Faculté
des Sciences
Département des Sciences Informatiques

UMONS
Université de Mons

Sécurité des systèmes informatiques

Auteur : Quentin Loos
1ère Master en sciences informatiques
Année académique 2012–2013

Faculté des Sciences • Université de Mons • 6, Av. du champ de Mars • 7000 Mons

Table des matières

Première partie
Unix

1 Protections des fichiers

1.1 Fichiers

Dans l'environnement du système d'exploitation Unix, tout est fichier ! Un fichier usuel possède un utilisateur propriétaire et un groupe propriétaire. Pour chaque fichier, des permissions de lecture, d'écriture et d'exécution sont stockés pour l'**utilisateur propriétaire**, le **groupe propriétaire**, et les **autres**. Ces permissions peuvent être écrite numériquement (4 = lecture, 2 = écriture, 1 = exécution). Un fichier exécutable doit donc avoir le droit d'exécution pour être lancé.
On peut lister les droits d'accès d'un fichier avec la commande *ls -l*
Ces droits d'accès sont représentés ainsi :

```
-rw-r--r-- 1 kent kent    6601 jan 12 19:48 cours.tex
```

1.2 Répertoires

Un répertoire, qui est un fichier, contient donc aussi des permissions. La lecture permet de **lister ce qu'il contient**, l'écriture permet d'**y mettre un fichier**, et l'exécution permet de le **sélectionner comme répertoire courant**.
Ces droits d'accès sont représentés ainsi :

```
drwxr-xr-x  2 kent kent  4096 jan 12 19:48 Sécurité des systèmes informatiques
```

1.3 Umask

Puisque le mécanisme de protection des fichiers est si important dans le système d'exploitation Unix, il est évident qu'un réglage approprié des bits de droits d'accès est requis pour une sécurité globale. Hormis l'ignorance de l'utilisateur, les cas de compromission de fichier les plus communs sont dus au réglage par défaut des bits de droit d'accès à la création des fichiers. Pour certains systèmes, ce réglage par défaut est l'octal 644, ce qui signifie que seul le propriétaire du fichier peut y écrire et y lire, pendant que tous les autres ne peuvent qu'y accéder en lecture. Dans nombre d'environnements « ouverts », ceci peut être acceptable. Cependant, au cas où il existe des données sensibles, l'accès en lecture pour les autres devrait être désactivé. La commande **umask** permet une configuration des permissions à la création de fichiers.

Voici quelques valeurs communues de umask.

```
umask 000: 666 & ~000 = 666 soit rw-rw-rw- (pas de sécurité)
umask 002: 666 & ~002 = 664 soit rw-rw-r-- (sécurité minimum)
umask 022: 666 & ~022 = 644 soit rw-r--r-- (sécurité modérée)
umask 027: 666 & ~027 = 640 soit rw-r----- (sécurité forte)
umask 077: 666 & ~077 = 600 soit rw------- (sécurité la plus forte)
```

1.4 SUID/SGID

Les fichiers exécutables peuvent changer les permissions de l'utilisateur pendant son exécution. Par exemple, pendant l'exécution de */usr/bin/passwd*, l'utilisateur obtient les droits **root** pour pouvoir accéder à */etc/passwd* et ainsi changer son mot de passe. Bien sûr, un utilisateur ne peut jouir du droit SUID que s'il détient par ailleurs les droits d'exécution du programme. Ce droit est utilisé lorsqu'une tâche, bien que légitime pour un utilisateur classique, nécessite des droits supplémentaires (généralement ceux de root). Il est donc à utiliser avec précaution.

Un fichier avec les droits

```
-rwxr-xr-x
```

auquel on ajoute le droit SUID aura donc la notation

```
-rwsr-xr-x
```

Le droit SUID possède la valeur octale 4000. L'exemple ci-dessus a donc comme valeur 4755.
Le droit GUID possède la valeur octale 2000. Le 's' se mets à la place de l'exécution groupe dans ce cas.

Sur les systèmes Unix, Setuid n'a pas d'effet sur les répertoires.
Le fonctionnement de Setgid sur un répertoire est très différent de celui sur les exécutables : si cette propriété est appliquée à un répertoire, tout fichier ou sous-répertoire créé dans ce répertoire parent appartiendra au groupe de celui-ci et non au groupe de l'utilisateur qui crée l'élément.
Lorsqu'un sous-répertoire est créé, il hérite lui aussi de la propriété Setgid.

SUID et SGID sont à éviter, d'un point de vue sécurité. En plaçant les bits de droits d'accès suid ou sgid sur un fichier exécutable, d'autres utilisateurs peuvent obtenir un accès aux mêmes ressources (par l'intermédiaire du fichier exécutable) comme le réel propriétaire du fichier. Ainsi si vous tapez **chmod ug+s /bin/bash** vous donnez les droits root à toute personne qui ouvre un terminal ou qui lance l'interpréteur de commande bash. Ou si on modifie malicieusement **/usr/bin/passwd**, on peut par exemple enregistrer les mots de passe tapés.

Pour corriger cette faille de sécurité, les fichiers suid/sgid accessibles en écriture devraient être recherchés et supprimés du système. Le signalement de tels fichiers par les utilisateurs normaux est aussi essentiel pour corriger des brèches de sécurité existantes. On peut par exemple garder une liste de ces fichiers dans un endroit sûr grâce à la commande

```
find / \( -perm -2000 -o -perm -4000 \) -type f -exec ls -la {} \; > ...
```

1.5 Sticky bit

Ce droit est utilisé pour manier de façon plus subtile les droits d'écriture d'un répertoire. En effet, le droit d'écriture signifie qu'on peut créer et supprimer les fichiers de ce répertoire. Le sticky bit permet de faire la différence entre les deux droits.
Lorsque ce droit est positionné sur un répertoire, les fichiers créés gardent la marque de leur propriétaire. Ainsi, ils ne peuvent être supprimés ou renommés par d'autres utilisateurs. La création de nouveaux fichiers est toujours possible pour tous les utilisateurs possédant le droit d'écriture sur ce répertoire.

Un fichier avec les droits

```
-rwxr-xr-x
```

auquel on ajoute le droit sticky bit aura donc la notation

```
-rwxr-xr-t
```

Le droit sticky bit possède la valeur octale 1000. L'exemple ci-dessus a donc comme valeur 1755.

L'exemple le plus pertinant pour le sticky bit est le dossier */tmp/* qui doit être accessible en écriture par tous les utilisateurs, sans que ceux-ci se suppriment leurs fichiers les uns les autres. Ils peuvent cependant les modifier.

1.6 Lien symbolique

Un lien symbolique est un fichier dans les systèmes Unix modernes qui permet de référencer de manière quasi-transparente un autre fichier. On peut dire qu'un lien symbolique est un alias d'un fichier ou d'un répertoire. Un lien symbolique a toujours les mêmes droits d'accès que le fichier sur lequel il pointe. Les droits d'accès indiqués pour un lien symbolique sont sans signification.

```
9839893 lrwxrwxrwx  1 kent kent    9 jan 12 21:45 cours_lien.tex -> cours.tex
```

Contrairement aux liens matériels, les liens symboliques peuvent pointer sur des fichiers, des répertoires, sur eux-mêmes ou sur des destinations qui n'existent pas (l'existence du nom pointé n'est même pas vérifiée lors de la création du lien par la commande *ln*). C'est seulement au moment d'accéder à un lien symbolique que la vérification est faite. Quand la destination d'un lien symbolique n'existe pas, on dit que le « lien est cassé ». Une tentative d'ouvrir un lien cassé conduit à un message d'erreur de type « fichier non trouvé », assez surprenant car le lien symbolique existe bel et bien.

1.7 D'autres fichiers

Liens physiques, sockets, devices, pipes, ... Ils contiennent aussi des droits. Si on change les droits d'un hard link, on change les droits de tous les fichiers, puisque les droits sont stockés dans les i-noeuds.

1.8 Changement des droits

Attention à l'usage de **chown**, surtout avec l'option **-R**. Il vaut mieux le faire fichier par fichier qu'avec l'option récursive. De plus, s'il y a des liens symboliques, au change aussi les permissions des fichiers pointés et on suit les répertoires pointés par les liens !
On peut à la limite utiliser l'option **-h** qui ne suit pas les liens symboliques.

