par opposition, la clé privée constitue un bien personnel et ne doit jamais être diffusée ou divulguée à qui que ce soit.

On pourra faire deux choses avec ces clés : *chiffrer* un contenu (l'encrypter à l'aide de la clé publique) afin d'assurer la **confidentialité** des échanges ou bien *signer* un message (calculer une valeur unique reposant sur le contenu du

message échangé et sur votre identité grâce à la clé privée. Cette valeur servira de signature, comme un cachet spécifique) afin d'assurer l'**authenticité** (que vous soyez bien l'expéditeur) et l'**intégrité** (que le message n'a pas été modifié après la signature).

1.2 Algorithmes de chiffrement basés sur des clés asymétriques

Il existe de nombreux algorithmes basés sur des clés asymétriques (RSA, DSA, ECDSA, Ed25519), mais les deux dominants, et de loin, sont RSA (nommé par les initiales de ses trois inventeurs Ronald **R**ivest, Adi **S**hamir et Leonard **A**dleman) et DSA (**D**igital **S**ignature **A**lgorithm). Nous allons regarder plus en détail l'algorithme RSA.

1.2.1 Chiffrement RSA : principe d'utilisation

Le chiffrement RSA est asymétrique : il utilise donc, comme nous l'avons vu précédemment, une paire de clés (des nombres entiers) composée d'une clé publique pour chiffrer et d'une clé privée pour déchiffrer des données confidentielles. Son fonctionnement général est donc basé sur le mécanisme du chiffrement asymétrique que nous venons de présenter. Voici un exemple concret d'utilisation.

Soient deux clés créées par une personne, souvent nommée Alice par convention, qui souhaite que lui soient envoyées des données confidentielles. Alice rend la clé publique accessible. Cette clé est utilisée par ses correspondants (Bob par exemple.) pour chiffrer les données qui lui sont envoyées. La clé privée est quant à elle réservée à Alice, et lui permet de déchiffrer ces données. La clé privée peut aussi être utilisée par Alice pour signer une donnée qu'elle envoie, sa clé publique permettant à n'importe lequel de ses correspondants de vérifier la signature. Pas mal, non !

1.2.2 Fonctionnement « détaillé » de l'algorithme RSA

Une condition indispensable pour que le mécanisme soit efficace est qu'il soit « calculatoirement impossible » de déchiffrer à l'aide de la seule clé publique, ou plus précisément qu'il soit impossible de reconstituer la clé privée à partir de la clé publique. Pour cela l'algorithme RSA utilise les congruences sur les entiers et le petit théorème de Fermat, pour obtenir des fonctions à sens unique (non inversibles). Voici une présentation générale du fonctionnement.

Tous les calculs se font modulo un nombre entier n qui est le produit de deux nombres premiers (p et q). Les messages en clair et chiffrés sont des entiers inférieurs à l'entier n (tout message peut être codé (représenté) par un entier). Les opérations de chiffrement et de déchiffrement consistent à élever le message à une certaine puissance modulo n (c'est l'opération d'exponentiation modulaire).

La seule description des principes mathématiques sur lesquels repose l'algorithme RSA n'est pas suffisante. Sa mise en œuvre concrète demande de tenir compte d'autres questions qui sont essentielles pour la sécurité. Par exemple le couple (clé privée, clé publique) doit être engendré par un procédé vraiment aléatoire qui, même s'il est connu, ne permet pas de reconstituer la clé privée. Les données chiffrées ne doivent pas être trop courtes, pour que le déchiffrement demande vraiment un calcul modulaire. C'est le processus de création des clés qui permet d'assurer ces propriétés.

1.2.3 Création des clés

Dans l'exemple précédant et suivant, Alice est en général le serveur et Bob, le client (donc vous). L'étape de création des clés est à la charge d'Alice (donc du serveur). Elle n'intervient pas à chaque chiffrement de message car les clés peuvent être réutilisées. La difficulté première, que ne règle pas le chiffrement, est que Bob (donc vous) soit bien certain que la clé publique qu'il détient est celle d'Alice. Le renouvellement des clés n'intervient que si la clé privée est compromise, ou par précaution au bout d'un certain temps (qui peut se compter en années).
Voici l'algorithme de création des clés RSA :

1. Choisir p et q, deux nombres premiers distincts ;
2. Calculer leur produit $n = pq$, appelé module de chiffrement ;
3. Calculer $\phi(n) = (p - 1)(q - 1)$ (c'est la valeur de l'indicatrice d'Euler en n) ;
4. Choisir un entier naturel e premier avec $\phi(n)$ et strictement inférieur à $\phi(n)$, appelé exposant de chiffrement ;
5. Calculer l'entier naturel d, inverse de e modulo $\phi(n)$, et strictement inférieur à $\phi(n)$, appelé exposant de déchiffrement ; d peut se calculer efficacement par l'algorithme d'Euclide étendu.

