Systèmes d'exploitation

Gestion de la sécurité

Pilot Systems - Gaël Le Mignot

INSIA SRT - 2007

Table des matières

Table des figures

Chapitre 1

Gestion de la sécurité

1.1 Les types de sécurité et de menaces

1.1.1 Les objectifs de la sécurité

La sécurité est un concept vaste, qui peut couvrir un grand nombre de problèmes et de techniques.
Les objectifs de la sécurité peuvent être divisés en grandes catégories comme suit :

1. Protéger les données contre une consultation non autorisée (confidentialité) ;
2. Protéger les données contre une modification ou destruction non autorisée (intégrité) ;
3. Protéger le système lui-même contre une tentative de le rendre inutilisable (disponibilité).

Les deux premiers objectifs sont assez simples à mettre en œuvre de manière théorique (même si, dans la pratique, les choses sont moins idylliques) mais par contre, la disponibilité est un problème complexe même en théorie. Un déni de service est très facile à réaliser, à partir du moment où on possède une bande passante et/ou une puissance de calcul suffisante.

Une autre problématique liée est la protection de la vie privée, qui peut être menacée par l'existence (et surtout l'utilisation) de bases de données toujours plus grandes et complètes. Mais c'est une problématique qui est plus politique (ou éthique) que technique, et ne rentre donc pas dans le cadre de ce cours.

1.1.2 Les types de menace

Parallèlement, la sécurité peut aussi être classifiée selon les types de menace :

1. Accident au niveau matériel (panne, incendie, ...) ;
2. Volonté d'enfreindre la politique de sécurité de la part d'un attaquant interne (déjà autorisé à utiliser le système) ;
3. Volonté d'enfreindre la politique de sécurité de la part d'un attaquant externe ;
4. Mauvaise manipulation de la part d'un utilisateur.

Le cas 4 (mauvaise manipulation de la part d'un utilisateur) est un peu différent, puisqu'il s'agit d'une opération que l'utilisateur est habilité à faire dans l'absolu. Il est en général considéré comme étant extérieur à la sécurité.

Dans les cas 2 et 3, il faut bien distinguer les niveaux d'attaque, un utilisateur non technique étant juste un peu curieux ne présente pas le même niveau de menace qu'une équipe de professionnelle payée par un concurrent pour récupérer des données confidentielles (espionnage politique ou industriel).

Enfin, il ne faut pas oublier qu'à côté de la sécurité informatique, la sécurité est aussi beaucoup un phénomène social et physique. Pour obtenir un mot de passe, on peut essayer de le trouver par des moyens techniques, mais on peut aussi l'obtenir d'un utilisateur en utilisant différentes techniques (plus ou moins légales).

1.2 Le concept de *Trusted Computing Base*

Un concept indispensable à comprendre en termes de sécurité est celui de la TCB, *« Trusted Computed Base »*. Il s'agit de l'ensemble des composants, matériels et logiciels, auxquels on doit faire confiance afin d'effectuer des garanties au niveau de la sécurité.

Typiquement dans un système Unix, la TCB pour les opérations usuelles est constituées du matériel (éventuellement une partie), du noyau, des programmes fonctionnant en super-utilisateur ainsi que des programmes “SUID root”.

Une faille de sécurité (c'est à dire la possibilité pour un utilisateur (interne ou externe) d'effectuer quelque chose qu'il n'a pas le droit de faire) ne peut provenir que d'un défaut de l'un des composants de la TCB (si celle-ci a bien été délimitée).

Cette TCB dépend de l'impératif de sécurité considéré. Par exemple, sur une machine hébergeant un serveur MySQL ainsi qu'un site web purement statique, la TCB pour la protection des données de la base SQL ne sera pas la même que celle pour la protection des données du site web (l'une contiendra le serveur SQL et l'autre le serveur Web, mais toutes les données contiendront le noyau, par exemple).

L'un des points importants en termes de sécurité est donc de minimiser la taille de la TCB, par exemple, en exécutant les serveurs sans les privilèges du super-utilisateur.

Réduire la TCB est aussi l'un des objectifs des systèmes à base de micro-noyaux.

1.3 L'authentification des utilisateurs

1.3.1 Problématique

Afin de contrôler la sécurité, il est possible d'effectuer des vérifications pour chaque opération. Par exemple, chaque fichier peut être protégé par un mot de passe, et ce mot de passe demandé à chaque accès au fichier.

En pratique, si ces formes de protection peuvent parfois être utilisées (les fichiers protégés par des mots de passe existent dans certaines applications), elles sont très fastidieuses.

Le mode de fonctionnement usuel est d'identifier la personne utilisant le système, et d'avoir des droits définis par utilisateur. La problématique est alors de trouver une méthode permettant d'identifier l'utilisateur qui soit à la fois fiable et pas trop contraignante.

1.3.2 Solutions possibles

Il y trois moyens d'identifier un utilisateur, que nous allons présenter rapidement.

L'identification sur ce que l'utilisateur connaît

La forme d'identification la plus simple est le fameux couple identifiant/mot de passe. Cette méthode d'identification peut avoir des variantes, mais repose toujours sur ce que l'utilisateur connaît.

La faille principale de cette méthode consiste en un mauvais choix du mot de passe (un mot du dictionnaire par exemple), rendant les attaques par force brute (essayer massivement des possibilités) ou des variantes possibles.

Si le mot de passe est bien choisi, elle est plutôt fiable, mais elle présente le risque d'oubli de la part de l'utilisateur, et le risque associé qui est l'écriture du mot de passe sur un papier.

Elle présente aussi des risques d'interception du mot de passe, que ce soit en écoutant sur un réseau, en regardant l'utilisateur taper, ...