1.9 Access Control List

Access Control List, abrégé en ACL, permet d'étendre les permissions de lecture, écriture et exécution. Elle permet un contrôle à la carte en fontion des différents utilisateurs ou groupes. Ainsi, il devient possible d'autoriser un utilisateur tiers à effectuer des opérations sur un fichier sans autoriser tout un groupe ou tout le reste du monde.

```
$ getfacl cours.tex
# file: cours.tex
# owner: kent
# group: kent
user::rw-
user:root:r--
user:kent:rw-
group::r--
mask::rw-
other::---
```

1.10 Résumé

Voici plusieurs astuces pour sécuriser votre machine Unix.

- Utiliser les points de montages, qui permettent d'empêcher l'exécution de binaire et de désactiver les effets du bit SUID.
- Ne pas être faignant et accorder 777 à un fichier qu'on veut partager avec qqun.
- Configurer le umask au minimum nécessaire.
- Recherche des programmes SUID et SGID. On peut aussi rechercher des répertoires modifiables par tout le monde par exemple.
- Quand c'est nécessaire, créer des autorisations plus strictes et souples avec les ACL.

– Protéger et consulter les logs !

2 Mots de passe

2.1 Shadow files

Le système Unix permet à tous les utilisateurs d'avoir un accès en lecture au fichier */etc/-passwd* où sont stockés des informations sur les utilisateurs. Par exemple, *ls -l* va accéder à */etc/-password* pour faire la conversion des UID des propriétaires de fichiers avec leur nom d'utilisateur. Bien que le système Unix applique une méthode de chiffrage dite à sens unique (hashage), et dans la plupart des systèmes une version de norme de chiffrement de données modifiée (DES, data encryption standard, norme de chiffrement de données), on connaît des méthodes pour casser des mots de passe. Parmi ces méthodes, les attaques par brute-force sont généralement les moins efficaces, tandis que les techniques utilisant des dictionnaires ou des heuristiques (comme des bons essais et des informations sur le mots de passe) ont tendance à réussir.

C'est pourquoi, les shadow files sont utilisés pour augmenter la sécurité en cachant les chiffrement des mots de passe. Ces hashs sont dans des fichiers que seul root peut consulter (ou les membres du groupe shadow). Par exemple, il n'y a plus de mot de passe dans */etc/passwd*, ils sont stockés dans */etc/shadow*.

2.2 Bons mots de passe

Le fichier */etc/passwd* contient des informations utiles telles que le nom d'utilisateur et des champs de commentaire. Les noms d'utilisateur sont particulièrement fructueux pour les casseurs de mot de passe, puisque nombre d'utilisateurs emploient des variantes de ces noms pour leur mots de passe (orthographe inverse, ajout d'un seul chiffre, etc.). Le champ de commentaire contient souvent des éléments tels qu'un nom de famille, un prénom, une adresse, un numéro de téléphone, le nom d'un projet et ainsi de suite.
Ainsi, il est fort probable qu'un assaillant persévérant trouvera des informations sur les utilisateurs dans le fichier /etc/passwd.

Un mot de passe devrait être changé régulièrement (et ne pas alterner entre deux ou trois mots de passe), un nombre important de caractères, des minuscules, majuscules, caractères spéciaux. Il ne faut toutefois pas exagérer, sinon on finira par les noter . . .
Les bons mots de passe sont faciles à se rappeler, ne sont pas des noms et ne se trouvent dans aucun dictionnaire.

3 Usage du compte root

Root, c'est le super utilisateur par excellence. Sous Unix, il peut tout faire, le meilleur comme le pire. Il a les pleins pouvoirs, et comme personne n'est au dessus de root, personne ne peut limiter l'étendue des dégats qu'il s'apprête à commettre.

Ccomment peut-on en arriver à faire des dégats ?

3.1 Les commandes fatales

Le premier cas, est le cas des commandes fatales. Celles dont on ne revient pas. Par exemple, un *rm **, ou pire un *rm -r **. Le premier est une suppression simple de tout fichier du répertoire courant. Le deuxième est fait la même chose, sauf que l'option **-r** permet de faire un récursivité, et donc déscendra dans les sous-répertoires et supprimera les fichiers.
En root, il existe une sécurité : la commande rm est un alias qui force la vrai commande rm à faire une demande de confirmation avant d'effectuer la supression réel du répertoire. Malgré tout, si notre commande *rm -r ** devient *rm -rf **, il ne demandera plus de confirmation (-f = force). Pour ceux qui doutent de la probabilité de pouvoir jamais taper cette commande, qu'ils considèrent le cas de quelqu'un qui souhaite retirer tout les fichiers musicaux, via un *rm -rf*

.mp3, pour faire un peu de place, et que par inadvertance, il tape un espace de trop, entre l'étoile et le point (*rm - rf * .mp3*). I just fucking shot myself !

3.2 Utilisation excessive

Le programme que nous avons télécharger se décompresse bien dans un répertoire à part, pas de problèmes de compilation, il n'y à plus qu'à lancer. Vous avez alors plusieurs cas possibles. Dans le cas le plus fréquent (et heureusement le plus fréquent) il ne se passe rien de notable. L'application fait son travail et rien d'autres. Les autres cas marginaux, devraient tout de même vous faire réfléchir. Et si, en s'installant, cette nouvelle configuration « écraisait » votre configuration actuelle ? Et si un utilisateur avait modifié le code pour y introduire un cheval de troie ou autres virus . . .Vous venez de laisser un grand trou dans vos défenses, car root à le droit de tout faire sur votre machine, y compris celui de modifier des droits sur des fichiers, changer le contenu d'autres fichiers, remplacer des fichiers par d'autres, mettre en place un service « pirate » sur n'importe quel port, et eventuellement écouter ce qui se passe sur votre réseau.
On utilise le compte root que quand c'est necessaire !

Dans l'exemple de **ls -al | grep sh**, si on oublie de grep, on exécute tous les exécutables présents dans */usr/bin*, en **root** ! De plus, s'il y a des liens symbolique, alors à cause du caractère '>', on va écraser le bon original !

4 PATH

PATH est une variable d'environnement qui pointe sur une liste de répertoires. Cette liste de répertoire est en fait les chemins d'accès aux exécutables. On n'est ainsi pas obligé de taper **/usr/bin/ls**, mais simplement **ls**, puisque */usr/bin/* est dans le PATH. L'ordre de cette recherche est indiquée par la séquence des répertoires listés dans le nom du PATH.

Si un utilisateur place le répertoire courant en première place dans le PATH, alors les programmes du répertoire courant seront exécutés en premier. Des programmes homonymes dans d'autres répertoires seront ignorés. Bien que l'accès aux fichiers et aux répertoires soient plus aisés avec une variable PATH définie de cette manière, il peut exposer un utilisateur à des chevaux de Troie préexistants.
Pour illustrer ceci, supposons qu'un cheval de Troie, semblable au programme **cat**, contienne une instruction qui donne des droits d'accès privilégiés à un assaillant. Le programme **cat** factice est placé dans un répertoire public */tmp/* , dans lequel un utilisateur travaille souvent. À présent, si l'utilisateur dispose d'une variable PATH avec en premier le répertoire courant, et qu'il exécute la commande **cat** dans */tmp/*, le programme **cat** factice dans */tmp/* sera exécuté et non la commande système **cat** située dans */bin/*.

Si un utilisateur place le répertoire courant en dernière place dans le PATH, alors les programmes du répertoire courant peuvent être exécutés. Supposons qu'un utilisateur souhaite lister les éléments du dossier courant mais qu'au lie de taper **ls** il tape **sl**. Si un programme malicieux **sl** existe, il sera exécuté.

Pas de chemins relatifs dans PATH !

5 NFS - Network File System

NFS fonctionne bien pour partager des systèmes de fichiers entiers avec de nombreux hôtes connus et de façon très transparente. Nombre d'utilisateurs ont accès à des fichiers sur un montage NFS sans même s'apercevoir que le système de fichiers qu'ils utilisent n'appartient pas à leur système local. Cependant, la facilité d'utilisation va de paire avec toute une série de problèmes potentiels de sécurité.