Comme e est premier avec $\phi(n)$, d'après le théorème de Bachet-Bézout il existe deux entiers d et k tels que $ed + k\phi(n) = 1$, c'est-à-dire que $ed \equiv 1 \pmod{\phi(n)}$: e est bien inversible modulo $\phi(n)$.

Le couple (n, e) est la clé publique du chiffrement, alors que le nombre d est sa clé privée, sachant que l'opération de déchiffrement ne demande que la clef privée d et l'entier n, connu par la clé publique (la clé privée est parfois aussi définie comme le triplet (p, q, d)).

1.2.4 Chiffrement/Déchiffrement des messages

Si M est un entier naturel strictement inférieur à n représentant un message, alors le message chiffré sera représenté par : $C \equiv M^e \pmod{n}$, l'entier naturel C étant choisi strictement inférieur à n.

Pour déchiffrer C, on utilise d, l'inverse de e modulo $(p-1)(q-1)$, et on retrouve le message clair M par $M \equiv C^d \pmod{n}$

Et après tout cela, si on passait à une partie plus pratique ! Comme nous avons un algorithme qui spécifie le fonctionnement de RSA, c'est un programme qui va vous générer cette paire de clé publique/privée qui fonctionne sur les principes décrits ci-dessus.

2 Protocole SSH (Secure SHell) : la solution pour sécuriser en pratique

SSH est un protocole plus complet que SSL, que nous avons rapidement évoqué la semaine dernière, mais qui a le même but : sécuriser les communications entre une machine A et une machine B.

2.1 Etablissement d'une connexion SSH par l'exemple

SSH est un protocole assez complexe, mais maintenant que nous avons vu tous les éléments nécessaires à sa mise en œuvre, cela devrait être plus simple pour vous. Et il est vraiment intéressant de savoir comment il fonctionne car c'est le système actuellement utilisé. Plutôt que de l'utiliser bêtement, nous allons étudier dans les grandes lignes son mode de fonctionnement.

SSH utilise les deux cryptages : asymétrique et symétrique. Cela fonctionne dans cet ordre :

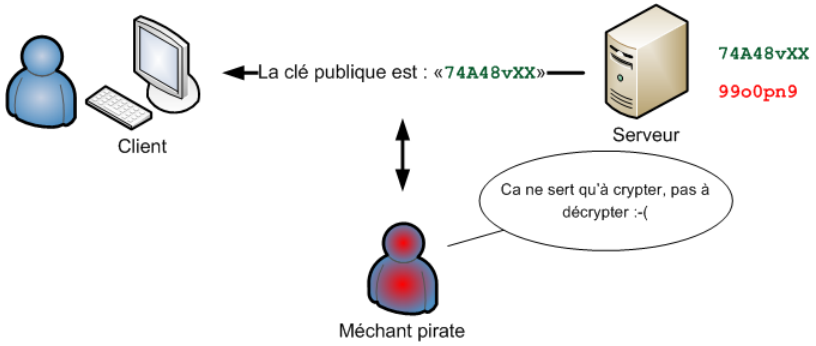
1. On utilise d'abord le cryptage asymétrique pour s'échanger une clé secrète de cryptage symétrique.
2. Ensuite, on utilise tout le temps la clé de cryptage symétrique pour crypter les échanges.

Pourquoi ne pas utiliser uniquement du cryptage asymétrique tout le temps me direz-vous ? Ce serait possible mais il y a un défaut : le cryptage asymétrique demande beaucoup trop de ressources de calcul. Le cryptage asymétrique est `100` à `1000` fois plus lent que le cryptage symétrique ! Le cryptage asymétrique est donc utilisé seulement au début de la communication, afin que les ordinateurs s'échangent la clé de cryptage symétrique de manière sécurisée. Ensuite, ils ne communiquent que par cryptage symétrique qui est plus rapide.