5.1 Accès hôte

NFS contrôle qui peut monter et exporter un système de fichiers en fonction de l'hôte présentant la demande de montage et non en fonction de l'utilisateur qui utilise le système de fichiers. Les hôtes doivent obtenir des droits explicites pour monter un système de fichiers. Le contrôle d'accès n'est cependant pas possible pour les utilisateurs, si ce n'est les autorisations de fichier et de répertoire. Autrement dit, lorsque vous exportez un système de fichiers via NFS vers un hôte distant, vous ne faites pas confiance uniquement à l'hôte qui monte le système de fichiers, mais aussi aux utilisateurs qui ont accès à cet hôte et, par conséquent, à votre système de fichiers. Ce risque peut être contrôlé, en exigeant des montages en lecture seule par exemple ou en réduisant les utilisateurs à un ID utilisateur et groupe commun, ce qui peut empêcher toutefois que le montage ne soit utilisé de la façon souhaitée à l'origine.

5.2 Autorisations de fichier

Lorsqu'un système de fichiers est monté en lecture-écriture par un hôte distant, la seule protection pour les fichiers partagés repose dans leurs autorisations et leur appartenance à un utilisateur et un groupe. Si deux utilisateurs partageant la même valeur d'ID utilisateur montent le même système de fichiers NFS, ils pourront modifier leurs fichiers l'un et l'autre. De plus, tout individu connecté en tant que super-utilisateur sur le système client peut utiliser la commande su pour devenir un utilisateur ayant accès à des fichiers spécifiques via le partage NFS.

Le comportement par défaut lors de l'exportation d'un système de fichiers via NFS est d'utiliser **root squashing**. Dans ce cas, si un utilisateur du client avec l'UID 0 essaye d'accéder (en lecture, écriture ou effacement) au système de fichiers, le serveur remplace l'UID par celui de l'utilisateur *nobody* du serveur. Ceci signifie que l'utilisateur root du client ne peut accéder/modifier les fichiers du serveur que seul le root du serveur peut accéder/modifier. Mais l'utilisateur root du client peut toujours utiliser **su** pour devenir n'importe qui et accéder à ses fichiers ! Ceci a une conséquence importante : tous les fichiers et binaires importants devraient appartenir à root, et pas bin ou un compte autre que root, car le seul compte auquel le root du client ne peut pas accéder est le compte root du serveur.

Si vous n'autorisez les utilisateurs qu'à lire les fichiers via votre partage NFS, vous pourriez utiliser l'option *all_squash*, qui fait en sorte que tous les utilisateurs accédant à votre système de fichiers exporté aient l'ID utilisateur de l'utilisateur nobody.

5.3 Sécurité client

Du côté client il y a quelques options de mount qui permettent de ne pas faire trop confiance au serveur. L'option nosuid interdit le démarrage de programmes suid depuis le système de fichiers NFS. C'est une option à utiliser systématiquement, car elle empêche le root du serveur de créer un fichier suid sur le système de fichiers NFS, puis de se loger dans le client en utilisateur et de lancer le programme suid pour devenir root sur le client. Il est aussi possible d'interdire l'exécution des fichiers du système de fichiers NFS avec l'option noexec. Mais ceci est beaucoup moins utile que nosuid car le système de fichiers contiendra très probablement au moins quelques scripts ou programmes à exécuter. Ces options se rentrent dans la colonne d'options, avec wsize et rsize, séparées par des virgules.

Accès en lecture, pas d'accès *root*.

Deuxième partie

Vulnérabilités dans les applications

L'installation d'applications sur votre ordinateur peut créer des vulnérabilités sur votre système. À moins d'être familier avec leur configuration, de télécharger les plus récentes mises à jour et de se tenir régulièrement informé des dernières brèches de sécurité qui sont périodiquement découvertes, l'installation de serveurs Proxy, FTP (en particulier FTP anonyme), SMTP, Sendmail, les services à distance (rsh, rlogin, etc.) et autres du même genre offrent de grands risques de créer des failles dans votre système. N'activez pas de tels services pour un ordinateur personnel

6 Place de l'ordinateur dans le réseau

L'ordinateur le plus sécurisé des attaques extérieures, est celui qui n'est pas connecté à un réseau ! Cependant, c'est souvent en réseau qu'un ordinateur est le plus utile. Il est donc capital de réfléchir sur sa place dans le réseau. Par exemple, un serveur d'impression doit-il être accessible depuis l'exterieur d'un lan ? Une adresse IP privée suffit peut-être, quitte à faire un VPN.

Un réseau d'ordinateurs n'est pas plus sûr que les machines qu'il connecte. Un seul hôte non sûr peut causer beaucoup de problèmes au réseau entier. Des firewalls ou des systèmes de détection d'intrusions sont inutiles si votre serveur comporte des services faciles à compromettre. Il est donc important de n'installer que ce qui est réellement nécessaire, et pas de services inutiles.

7 FTP anonyme

File Transfer Protocol est un protocole de communication destiné à l'échange de fichiers sur un réseau TCP/IP. Il permet, depuis un ordinateur, de copier des fichiers vers un autre ordinateur du réseau, ou encore de supprimer ou de modifier des fichiers sur cet ordinateur. Ce mécanisme de copie est souvent utilisé pour alimenter un site web hébergé chez un tiers.

Le FTP anonyme permet la connexion à un serveur ftp sans entrer de login. Cependant il faut l'éviter car laisser trop de répertoires en droit d'écriture et/ou d'exécution est plus que dangereux pour la sûreté du système. Le pirate pourrait y installer ou y exécuter des codes malveillants lui permettant d'accroître son pouvoir sur la machine.
L'idéal est de créer un répertoire pour le download en anonyme, ce répertoire doit avoir comme propriétaire root pour eviter l'upload de fichiers.

Installez un serveur FTP anonyme seulement en cas d'absolue nécessité. Si vous devez le faire, limitez au maximum les droits sur les différents répertoires et fichiers laissés au public.

Si il est quand même nécessaire d'autoriser l'upload, malgré que cela ne soit pas conseillé, il faut utiliser un système de fichier différent.

Si vous voulez que les utilisateurs anonymes puissent déposer des fichiers, créez le répertoire ftp/pub/incoming (propriétaire root, droits 733). Faîtes un **chmod +t ftp/pub/incoming**. Normalement le démon FTP interdit aux utilisateurs anonymes d'écraser un fichier existant, mais un utilisateur normal pourrait détruire n'importe quoi. En mettant les droits à 1733 ce ne sera plus possible. On pourrait aussi configurer ftp pour que les fichiers créés le soient avec les droits 600 et appartiennent à root (ou tout autre utilisateur).

8 Serveur Web

Un serveur web ne doit pas avoir plus de droits que nécessaire ! Par exemple, un programme peut être exécuté sur un serveur web, et on peut lui passer tout un tas de requêtes. Toutefois une erreur du même type que l'injection sql permet d'exécuter des commandes avec les droits du service exécuté sur le serveur web. Si le serveur web tourne en tant que **root** cela pose un énorme problème ! Mieux vaut utiliser un utilisateur particulier, par exemple **www-data**.

Mettre les privilèges des démons au minmum nécessaire, ne pas faire tourner les serveurs en root

Imaginons la mise en place d'un CMS. On upload donc les fichiers (les fichiers appartiennent à **toto**, par exemple). Le serveur web (**www-data**) doit avoir aussi accès aux fichiers uploadés. Il serait tentant d'autoriser la lecture à tous, mais cela entraîne un problème de sécurité. Une solution serait d'ajouter **www-data** aux groupes de l'utilisateur (c'est à dire ici, le groupe **toto**). Ainsi, en autorisant le lecture aux membres du groupes **toto**, **www-data** pourra lire les fichiers uploadés, étant dans ce groupe.

Le serveur web doit aussi pouvoir écrire dans certains dossiers. Cependant, il faut que ce qui est uploadé puisse être lisible par l'utilisateur. Si on upload un fichier à partir du site, il aura les droits du serveur web (**www-data**). Cela impose qu'il y ait un dossier avec les droits en écriture pour le groupe **www-data**. Pour que l'utilisateur puisse lire le fichier, on va mettre un SGID sur le dossier *upload*. Ainsi les fichiers créés par **www-data**, auront comme groupe **toto**.