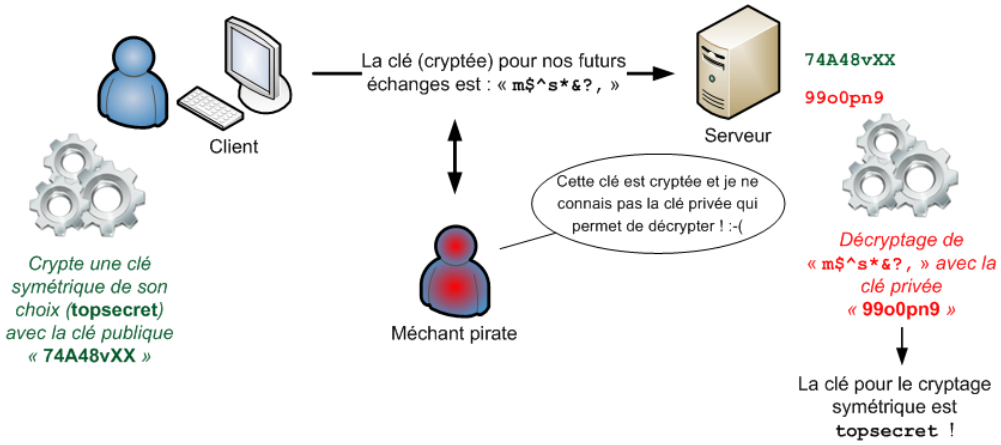
Voici les étapes de l'établissement d'une connexion SSH :

TD séance n° 15
Internet et Sécurité

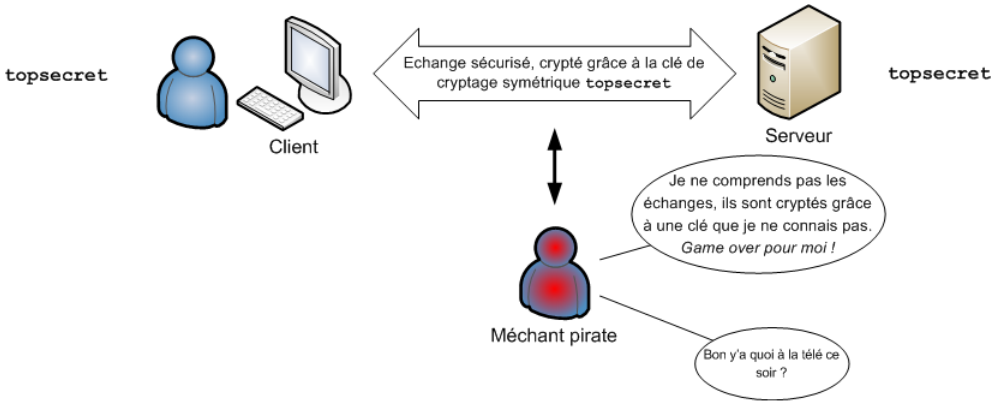
– Le serveur envoie sa clef publique au client. Celui-ci vérifie qu'il s'agit bien de la clef du serveur, s'il l'a déjà reçue lors d'une connexion précédente et vous informe d'un problème dans le cas contraire.



– Le client génère une clef de cryptage symétrique (« topsecret » dans notre exemple) et l'envoie au serveur, en chiffrant l'échange avec la clef publique du serveur (chiffrement asymétrique). Le serveur déchiffre cette clef secrète en utilisant sa clé privée, ce qui prouve qu'il est bien le vrai serveur.



– Pour le prouver au client, il chiffre un message standard (Cf. RFC4256) avec la clef secrète et l'envoie au client. Si le client retrouve le message standard en utilisant la clef secrète, il a la preuve que le serveur est bien le vrai serveur.
– Une fois la clef secrète échangée, le client et le serveur peuvent alors établir un canal sécurisé grâce à la clef secrète commune (chiffrement symétrique).



Il est maintenant temps de voir quels outils sont nécessaires pour mettre en place et utiliser `ssh`.

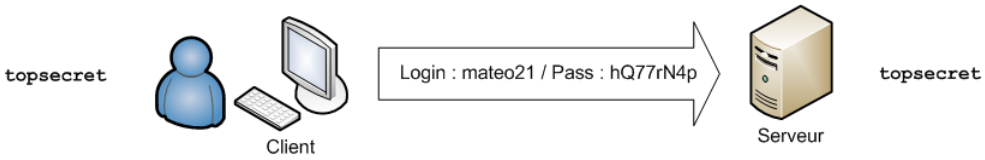
TD séance n° 15
Internet et Sécurité

2.2 Authentification distante à l'aide de SSH

La connexion via SSH est donc réalisée suivant le principe décrit précédemment. Deux types d'authentifications sont alors possible. L'authentification dite simple et une authentification par clés.

2.2.1 Authentification simple

Une fois que le canal sécurisé est en place (comme décrit dans la section précédente), le client s'identifie de la même manière que lorsque vous vous connectez sur la machine physique, en tapant votre identifiant et votre mot de passe, mais ces échanges d'informations se font de manière cryptée.



- Le client envoie un nom d'utilisateur et un mot de passe lors de l'établissement de la connexion ;
- Le serveur vérifie si l'utilisateur en question peut obtenir un accès à la machine, donc si le mot de passe fourni est valide.
- Si les vérifications se sont passées sans problèmes, l'authentification est alors réussie. Le canal sécurisé reste en place jusqu'à ce que l'utilisateur se déconnecte.
- Sinon, l'accès au serveur est refusé et la connexion est immédiatement interrompue.

La commande qui permet cela est la commande `ssh` en spécifiant le nom de l'utilisateur et la machine sur laquelle on souhaite se connecter : `ssh user@host.domain.fr` ou `ssh user@adresse_ip`.

Le protocole SSH vous permet ainsi de vous connecter à un ordinateur distant de manière sûre et d'avoir accès à un interpréteur de commandes qui va vous permettre d'exécuter des commandes sur la machine sur laquelle vous vous êtes connectés. Et nous retrouvons ici l'intérêt d'avoir appris à utiliser les lignes de commandes au début du cours ! La boucle est bouclée (qu'est-ce qu'il est bien fait ce cours ☺ !)

2.2.2 Authentification par clés

Une autre méthode existe pour s'authentifier ; elle s'appuie sur les clés du client. Pour rappel, ces clés ont été générées avant la connexion (on parle d'une paire de clés, comprenant une clé publique et une clé privée) :

- Le serveur crée un « challenge » avec la clé publique du client pour le tester.
- Le client reçoit ce challenge. Son but : le déchiffrer avec sa clé privée (client).
- Le client renvoie sa réponse lorsqu'il a réussi à déchiffrer le challenge.
- Le serveur autorise alors l'accès au client si et seulement si la réponse reçue est correcte.

Il reste alors à voir comment créer une paire de clés publique/privée pour réaliser ce type de connexion (cette opération est effectuée sur le serveur quand vous l'installez, mais là, nous allons générer des clés pour le client, donc vous).

2.3 Création des clés RSA ou DSA pour SSH

Pour réaliser une authentification par clé, il est nécessaire de créer ses propres clés. Et celles-ci doivent respecter des caractéristiques importantes pour garantir la sécurité.

2.3.1 Taille de clé

Une clé a une taille, exprimée en nombre de bits. À ce jour, une taille de `2`Ko, c'est-à-dire `2048`bits, est considérée raisonnablement sécurisée pour RSA, en tout cas pour un particulier. Même si quelqu'un avec une très grosse puis-

sance de calcul avait envie de casser votre chiffrage pour jeter un œil à ces données, il leur faudrait beaucoup de temps pour y parvenir. Plus la taille de la clé est importante, plus la sécurité augmente, mais le temps de calcul nécessaire pour la générer et pour l'utiliser aussi.

La commande qui permet de générer une paire de clés publique/privée est `ssh-keygen`. Elle peut prendre plusieurs paramètres dont `-t` pour spécifier le type de clés à générer (quel algorithme de clé asymétrique on utilise : RSA, DSA, ...) et `-b` pour spécifier la taille/longueur de la clé. Lorsque vous utilisez la commande `ssh-keygen`, le programme vous demandera de saisir une passphrase. Mais qu'est-ce que c'est ?

2.3.2 Passphrase

Vous avez l'habitude d'utiliser le terme mot de passe. Et vous utilisez cela couramment pour vous connecter sur un ordinateur. Durant le premier cours, nous vous avons rappelé les règles de création d'un mot de passe en lui imposant une taille minimum, mais aussi de ne pas utiliser des choses trop évidentes (dates de naissance, prénom, mots du dictionnaire, etc.).

Par contraste, une *passphrase* est une petite phrase, ou tout au moins un morceau de phrase, donc plusieurs mots. On parle bien de mots, parce qu'en l'occurrence, utiliser de vrais mots et noms ne pose plus guère de problème, dès lors que l'on a une phrase suffisamment longue, surtout si les mots n'ont aucun rapport (ce n'est pas une phrase très connue qui pourrait être testée par dictionnaire de phrases). Vous allez rétorquer que c'est beaucoup plus long à taper qu'un mot de passe. Mais cela peut-être un mot de passe au sens plus classique si vous préférez, mais n'utilisez pas le même que pour vous connecter.

Le but de la passphrase est de protéger l'accès à la clé privée. Donc assurez-vous d'utiliser une passphrase ou un mot de passe pour protéger votre clé privée pour le déchiffrage ou pour la signature. Ceci évitera que même si quelqu'un obtient votre clé privée, il puisse l'utiliser.

Un programme permet aussi de mémoriser la passphrase pour vous éviter, lors d'une session, de devoir la retaper à chaque connexion distante. Cette commande est `ssh-agent`. C'est comme un coffre-fort qui contiendra la passphrase le temps d'une session et qui communiquera les informations nécessaires à chaque fois que celle-ci sera nécessaire.