9 Attaque par symblink

Ce type d'attaques consiste a créér dans */tmp/* un lien symbolique d'un fichier temporaire qu'une application va peut-être créer. Ce lien symbolique pointe vers un fichier arbitraire (par exemple, */etc/passwd*). Lorsque l'application créera son propre lien, si cette application est lancée par root, elle écrasera le lien existant et donc le fichier *passwd*.

10 Injection SQL

Une injection SQL est un type d'exploitation d'une faille de sécurité d'une application interagissant avec une base de données, en injectant une requête SQL non prévue par le système et pouvant compromettre sa sécurité. On peut injecter du code SQL, à travers un champ de type formulaire par exemple. En devinant le code sql derrière l'application, on peut exécuter du code SQL utile. La solution est de traiter la chaine de caractère en échappant certains caractères (*mysql_real_escape_string*).

11 Buffer overflow

En informatique, un dépassement de tampon ou débordement de tampon (en anglais, buffer overflow) est un bug par lequel un processus, lors de l'écriture dans un tampon, écrit à l'extérieur de l'espace alloué au tampon, écrasant ainsi des informations nécessaires au processus.

Lorsque le bug se produit non intentionnellement, le comportement de l'ordinateur devient imprévisible. Il en résulte souvent un blocage du programme, voire de tout le système.

Le bug peut aussi être provoqué intentionnellement et être exploité pour violer la politique de sécurité d'un système. Cette technique est couramment utilisée par les pirates. La stratégie de l'attaquant est alors de détourner le programme bugué en lui faisant exécuter des instructions qu'il a introduites dans le processus.

À ce jour, ils existent encore beaucoup de buffer overflow car il existe encore beaucoup d'interpréteurs écrit à la base en C, qui eux même peuvent contenir des erreurs de programmation pouvant amener à des buffers overflow.

On peut ainsi modifier certaines données dans la pile, et par exemple injecter du code, changer l'adresse de retour de la fonction, ...

12 Majordomo

Une faille a existé dans Major Domo, on peut ainsi exécuter des commandes en **root**. Passer à une nouvelle version de l'application ne suffit pas toujours, le makefile doit être modifié car la configuration par défaut est dangereuse

Major Domo, c'est de la merde en somme !

13 Sendmail

Sendmail a besoin des privilèges **root**. Plusieurs raisons essentielles permettent de jusitifier l'exécution de Sendmail en tant que root, notamment :

- Sendmail écoute sur le port 25, port privilégié (inférieur à 1024).
- Les utilisateurs génèrent des fichiers .forward dans leur répertoire personnel afin de transférer leurs messages vers d'autres adresses SMTP. Sendmail doit pouvoir accéder à ce fichier afin d'en lire le contenu, même si les utilisateurs ont disposé un droit d'accès restrictif sur leur répertoire personnel (mode 700).
- endmail doit pouvoir devenir temporairement propriétaire des fichiers de listes de messagerie (mailing-lists) gérées par les utilisateurs (en prenant l'identité de l'utilisateur).

On ne veut donc pas pouvoir laisser du code s'exécuter en tant que **root** à travers de Sendmail. On va donc utiliser un shell restreint (**smrsh**).

13.1 D'autres précautions

Désactiver vrfy et expn La variable *vrfy* vérifie si l'utilisateur existe bien, est bien créé dans la DB du mailer –> à désactiver pour garder la confidentialité des users. EXPN permet aussi de lister les personnes d'une liste ou les alias d'une adresse.
La variable *expn* permet aussi de lister les personnes d'une liste ou les alias d'une adresse.

Attention à la longeur maximale des messages → perte d'efficacité des serveurs

Relais SMTP : il faut bien configuere le serveur mail pour éviter qu'il relaire du spam par ex. Une connexion VPN peut passer outre cette protection –> rejeter les relais SMTP non désirés

Mail Queue donne tous les messages qui ont subis une erreur lors de l'envoie. Le problème c'est qu'on voit les adresses email des autres et à qui ils ont envoyés.
Sendmail a été corrigé, et dispose maintenant d'un SGID, plutôt que d'un SUID root.

Sur AIX, version pourri. On compile soit meme. Lors d'une réinstallation, il faut recréer les liens symboliques pour être sûr que c'est la nouvelle version qui sera utilisée.

Il existe une option pour que sendmail refuse de s'exécuter avce des droits sur les répertoires trop légers (*DontBlameSendmail*)

Si sendmail a besoin absolument de droits d'accès sur des répertoires, les laisser tel quel et utiliser d'autres répertoires pour les fichiers de config

14 TCP Wrapper

TCP Wrapper est un terme anglais qui signifie littéralement emballage TCP. Il s'agit d'une technique de sécurité particulière aux réseaux gérés par des systèmes Unix.
Afin de restreindre et de tracer les accès de certains services en fournissant l'origine de la requête sans pour autant changer les sources des daemon ni réduire l'accès à des services qui permettraient les tentatives d'intrusion, un programme se charge de lancer le serveur voulu si le client est autorisé à se connecter.

Initialement seulement disponible pour les super-serveurs comme inetd, ce service d'ACL réseau est disponible à tout daemon qui est lié avec la bibliothèque logicielle libwrap.
Il permet de filtrer l'accès aux démons par adresse IP, sous-réseau, et nom d'hôte. En comparaison du contrôle d'accès qu'on trouve généralement dans la configuration des démons, TCP Wrapper permet la reconfiguration à chaud sans avoir besoin de redémarrer un daemon pour modifier les règles de filtrage.

Les démons sont soit lancés au démarrage de la machine, soit à la demande, quand c'est necessaire.
Les TCP Wrappers rajoutent une protection supplémentaire aux services lancés par inetd (qui permet de lancer des serveurs démons), c'est en fait un contrôle d'accès, par exemple pour se connecter avec telnet, il faut d'abord passer le contrôle des TCP Wrappers, une fois passer ce contrôle on peut alors se connecter à telnet, si on ne passe pas le contrôle des TCP Wrappers, la session telnet n'est même pas ouverte. Les TCP Wrappers permettent évidemment de loguer chaque connexion (réussie ou pas). Les fichiers de configuration des TCP Wrappers pour fixer les accès sont */etc/hosts.allow* et */etc/hosts.deny*. Ces fichiers déterminent qui peut ou ne peut pas accèder aux systèmes (ou du moins aux services lancés par inetd).
La règle est d'interdire tout à tout le monde, puis d'autoriser uniquement certains postes très limités à utiliser vos services.

15 Antivirus

fichiers .mc utiliser un shell restreint, appliquer des filtres, définir des alias pour les répertoires

MIMEDefang filtre à utiliser(PERL) qui détecte les virus, les mets en quarantaine (au cas où qqun en aurait besoin) et permet aussi de bloquer certains types de fichiers (mp3).

McAfee Le problème sous Unix, McAfee n'est pas un démon mais une ligne de commande lancé à chaque fichier à scanner. Donc, à chaque mail, contenant une pièce jointe, on lance une instance de McAfee (~100Mb).

On change l'extension des fichiers exe par exemple, pour qu'il ne soit plus exécutable sous Windows.