3 Conclusion

Le protocole SSH est un moyen **fiable** et **sûr** de protéger l'accès à des serveurs et qui simple à mettre en place. Grâce à ce protocole, il devient très difficile d'usurper l'identité d'une machine / d'une personne grâce au système de clé publique et privée uniques générée aléatoirement (en plus d'être chiffrée). Ainsi, pour établir une connexion entre un client et un serveur en SSH, l'authentification (et donc l'accès) ne sera possible que lorsque la vérification des clés entre les parties sera en adéquation. Dans le cas contraire, la connexion est coupée (et pour aller plus loin, il est même possible de black-lister les clés indésirables, corrompues ou autres) et l'authentification rejetée.

Actuellement, il est recommandé d'utiliser l'algorithme RSA avec une clé d'une longueur d'au moins `2048` bits. ML

Remerciements : ce cours a été réalisé grâce à plusieurs sources en ligne :

- https://openclassrooms.com/courses/reprenez-le-controle-a-l-aide-de-linux/la-connexion-securisee-a-distance-avec-ssh
- http://www.git-attitude.fr/2010/09/13/comprendre-et-maitriser-les-cles-ssh/
- https://fr.wikipedia.org/wiki/Chiffrement_RSA
- https://computerz.solutions/le-protocole-ssh/

TD séance n° 15
Internet et Sécurité

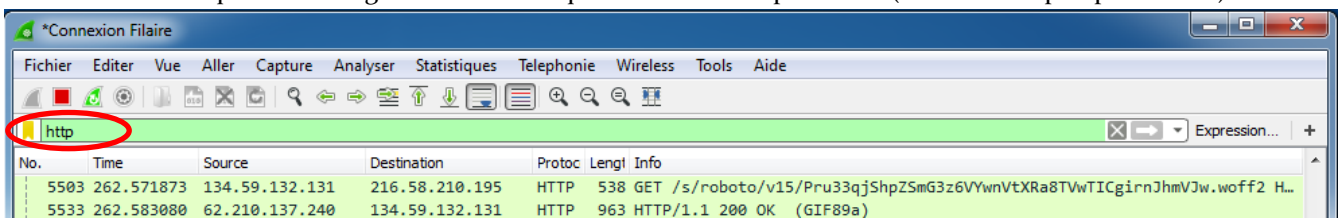
Exercices

Nous allons réaliser ce TD à la fois sous les environnements Linux et Windows. *Les premiers exercices sont à réaliser sous Windows*. Nous signalerons quand changer d'environnement, mais vous devez être à l'aise avec cela maintenant.

4 Exercices obligatoires

4.1 Ecouter les messages sur l'interface réseau d'une machine

Il est possible « d'écouter » les messages qui circulent sur l'interface réseau d'une machine. Un outil simple pour réaliser cette opération est d'utiliser le logiciel *WireShark*[1] à l'aide des ressources du TD. Installez le logiciel sur votre machine (il faut ajouter l'installation de *WinPcap* si l'option n'est pas cochée, mais *USBPcap* n'est pas nécessaire). Puis au lancement du logiciel, sélectionnez la (ou les) interface(s) que vous souhaitez espionner.
Vous pouvez constater la quantité de trafic réseau qui passe sur cette interface (et pourtant vous ne faites rien de particulier). Pour éviter d'avoir trop d'informations, nous allons sélectionner le filtrage suivant le protocole HTTP et ainsi n'écouter que les échanges sur le réseau qui concernent ce protocole (ah ! c'est un peu plus calme).



Exercice n°1:

Ouvrez un navigateur Internet sur la même machine. Connectez-vous à l'adresse http://stephane.lavirotte.com/.
Que constatez-vous ? (par exemple, y'a-t-il un seul ou plusieurs message).

Exercice n°2:

Remontez dans la liste des messages sur un message du type : `HTTP GET / HTTP/1.1`.
A quoi correspondent les autres messages ?

Exercice n°3:

Connectez-vous à l'adresse : http://stephane.lavirotte.com/teach/cours/envinfo1/passwd.html. Remplissez le formulaire (attention à ne pas utiliser votre vrai mot de passe) et validez-le. A l'aide de Wireshark, arrivez-vous à voir passer le mot de passe de manière lisible.

Exercice n°4:

Réalisez la même opération, mais cette fois ci, vous modifierez le filtre de *WireShark* en mettant `tcp.port==443` au lieu de `http` et en vous connectant à l'adresse :

https://stephane.lavirotte.com/teach/cours/envinfo1/passwd.html.

Quelle différence y a-t-il entre ces deux adresses ? Retrouvez-vous le mot de passe en clair ? Retrouvez-vous-même la nature des messages échangés comme tout à l'heure ?

[1] http://www.wireshark.org/

TD séance n° 15
Internet et Sécurité

4.2 Connexion à distance : accès à un terminal distant via SSH

Nous allons maintenant utiliser un client `ssh` pour nous connecter à un serveur `ssh`. L'établissement de cette connexion se fera donc de manière sécurisée et tous les messages qui seront échangés entre les deux machines seront cryptés (chiffrés). Installez pour cela le logiciel `putty` qui est un client `ssh` pour votre machine Windows (vous trouverez `putty` dans l'archive des ressources du TD).

Exercice n°5: Connexion avec authentification simple depuis Windows

Nous allons nous connecter sur un des serveurs ssh de Polytech Nice Sophia. Il y a actuellement deux serveurs qui vous permettent de vous connecter en ssh : `morag.polytech.unice.fr` et `carabosse.polytech.unice.fr`.
Choisissez l'un ou l'autre des serveurs et connectez-vous à l'aide de vos identifiants Polytech Nice Sophia (les mêmes que ceux que vous utilisez pour vous connecter sur la machine physique).
A l'aide de quel outil pouvez-vous vérifier que n'avez pas de soucis à vous faire quant à la confidentialité ?
Dire ce que vous voyez à l'aide de cet outil qui peut vous rassurer sur la confidentialité des messages échangés ?

Nous allons maintenant tenter de faire une connexion avec authentification par clé. Il est donc nécessaire de générer nos propres clés. *Les exercices suivants sont à faire sous Linux (nous reviendrons sous Windows plus tard, cela vous sera précisé).*

Exercice n°6: Génération d'une paire de clés publique/privée depuis Linux

Quelle est la commande sous Linux pour générer la paire de clé publique/privée ? Donnez la commande qui permet de générer des clés RSA d'une longueur de `4096` bits. Attention quant à l'utilisation de cette commande : elle vous demande d'abord le nom de fichier où sauvegarder la clé (faire entrer pour conserver le fichier par défaut), puis il vous demande la passphrase pour protéger votre clé privée. Ne pas mettre une passphrase vide et évitez d'utiliser votre mot de passe de compte.

Quels sont les fichiers qui ont été créés et dans quel dossier ? Quels sont les droits sur ces fichiers ? Expliquez la raison de ces droits ?

Exercice n°7: Connexion avec authentification simple depuis Linux

Toujours sous Linux, donnez la commande pour vous connecter avec votre nom d'utilisateur Polytech Nice Sophia sur la machine `morag.polytech.unice.fr`.

Comment pouvez-vous vous assurer à tout moment dans votre terminal sur quelle machine vous travaillez ? Pour vous déconnecter du serveur `morag` et revenir sur votre machine, il vous suffit de taper la commande `exit`.

Exercice n°8: Copie de la clé publique sur le serveur pour une authentification par clé

Pour arriver à faire une authentification par clé, il est nécessaire d'aller enregistrer la clé publique sur le serveur. La commande permettant de faire cette copie à l'aide de `ssh` est la commande `scp`. Cette commande utilise le protocole SSH pour faire la copie d'un fichier d'une machine à une autre. Vous procéderez donc de la manière suivante :

`scp ~/.ssh/id_rda.pub` *`login`*`@morag.polytech.unice.fr:mypubkey`

Pourquoi copier la clé publique et pas la clé privée ?

Pour que le serveur la connaisse, il est nécessaire de l'ajouter dans le fichier `.ssh/authorized_keys`. Reconnectez-vous donc sur le serveur `morag.polytech.unice.fr`.

Dans le cas où le fichier `.ssh/authorized_keys` n'existe pas, donnez la commande qui permet de déplacer le fichier `mypubkey` au bon endroit avec le bon nom.

TD séance n° 15
Internet et Sécurité

Dans le cas où le fichier `.ssh/authorized_keys` existe, donnez la commande qui permet d'ajouter à la fin du fichier existant la clé contenue dans le fichier `mypubkey`.

Exercice n°9: Connexion avec authentification par clé

Déconnectez-vous de `morag` et tentez de vous reconnecter. Quelle différence constatez-vous ? Pourquoi ?

Exercice n°10: Connexion avec un agent sous Linux

Maintenant que nous pouvons nous connecter par clé, nous pouvons utiliser un agent qui mémorisera en mémoire la passphrase utilisée pour éviter d'avoir à la retaper à chaque connexion distante. Pour cela, il faut faire les actions suivantes :

`ssh-agent`

Copier/coller les variables d'environnement qui s'affichent

`ssh-add ~/.ssh/id_rda`

Saisir la passphrase

Maintenant, connectez-vous sur `morag`. Que constatez-vous ?