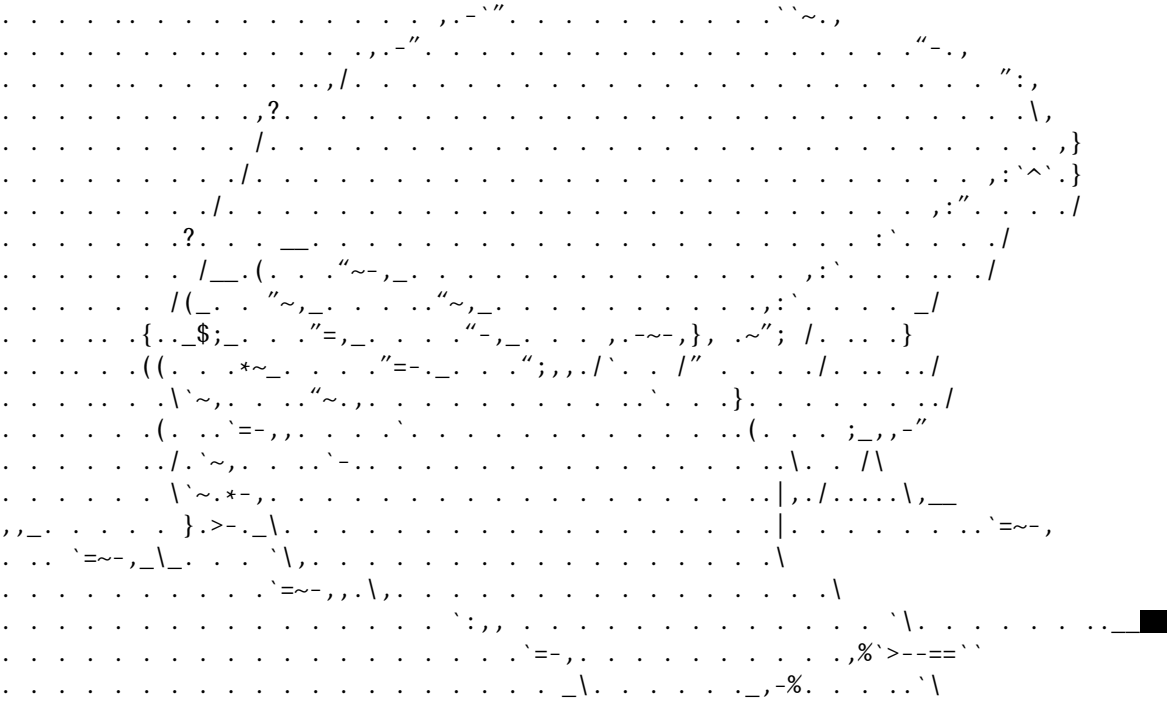
Est-ce qu'on préviens le destinataire qu'un des mail qu'il a reçu contient un virus ? Non, on prévient pas les gens.

Est-ce qu'on préviens l'expéditeur Non, c'est souvent une fausse adresse.

Installer un client antivirus Au cas où le mail contient un lien vers un virus.

Troisième partie

Windows



Que dire ? 3 captures d'écran ...

- Une partition pour Windows, une autre pour les données (NTFS)
- Sur un ordinateur critique, pas de dual-boot
- Attention aux partages réseau (pas sans mot de passe)
- Sauvegardes

Quatrième partie

Crytpographie

16 PGCD

$pgcd(a,b)$ est égal au plus petit entier positif de $I = \{ax + by \mid x, y \in \mathbb{Z}\}$

Bézout Deux nombres a, b sont premier entre eux, si le $pgcd(a,b) = 1$

Théorème d'Euclide Si $b \leq a, pgcd(a,b) = (b, a \mod b)$

Petit théorème de Fermat Si p est un nombre premier et si a est un entier non divisible par p, alors $a^{p-1} - 1$ est un multiple de p.
ou
si p est un nombre premier et si a est un entier quelconque, alors $a^p - a$ est un multiple de p.

17 Clés asymetriques

Pour résoudre le problème de l'échange de clés, la cryptographie asymétrique a été mise au point dans les années 1970. Elle se base sur le principe de deux clés : une publique, permettant le chiffrement et une privée, permettant le déchiffrement.
Comme son nom l'indique, la clé publique est mise à la disposition de quiconque désire chiffrer un message. Ce dernier ne pourra être déchiffré qu'avec la clé privée, qui doit rester confidentielle.

17.1 RSA

- Choisir p et q, deux nombres premiers distincts
- Noter n leur produit, appelé « module de chiffrement » : $n = pq$
- Calculer l'indicatrice d'Euler de n : $\varphi(n) = (p-1)(q-1)$.
- Choisir e, un entier premier avec $\varphi(n)$, appelé « exposant de chiffrement ».
- Comme e est premier avec $\varphi(n)$, il est, d'après le théorème de Bachet-Bézout, inversible mod $\varphi(n)$, c'est-à-dire qu'il existe un entier d tel que $ed \equiv 1 \pmod{\varphi(n)}$.

L'entier d est l'exposant de déchiffrement. Le couple (n,e) est appelé clé publique, alors que le couple (n,d) est appelé clé privée.

Si M est un entier inférieur à n représentant un message, alors le message chiffré sera représenté par $C \equiv M^e \pmod{n}$

Pour déchiffrer C, on utilise d, l'inverse de $e \mod \varphi(n)$ et on calcule $C^d \pmod{n}$. On a alors,

$$C^d \pmod{n} \equiv (M^e)^d \pmod{n} \equiv M^{ed} \pmod{n}$$

Comme $ed \equiv 1 \pmod{\varphi(n)}$ par définition de modulo, on a

$$ed = 1 + k\varphi(n) = 1 + k(p-1)(q-1), k \in \mathbb{N}.$$

Or, pour tout entier M,

$$M^{1+k(p-1)(q-1)} \equiv M \pmod{p}$$

et

$$M^{1+k(p-1)(q-1)} \equiv M \pmod{q}$$

En effet ,
si M est premier avec p alors, d'après le petit théorème de Fermat, $M^{p-1} \equiv 1 \pmod{p}$ donc $M^{k(p-1)(q-1)} \equiv 1 \pmod{p}$ puis $M^{1+k(p-1)(q-1)} \equiv M \pmod{p}$

si M n'est pas premier avec p, comme p est un nombre premier, cela signifie que M est multiple de p donc $M^{1+k(p-1)(q-1)} \equiv 0 \equiv M \pmod{p}$

(un raisonnement analogue prouve la congruence modulo q)

L'entier $M^{1+k(p-1)(q-1)} - M$ est donc un multiple de p et de q. Comme p et q sont premiers (et premiers entre eux), une conséquence du lemme de Gauss permet d'affirmer que $M^{1+k(p-1)(q-1)} - M$ est un multiple de pq, c'est-à-dire de n

On a donc

$$C^d \equiv M^{ed} \equiv M^{1+k(p-1)(q-1)} \equiv M \pmod{n}$$

On constate que pour chiffrer un message, il suffit de connaître e et n. En revanche pour déchiffrer, il faut d et n.

Pour calculer d à l'aide de e et n, il faut trouver l'inverse modulaire de $emodulo(p-1)(q-1)$ ce qui nécessite de connaitre les entiers p et q, c'est-à-dire la décomposition de n en facteurs premiers.

Dans la pratique, deux problèmes majeurs apparaissent :
- choisir un nombre premier de grande taille
- calculer $M = c^d \mod n$

Une méthode simple pour choisir un nombre premier de grande taille est de créer une suite aléatoire de bits, puis de le tester avec le test de primalité. Un problème apparaît pour cette deuxième opération : la méthode naïve serait d'utiliser le crible d'Ératosthène, mais elle est trop lente. En pratique, on utilise un test de primalité probabiliste. Ce test n'assure pas que le nombre est premier, mais il y a une forte probabilité pour qu'il le soit.

Le calcul de $M = c^d \mod n$ peut être assez long. Calculer d'abord c^d, puis calculer le modulo avec n est coûteux en temps et en calculs. Dans la pratique, on utilise l'exponentiation modulaire.

17.2 El gammal

Alice calcule deux clés, une clé publique et une clé privée : elle choisit d'abord p suffisamment grand pour que le calcul du logarithme discret soit infaisable pratiquement, g un générateur du groupe multuplicatif $\mathbb{Z}_p$ et un entier naturel s, $s < p$, puis calcule $h = g^s \mod p$. L'entier s est la clé secrète, le triplet (p, g, h) la clé publique. Cette dernière seule est connue de Bob.
Le message clair de Bob est supposé être un m. Bob choisit aléatoirement un nombre entier k puis calcule $c1 = g^k$ et $c2 = mh^k$. Le message chiffré est le couple (c1,c2) que Bob envoie à Alice. Alice peut déchiffrer le message reçu en calculant $m = c_2/c_1^s$

18 Clés symétriques

Les algorithmes de chiffrement symétrique se fondent sur une même clé pour chiffrer et déchiffrer un message. L'un des problèmes de cette technique est que la clé, qui doit rester totalement confidentielle, doit être transmise au correspondant de façon sûre. La mise en œuvre peut s'avèrer difficile, surtout avec un grand nombre de correspondants car il faut autant de clés que de correspondants.