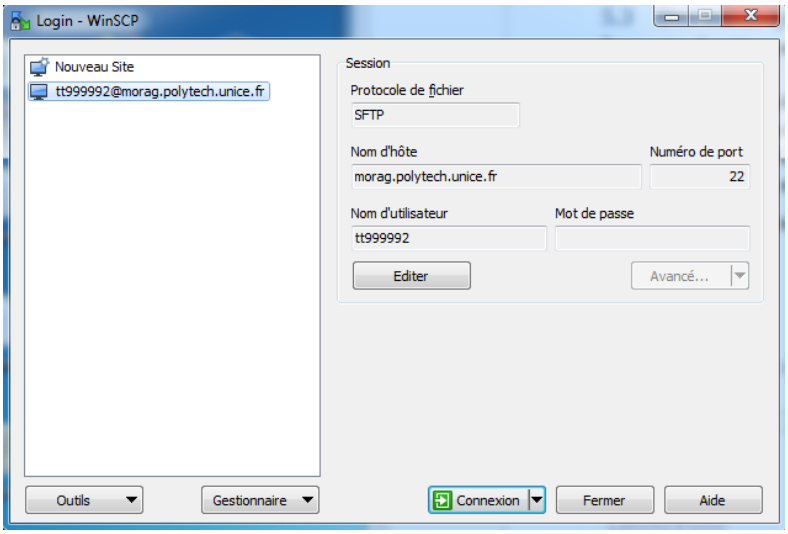
4.3 Des outils qui utilisent SSH pour copier des fichiers à distance ou les éditer

Pour cette dernière partie du TD, *nous retournons sous Windows*.

Se connecter de manière distante à une machine permet de travailler sur celle-ci. Mais il est aussi possible d'avoir accès uniquement aux fichiers qui se trouvent sur cette machine distante à l'aide du même protocole SSH. Pour cela, vous pouvez utiliser un logiciel qui vous affichera le contenu des fichiers sur la machine distante en ouvrant une connexion SSH, et donc de manière sécurisée

Exercice n°11: Accès aux fichiers d'un compte distant

Installez le logiciel WinSCP qui vous est fourni dans les ressources du TD. Configurez-le de manière à vous connecter sur `morag` à l'aide de votre compte Polytech Nice Sophia.



Quel est le type d'authentification qui est utilisé si vous ne faites aucune configuration particulière ?

Exercice n°12: Récupération des fichiers de clés générés sous Linux depuis Windows

Une fois connecté, récupérez les fichiers de clés publique/privée de votre compte sur votre ordinateur Windows. Pourquoi le dossier `.ssh` n'apparaît pas ?

TD séance n° 15
Internet et Sécurité

Pour vous y rendre, il vous suffit de double-cliquer sur le chemin `/sfe/home/…` pour y ajouter un dossier `.ssh`. Copiez le fichier `id_rsa` sur votre machine Windows. A quoi correspond ce fichier ?

Les outils sous Windows utilisent un autre format de sauvegarde des clés. C'est l'outil `PuttyGen` permet de créer ou d'importer et convertir des clés pour une utilisation sous Windows.

Exercice n°13: Import d'une paire de clés publique/privée pour une utilisation depuis Windows

Grâce au menu conversion, importer votre clé privée. Le programme vous demande la passphrase pour garantir que vous êtes bien l'utilisateur que vous prétendez être. Sauvegardez cette clé privée dans un fichier `.ppk` grâce au bouton « Save Private Key ».
Détruisez le fichier `id_rsa` et vider votre corbeille pour que celui-ci ne tombe pas entre de mauvaises mains.

Exercice n°14:

Sous WinSCP, éditez la configuration existante puis aller sur « Avancé… ». Dans la section SSH / Authentification, renseignez le chemin pour votre clé privée (fichier `.ppk`). Sauvegardez votre profil et tentez une connexion sur le serveur morag.polytech.unice.fr. Vous devez maintenant faire une authentification par clé. Qu'est-ce qui vous permet de vous en assurer ?

Exercice n°15: Connexion avec un agent sous Windows

Comme sous Linux, il est possible d'utiliser un programme agent sous Windows pour mémoriser votre passphrase et vous éviter de taper celui-ci à chaque nouvelle connexion. Le programme sous Windows est `Pageant`. Lancez ce programme (il démarre la zone des programmes en bas à droite de l'écran), ajoutez votre clé privée et donnez votre passphrase. Il mémorise ces informations tant qu'il s'exécute (si vous fermez la fenêtre, il continuera à s'exécuter jusqu'au prochain redémarrage de Windows).