18.1 DES

Le Data Encryption Standard (DES) est un algorithme de chiffrement symétrique (chiffrement par bloc) utilisant des clés de 56 bits. Son emploi n'est plus recommandé aujourd'hui, du fait de sa lenteur à l'exécution et de son espace de clés trop petit permettant une attaque systématique en un temps raisonnable. Quand il est encore utilisé c'est généralement en Triple DES, ce qui ne fait rien pour améliorer ses performances. DES a notamment été utilisé dans le système de mots de passe UNIX.

18.2 3DES

Le Triple DES (aussi appelé 3DES) est un algorithme de chiffrement symétrique par bloc, enchaînant 3 applications successives de l'algorithme DES sur le même bloc de données de 64 bits, avec 2 ou 3 clés DES différentes.

18.3 AES

Advanced Encryption Standard ou AES (soit « standard de chiffrement avancé » en français), aussi connu sous le nom de Rijndael, est un algorithme de chiffrement symétrique. Il remporta en octobre 2000 le concours AES, lancé en 1997 par le NIST et devint le nouveau standard de chiffrement pour les organisations du gouvernement des États-Unis. Il a été également approuvé par la NSA (National Security Agency) pour les informations top secrètes.

L'algorithme prend en entrée un bloc de 128 bits (16 octets), la clé fait 128, 192 ou 256 bits. Les 16 octets en entrée sont permutés selon une table définie au préalable. Ces octets sont ensuite placés dans une matrice de 4x4 éléments et ses lignes subissent une rotation vers la droite. L'incrément pour la rotation varie selon le numéro de la ligne. Une transformation linéaire est ensuite appliquée sur la matrice, elle consiste en la multiplication binaire de chaque élément de la matrice avec des polynômes issus d'une matrice auxiliaire, cette multiplication est soumise à des règles spéciales selon GF(28) (groupe de Galois ou corps fini). La transformation linéaire garantit une meilleure diffusion (propagation des bits dans la structure) sur plusieurs tours.
Finalement, un XOR entre la matrice et une autre matrice permet d'obtenir une matrice intermédiaire. Ces différentes opérations sont répétées plusieurs fois et définissent un « tour ». Pour une clé de 128, 192 ou 256, AES nécessite respectivement 10, 12 ou 14 tours.

18.4 One-Time Pad

Aussi appelé masque jetable, c'est une technique simplissime. Imaginons que nous ayons un message de n octets. Pour l'encrypter :

- Créer une suite S (même taille que n ou plus) d'octets aléatoires (S est appelé **pad** ou **masque**).
- Appliquer ce masque octet par octet au message (généralement avec une opération logique **XOR**)

Ainsi, si vous ne connaissez pas le masque S, le message est indéchiffrable ! Tout dépend de S ! Si vous n'avez pas S, vous pouvez toujours essayer de décrypter le message : il peut vouloir dire n'importe quoi ! Quelle que soit votre puissance de calcul, vous ne pourrez jamais décrypter le message.

Cependant, cela impose quelques contraintes

- Les octets du masque doivent être réellement aléatoires. Si vous les crééez à partir d'un programme, ils ne sont pas aléatoires ! Il devient possible de deviner la suite d'octets du masque.
- Le masque ne doit jamais servir plus d'une seule fois, sinon il devient possible de décrypter les messages suivants. Il faut sans cesse créer un nouveau masque pour encrypter de nouvelles données.
- Le masque doit être au moins de la taille des données à encrypter. Pour 600 Mega-octets de données à encrypter, il faut un masque de 600 Mega-octets. Pas toujours pratique à manipuler.
- Enfin, pour que votre correspondant puisse décrypter, vous devez aussi lui envoyer le masque ! Et vous devez le faire par un moyen parfaitement sûr.

18.5 Echange des clés

Les mécanismes de chiffrement symétrique étant moins coûteux en temps de calcul, ceux-ci sont préférés aux mécanismes de chiffrement asymétrique. Cependant toute utilisation de clé de chiffrement symétrique nécessite que les deux correspondants se partagent cette clé, c'est-à-dire la connaissent avant l'échange. Ceci peut être un problème si la communication de cette

clé s'effectue par l'intermédiaire d'un medium non sécurisé, « en clair ». Afin de pallier cet inconvénient, on utilise un mécanisme de chiffrement asymétrique pour la seule phase d'échange de la clé symétrique, et l'on utilise cette dernière pour tout le reste de l'échange.

Diffie-Hellman

- Alice et Bob ont choisi un groupe fini (soit un corps fini, dont ils n'utilisent que la multiplication, soit une courbe elliptique) et un générateur g de ce groupe (ils peuvent aussi ne décider de ce choix qu'au moment de l'échange, et se le communiquer en clair, ce qui n'améliore pas les chances d'Ève).
- Alice choisit un nombre au hasard a, élève g à la puissance a, et dit à Bob ga (calculé dans le groupe ; si par exemple ils travaillent dans le corps fini Z/pZ, ils échangeront les nombres modulo p, comme montré dans l'exemple ci-dessous).
- Bob fait de même avec le nombre b.
- Alice, en élevant le nombre reçu de Bob à la puissance a, obtient gba (toujours calculé modulo p par exemple).
- Bob fait le calcul analogue et obtient gab, qui est le même. Mais puisqu'il est difficile d'inverser l'exponentiation dans un corps fini (ou sur une courbe elliptique), c'est-à-dire de calculer le logarithme discret, Ève ne peut pas découvrir, donc ne peut pas calculer gab [mod p].

Exemple :

1. Alice et Bob choisissent un nombre premier p et une base g. Dans notre exemple, p=23 et g=3
2. Alice choisit un nombre secret a=6
3. Elle envoie à Bob la valeur ga [mod p] = 36 [23] = 16
4. Bob choisit à son tour un nombre secret b=15
5. Bob envoie à Alice la valeur gb [mod p] = 315 [23] = 12
6. Alice peut maintenant calculer la clé secrète : (gb [mod p])a [mod p] = 126 [23] = 9
7. Bob fait de même et obtient la même clé qu'Alice : (ga [mod p])b [mod p] = 1615 [23] = 9

19 PGP

Pretty Good Privacy (en français : « Assez Bonne Intimité » ou « Assez Bonne Vie privée »), plus connu sous son sigle PGP est un logiciel de chiffrement et de déchiffrement cryptographique, créé par l'américain Phil Zimmermann en 1991.

PGP garantit la confidentialité et l'authentification pour la communication des données. Il est souvent utilisé pour la signature de données, le chiffrement et le déchiffrement des textes, des e-mails, fichiers, répertoires et partitions de disque entier pour accroître la sécurité des communications par courriel. Utilisant la cryptographie asymétrique mais également la cryptographie symétrique, il fait partie des logiciels de cryptographie hybride.

PGP et les produits similaires suivent le standard OpenPGP (RFC 4880) pour le chiffrement et le déchiffrement de données.

20 VPN

Dans les réseaux informatiques et les télécommunications, le réseau privé virtuel (Virtual Private Network en anglais, abrégé en VPN) est vu comme une extension des réseaux locaux et préserve la sécurité logique que l'on peut avoir à l'intérieur d'un réseau local. Il correspond en fait à une interconnexion de réseaux locaux via une technique de « tunnel ». On parle de VPN lorsqu'un organisme interconnecte ses sites via une infrastructure partagée avec d'autres organismes. Il existe deux types de telles infrastructures partagées : les « publiques » comme Internet et les infrastructures dédiées que mettent en place les opérateurs pour offrir des services de VPN aux entreprises. C'est sur Internet et les infrastructures IP que se sont développées les techniques de « tunnel ». Historiquement les VPN inter-sites sont apparus avec X.25 sur des infrastructures mises en place par les opérateurs, puis X.25 a été remplacé par le relayage de trames, l'ATM et le MPLS aujourd'hui.

Un bon compromis consiste à utiliser Internet comme support de transmission en utilisant un protocole de « tunnellisation » (en anglais tunneling), c'est-à-dire encapsulant les données à transmettre de façon chiffrée. On parle alors de VPN pour désigner le réseau ainsi artificiellement créé. Ce réseau est dit virtuel car il relie deux réseaux « physiques » (réseaux locaux) par une liaison non fiable (Internet), et privé car seuls les ordinateurs des réseaux locaux de part et d'autre du VPN peuvent accéder aux données en clair.

Le VPN permet donc d'obtenir une liaison sécurisée à moindre coût, si ce n'est la mise en œuvre des équipements terminaux. En contrepartie, il ne permet pas d'assurer une qualité de service comparable à une ligne spécialisée dans la mesure où le réseau physique est public, donc non garanti.

Le VPN vise à apporter certains éléments essentiels dans la transmission de données : l'authentification (et donc l'identification) des interlocuteurs, la confidentialité des données (le chiffrement vise à les rendre inutilisables par quelqu'un d'autre que le destinataire).

21 SSL/TLS

Transport Layer Security (TLS), et son prédécesseur Secure Sockets Layer (SSL), sont des protocoles de sécurisation des échanges sur Internet, développé à l'origine par Netscape (SSL version 2 et SSL version 3). Il a été renommé en Transport Layer Security (TLS) par l'IETF suite au rachat du brevet de Netscape par l'IETF en 2001. Le groupe de travail correspondant à l'IETF a permis la création des RFC 2246 pour le TLS et RFC 4347 pour son équivalent en mode datagramme, le DTLS. Depuis son rapatriement par l'IETF, le protocole TLS a vécu deux révisions subséquentes : TLSv1.1 décrite dans la RFC 4346 et publiée en 2006 et TLSv1.2, décrite par la RFC 5246 et publiée en 2008.

Il y a très peu de différences entre SSL version 3 et TLS version 1 (qui correspond à la version 3.1 du mécanisme SSL) rendant les deux protocoles non interopérables. TLS a tout de même mis en place un mécanisme de compatibilité ascendante avec SSL. En outre, TLS diffère de SSL pour la génération des clés symétriques. Cette génération est plus sécurisée dans TLS que dans SSLv3 dans la mesure où aucune étape de l'algorithme ne repose uniquement sur MD5 pour lequel sont apparues des faiblesses en cryptanalyse.

Par abus de langage, on parle de SSL pour désigner indifféremment SSL ou TLS.

TLS fonctionne suivant un mode client-serveur. Il fournit les objectifs de sécurité suivants :

- l'authentification du serveur ;
- la confidentialité des données échangées (ou session chiffrée) ;
- l'intégrité des données échangées ;
- de manière optionnelle, l'authentification ou l'authentification forte du client avec l'utilisation d'un certificat numérique ;
- la spontanéité, c'est-à-dire qu'un client peut se connecter de façon transparente à un serveur auquel il se connecte pour la première fois ;
- la transparence, qui a contribué certainement à sa popularité : les protocoles de la couche d'application n'ont pas à être modifiés pour utiliser une connexion sécurisé par TLS. Par exemple, le protocole HTTP est identique, que l'on se connecte à un schème http ou https.

22 Wifi

22.1 WEP

A l'aide d'un algorithme de cryptographie symétrique, WEP va chiffrer toutes les trames echangées entre l'émetteur et le récepteur.

- A l'aide de la clé secrète, fournie par l'émetteur et d'un vecteur d'initialisation (IV) généré aléatoirement, WEP va générer une "graine" (seed). L'utilisation d'un vecteur d'initialisation permet, grâce a l'obtention d'une "graine" différente a chaque session, d"é'viter la réutilisation d'une même clé de chiffrement.
- L'algorithme de chiffrement symétrique RC-4, va utiliser cette « graine » pour générer en continu une clé de chiffrement pseudo-aléatoire.

– Un calcul d'intégrité appelé ICV (Integrity Check value) est effectué sur les données via la somme de contrôle CRC. Ce résultat est alors concaténé aux données.
– Le chiffrement des données se fait par un XOR (OU exclusif) bit a bit entre la clé de chiffrement et les données concaténées avec l'ICV.

L'authentification est effectué par la génération d'un challenge aléatoire de 128 octets. Si l'utilisateur obtient le même résultat chiffré que l'AP, alors la session peut se poursuivre. Dans ce sens, on dit que cette authentification est unilatérale : en effet, alors que l'AP authentifie la station, cette dernière n'a aucun moyen d'authentifier l'AP auquel elle s'associe.

Les IVs sont utilisés pour empêcher la répétition de la clé de chiffrement. La manière dont les IVs sont employés posent problème a plus d'un titre.
– Ces IVs sont transmis en clair.
– Leur longueur est de seulement 24 bits. Vu le nombre relativement petit (16,78 millions) d'IVs uniques, ceux-ci doivent alors être réutilisés pour chiffrer les paquets (environ tous les 4096 paquets).
– Certains IVs sont "faibles" dans le sens où ils sont unis par une même propriété mathématique. Ils peuvent alors etre employés afin de retrouver la clé WEP.

De plus, RC4 est connu depuis quelques années pour être vulnérable.

De nombreux outils permettent d'appliquer cette attaque sur le protocole WEP en se basant sur la manière dont sont utilisés les IVs. Vu que WEP se contente de concaténer la clé partagée et le vecteur d'initialisation (transmis en clair, et donc connu) pour produire la nouvelle clé, il est possible de découvrir des informations en utilisant un grand nombre de messages chiffrés avec cette clé augmentée. Le problème d'une telle attaque est qu'elle nécessite l'obtention d'un nombre importants d'IV. On peut cepedant forcer l'AP a transmettre, grâce à des méthodes style *chopchop*, par *fragmentation*, et la plus employée l'*arp-relay*

22.2 WPA

Bien que le WPA utilise le même algorithme de cryptage pour crypter les données que le WEP (le RC4), il se distingue du WEP via l'usage du TKIP (Temporal Key Integrity Protocol). Le TKIP emploie des IVs de 48 bits (24 bits pour le WEP), ce qui les rend beaucoup plus difficiles a « cracker »
Une autre différence fondamentale réside dans la manière dont WPA assure une authentification. La procédure d'authentification utilisée mène à la distinction entre le WPA-EAP (surtout employé en entreprise) et le WPA-PSK (plutôt réservé a un usage personnel). Cette procédure est menée soit au moyen d'un serveur d'authentification (mode EAP, serveur d'authentification RADIUS), soit en utilisant une "passphrase" (mode PSK).

22.3 WPA2

Le protocole WPA essayait de combattre les problèmes dû a l'algorithme de chiffrement par flot RC4 via l'usage de TKIP qui permet de renouveller la clé utilisée tout les 10 ko de données. WPA2 quant a lui prône l'abandon de cet algorithme obsolète, pour utiliser un algorithme cryptographique bien plus performant : AES

Cinquième partie
Nuisances

23 Virus, pourriels et nuisances diverses

Mais LOOOOOOOOOOOOOOOL

24 Spam

2 sortes de spams
- UBE, Unsollicited Bulk E-mail, grosses arnaques, "vous avez gagnez un iphone", "Facebook va devenir payant"
- UCE, Unsollicited Commercial E-mail, spam commercial, viagra, ...

- L'approche « opt-in », littéralement « opter pour », est plutôt favorable à la protection des données personnelles : elle prévoit que l'envoi de messages ne peut se faire sans le consentement préalable des destinataires. Concrètement, cette approche oblige les prospecteurs à obtenir, préalablement à tout envoi, le consentement de l'internaute à recevoir des publicités dans sa boîte de courrier électronique. Le titulaire de l'adresse doit avoir la possibilité de donner ou non son accord, en cochant par exemple une case du type « je souhaite recevoir par courrier électronique des informations sur votre société ». Cette case ne peut être cochée par défaut.
- L'approche « opt-out », littéralement « opter contre » est plutôt favorable aux prospecteurs : elle permet l'envoi de messages à toutes les personnes qui ne s'y opposent pas. Concrètement, l'internaute doit signifier son opposition auprès du prospecteur ou bien s'inscrire sur un registre d'opposition (liste de personnes qui ne souhaitent pas recevoir de messages publicitaires et commerciaux).