Après avoir quitté WinSCP si ce n'est déjà fait, relancez-le. Il ne doit plus vous demander votre mot de passe car `pageant` lui fournit automatiquement. Avez-vous réussi à faire cette connexion sans passphrase ni mot de passe à rentrer manuellement ?

Et voilà, vous avez accès sans mot de passe ou sans passphrase à votre compte Polytech Nice Sophia depuis votre machine Windows. Durant ce TD, nous avons fait ces manipulations depuis la machine virtuelle Windows, mais bien entendu, cela pourrait être réalisé depuis votre machine à la maison pour vous permettre de récupérer vos fichiers sur votre compte Polytech Nice Sophia à la maison ! Elle est pas belle la life !

TD séance n° 15
Internet et Sécurité

Pour aller plus loin

Exercice n°16:

Si je souhaite savoir quels sont tous les nouveaux mails qui sont consultés par la ou les personnes sur la machine, quel est le protocole que je dois surveiller à la place d'HTTP ? Et je peux vous assurer que nous serions alors capables de lire le contenu des mails reçus par la personne sur ce poste de travail. Toutefois pour gagner du temps, nous ne le ferons pas ici.

Et si nous souhaitions espionner tous les mails que la personne envoie et à qui ?

Exercice n°17:

Si je souhaite espionner tous les messages que les machines de la salle échangent avec l'extérieur du réseau local pendant la durée du TD, où dois-je écouter. Justifier avec les connaissances antérieures que vous avez acquises.

Vous comprenez maintenant l'importance de sécuriser les communications que vous avez avec les autres ? Avec l'utilisation d'un protocole sécurisé (HTTPS par exemple), on peut toujours voir les messages passer mais les lire ne sert à rien car le message est incompréhensible si l'on a pas la clé de déchiffrement.

Pourquoi recopier un fichier depuis les serveurs de Polytech sur ma machine locale (mon PC sous Windows par exemple) alors que je pourrai directement l'éditer sur le serveur. Bien sûr, il faut que cela soit sécurisé, donc je vais le faire avec SSH, bien sûr !

Pour éditer un fichier sur la machine distante, il est toujours possible de se connecter sur celle-ci et d'utiliser un éditeur dans la console de connexion (que l'on a pu obtenir avec `putty`). Mais il est aussi possible d'éditer un fichier sur la machine distante avec un éditeur comme Notepad++, ce qui est tout de même beaucoup plus confortable à l'utilisation.

Exercice n°18: Edition à distance de fichiers avec Notepad++ sous Windows

Après avoir installé Notepad++, aller dans le menu « *Compléments* » puis « *Plugin Manager* / *Show Plugin Manager* ». Dans la liste, sélectionnez NppFTP puis Install. Après le redémarrage du logiciel, vous disposez du nouveau plugin NppFTP. Sélectionnez « *Show NppFTP Window* ». Dans la partie droite s'affiche deux nouvelles zones. Sur les paramètres (petite engrenage en haut à droite), sélectionnez « *Profile Settings* ». Vous pouvez alors configurer les paramètres de connexion en faisant « *Add New* ». Les informations à renseigner sont du même type que pour le logiciel WinSCP. Pensez bien à sélectionner le protocole SFTP (Secure FTP) qui utilise SSH. Dans l'onglet authentification vous pouvez renseigner votre clé privée pour réaliser une authentification par clé au lieu de l'authentification simple. Et comme pour les exercices précédents, si pageant s'exécute, vous pourrez vous connecter dans avoir à taper ni passe phrase, ni mot de passe.

Exercice n°19: Utiliser les commandes Unix sous Windows

Et pour boucler la boucle, maintenant que vous avez apprécié les commandes Unix et tout ce qu'elles permettent, il ne reste qu'un pas à franchir pour utiliser celles-ci sous Windows. Pour cela, il vous suffit d'installer Cygwin
Rendez-vous à l'adresse http://www.cygwin.com/ et vous pourrez installer tout un environnement Unix sous Windows.

Exercice n°20: Création d'une paire de clés publique/privée pour une utilisation depuis Windows

Créez une paire de clés pour votre ordinateur sous Windows. A votre avis, pourquoi l'outil vous demande de bouger frénétiquement la souris ?

TD séance n° 15
Internet et Sécurité

Mais le problème est alors, comment ajouter la clé publique dans le format ppk à mon serveur sous Linux. Ah et bien là, je n'ai pas de solution pour le moment. C'est pourquoi, nous avons procédé autrement durant le TD.