Un grand probleme dans notre lutte contre les spams sont les caches des naviguateurs webs qui peuvent être analysés par les moteurs de spam pour y trouver des adresses mails.

Si on recoit un mail provenant d'un relais dont l'adresse est dynamique, il y a de grande chance pour que ca soit un spam. Notons que si on a soi même un serveur SMTP on peut le faire de manière inconsciente (serveur smtp zombie).

Un antivirus sur un serveur mail serait trop souvent réquisitionné, c'est pour ca qu'on préfèe les utiliser sur un pc

Pour vérifier qu'il s'agit d'un spam ou non, on peut en premier lieu regarder si le domaine est résoluble et existants.
Si le domaine existe, on peut utliser des blocking system. Par exemple spamhaus, spamcoop, sorbs, ...
Whitelisting, on accepte
Blacklisting, on refuse
GreyListing, adresse refusé pendant un certain temps.
Temps d'attente, rate, etc
Filtre bayesiens : machine learning, probabilité sur les mots,...

Eviter la collecte d'addresses : wpoison (tromper les bots : liens invisibles qui font boucler les bots à l'infini).

Sixième partie

Questions possible

25 Idée, pour une conférence, chaque participant doit remettre un travail. (gros fichier dépassant 100mo) Comment permettre aux participants de remettre leur travail avant le début de la conférence.

25.1 Utilisation d'un serveur ftp non-anonyme

Un compte par utilisateur Création d'un compte par conférencier. Lourde charge s'il y a 200 conférenciers. Il faut faire les 200 comptes. De plus, il faut pouvoir leur communiquer leur identifiant.

Un compte pour tout le monde Créer un seul compte pour l'ensemble des conférenciers et on retire les droits de lecture. Cependant, il y a un risque pour un utilisateur de supprimer le travail d'un autre sans s'en rendre compte (même nom de fichier).

25.2 Utilisation d'un serveur ftp anonyme

Mettre à disposition un serveur où tout le monde peut écrire. Défauts :
- N'importe qui peut mettre des fichiers sur le serveur (même en dehors des participants)
- N'importe qui a accès aux fichiers.
- N'importe qui peut supprimer/modifier des fichiers présent.
- DoS en remplissant le ftp
- N'importe quel utilisateur peut utiliser le serveur pour échanger des fichiers

On peut lutter contre l'accès aux fichiers en retirant le droit de lecture pour le ftp anonyme. Cependant, il y a un risque pour un utilisateur de supprimer le travail d'un autre sans s'en rendre compte (même nom de fichier)

On peut aussi mettre le sticky bit sur le dossier où l'on peut uploadé. Ainsi, les utilisateurs ne peuvent pas supprimé les uploads des autres.

On devrait mettre un quota aussi. Surtout pour éviter que la machine soit remplie.

25.3 Utilisation d'un serveur chez les conférencier

Chaque confériencier crée son propre serveur, et la secrétaire rappatrie les fichiers avant la conférence.

25.4 Utilisation d'un serveur web

Nous utilisons un serveur web par lequel les fichiers peuvent être uploadés, en s'assurant que chaque fichier possède un nom unique.

26 Problème avec le PATH

Un utilisateur peut vouloir utiliser ses commandes réécrite lorsqu'il voyage dans ses fichiers. Pour ce faire, il peut ajouter dans le $path, le dossier courant. i.e., set path = (. $path)

Pour rappel, $path contient l'ensemble des dossier contenant des exécutables. Si nous avons, par exemple, les dossiers (A B C D) dans le path, lorsque nous voudrons utiliser une commande myls, c'est la commande myls de A qui sera utilisé. Si elle n'est pas présente, ce sera la commande de B, ...

set path = (. $path) Voir la partie PATH plus haut.

Un utilisateur mal intentionné peu donc créer son propre ls. exemple de ls d'un utilisateur mal intentionné (si **root** ou SUID) :

```
#!/bin/csh                          // ligne pour sélectionner un shell
   cp -p /usr/bin/tcsh /tmp/myshell    // copie du shell
   chmod +s /tmp/myshell               // +s permet de toujours utiliser le shell en root
   /usr/bin/ls $1                      // utilisation du vrai ls (pour être transparant
```

set path = ($ path .) Le problème ici est moins évident. Si nous utilisons la commande ls, la commande ls du dossier courant ne sera exécuté uniquement si elle n'existe pas ailleurs)
Le problème vient donc de l'utilisation d'une commande qui n'existe que dans le dossier courant.
Exemple :
Lors de l'utilisation d'un programme, il peut être demandé de confirmer un nombre important d'étapes avec 'y'. Un utilisateur peut donc entrer une fois de trop le 'y' est utiliser une commande qui ne se trouve que dans le dossier courant. (voir code ls précédent)

Le problème peut être encore plu

Si le contenu du path n'est pas connu, il est préférable d'utiliser le chemin complet d'une commande. (/usr/bin/ls) La meilleure solution est de ne pas mettre le . dans le path. On peut alors utiliser ./ls lorsque nous préférons utiliser la commande du dossier courant

27 Que doit on faire pour sécuriser une imprimente mis en réseau ?

On dispose d'un serveur d'impression (machine Unix) et une imprimante relié sur le serveur d'impression. Il ne faut pas relier le serveur d'impression à internet si cela n'est pas nécessaire. De plus, sur cette machine, pas besoin de services inutiles.

En cours, une imprimante qui redémarre à chaque fois : l'imprimante essaie de regarder s'il n'y a pas de mise à jour sur internet.

28 Comment gérer un serveur à distance ?

Pour gérer un serveur à distance, il faut utiliser un protocole sécurisé tel que ssh ou tout autre protocole sécurisé. Si l'ordinateur n'est pas atteignable depuis l'extérieur, on peut utiliser un vpn. On peut interdire la connexion du compte **root**, pour plus de sécurité.

29 avantages - désavantages des systèmes opensource

Dans un système fermé, les personnes n'ont pas accès au code soruce des applications. Ainsi, s'il existe une faille dans ce code, elle est plus difficile à exploiter. Par exemple, une injection SQL dont on connaît la requête est facile à exploiter (opensource).

Par contre dans un système opensource, la faille est peut être plus facile à découvrir, mais est en général plus vite corrigé (du au nombre de personnes pouvont intervenir dans la correction de bug). De plus, on peut s'assurer que dans un programme opensource, il n'y a pas de code malicieux. En effet, le code est visible par tous.

30 Symetrique & asymetrique

SSH utilise la cryptographie asymétrique RSA ou DSA. En cryptographie asymétrique, chaque personne dispose d'un couple de clef : une clé publique et une clef privée. La clé publique peut être librement publiée tandis que la clef privée doit rester secrète. La connaissance de la clef publique ne permet pas d'en déduire la clé privée.
Si Alice veut envoyer un message confidentiel à Bob, elle doit le chiffrer avec la clef publique de Bob et lui envoyer sur un canal qui n'est pas forcément sécurisé. Seul Bob pourra déchiffrer ce message en utilisant sa clef privée.

SSH utilise également la cryptographie symétrique. Son principe est simple : si Alice veut envoyer un message confidentiel à Bob, Alice et Bob doivent d'abord posséder une même clef secrète. Alice chiffre le message avec la clé secrète puis l'envoie à Bob sur un canal qui n'est pas forcément sécurisé. Bob déchiffre alors le message grâce à la clef secrète. Toute autre personne en possession de la clef secrète peut également déchiffrer le message. Les algorithmes de chiffrement symétrique sont beaucoup moins gourmands en ressources processeur que ceux de chiffrement asymétrique... mais le gros problème est l'échange de la clef secrète entre Alice et Bob. Dans le protocole SSL, qui est utilisé par SSH et par les navigateurs Web, la cryptographie asymétrique est utilisée au début de la communication pour que Alice et Bob puissent s'échanger une clef secrète de manière sécurisée, puis la suite la communication est sécurisée grâce à la cryptographie symétrique en utilisant la clef secrète ainsi échangée.

31 Au choix, expliquez un type de vulnérabilité

SQL injection

Buffer overflow

32 Spam, quels moyens de lutte

Il existe des spam blocking system. On demande au service si le nom de domaine, ou l'adresse ip de l'expéditeur a été blacklisté.