Une variante consiste à ne pas utiliser un mot de passe, mais des questions auxquelles l'utilisateur est censé connaître la réponse. Avec suffisamment de questions, on peut diminuer le risque que le mot de passe ait pu avoir été lu. Cependant, les questions du type "quel est le nom de votre chien ?" ou "dans quelle rue était votre école primaire ?" ne résistent pas bien longtemps à un attaquant connaissant l'utilisateur, ou ayant réalisé une enquête.

L'identification sur ce que l'utilisateur possède

Ces méthodes reposent sur l'utilisation d'un objet particulier possédé par l'utilisateur. C'est le cas des clés usuelles (non informatiques, mais qui peuvent protéger la salle où se trouvent les serveurs par exemple), mais aussi de techniques comme des clés USB à insérer dans l'ordinateur pour avoir accès au service, ou bien les codes de protection de logiciels du type "quel est le 3ème mot de la 5ème page du manuel ?".

Ces méthodes sont en général combinées avec les méthodes sur ce que l'utilisateur connaît (par exemple, une clé USB et un mot de passe sont nécessaires), car sinon elles sont très vulnérables à la perte de l'objet en question.

L'utilisation combinée d'un objet et d'un code est par exemple utilisée par les distributeurs automatiques de billets de banque.

L'identification sur ce que l'utilisateur est

Ces méthodes, nommées parfois biométrie, utilisent des caractéristiques physiques de l'utilisateur (empreinte digitale ou des rétines, reconnaissance de la voix, ...).

Elles sont très fiables, mais assez contraignantes, et nécessitant de l'équipement parfois lourd. Leur généralisation pose aussi des risques sur le respect de la vie privée, qui sortent du cadre de ce cours, mais qu'il ne faut pas négliger non plus.

Ces techniques sont en général utilisées uniquement dans les environnement de haute sécurité.

1.4 Comment contrer les attaques ?

1.4.1 Les accidents

La seule protection réelle contre les accidents consiste à effectuer des sauvegardes, à intervalles réguliers ou en temps réel, et si possible dans un endroit géographiquement distant.

Des techniques comme le RAID permettent de protéger les données contre certaines pannes, mais pas contre toutes.

Les sauvegardes peuvent être réalisées à chaud ou à froid, c'est à dire, sans interruption de service, ou avec interruption de service. Effectuer une sauvegarde à chaud, sans risquer d'avoir des données incohérentes, requiert en général une coopération étroite avec les logiciels (par exemple, avec le serveur SQL pour une base de données).

1.4.2 Les attaques de l'intérieur du système

La plupart des attaques proviennent d'un utilisateur légitime du système, tentant d'effectuer une action qu'il n'est pas autorisée à faire.

Nous allons voir les principaux types d'attaques pour un utilisateur ayant un accès au système, et quelques solutions possibles.

Les chevaux de Troie

Les chevaux de Troie sont des programmes qui ont l'air d'effectuer une tâche anodine mais qui en réalité ne le sont pas.

Par exemple, sur un système Unix, un utilisateur peut créer un programme s'appelant "su" et enregistrant le mot de passe saisi dans un fichier. Il modifie son `PATH` pour que ce programme soit prioritaire, et appelle un administrateur système en prétextant une panne. Si l'administrateur tente de passer `root` sans prendre de précautions, l'utilisateur aura le mot de passe. La même chose est possible sur des systèmes graphiques en simulant l'écran de login, par exemple.

La réponse à ces attaques en général purement humaine : l'administrateur doit prendre ses précautions. Sur certains systèmes, comme GNU/Linux ou Windows, il existe une combinaison de touches permettant d'amener de manière sécurisée un écran de login.

1.4.3 Les attaques de l'extérieur du système

Virus et vers

Un virus est un petit morceau de programme, qui contamine (modifie) un autre programme et lui fait réaliser des actions diverses, dont l'une d'entre elles est toujours d'infecter d'autres programmes. Exactement comme un virus en biologie, qui infecte des cellules et les force à produire des copies du virus. Un virus se propage d'ordinateur en ordinateur lorsque l'on copie des programmes contaminés (ou éventuellement, lorsqu'on transfère des documents contaminés, pour des virus s'attaquant à des logiciels comme Word via les documents).

Un ver un similaire à un virus, mais il se propage tout seul à travers le réseau, par exemple en s'envoyant automatiquement par e-mail, ou en attaquant tous les serveurs HTTP qu'il trouve.

La première solution pour se protéger contre les virus est, comme bien souvent, quelques précautions de bon sens : ne pas exécuter de programmes dans lesquels on n'a pas confiance, ne jamais faire en administrateur ce qu'on peut faire en tant que simple utilisateur, ne pas utiliser de logiciels fortement vulnérables.

Un autre point important sur les virus et les vers est qu'ils ne se reproduisent qu'en terrain homogène : l'omniprésence d'un seul système (Windows) et d'une seule architecture (ia32) rend beaucoup plus simple la propagation des virus et des vers, tandis que lorsque l'environnement est divers (plusieurs systèmes, plusieurs architectures matérielles, ...) la propagation est beaucoup plus difficile.

Il existe des logiciels anti-virus qui tentent de détecter et de neutraliser les virus, mais leur fiabilité n'est que partielle.

1.4.4 Les attaques applicables dans les deux cas

Les portes dérobées

Une porte dérobée est un morceau de code ou un programme laissé par un programmeur ou un administrateur qui lui permet d'obtenir des droits sur le système. L'utilisation typique est celle d'un administrateur qui souhaite garder les accès sur une machine même après son départ de la société, ou alors un programmeur qui souhaite pouvoir accéder aux données manipulées par son programme a posteriori.

La seule garantie réelle contre ce type d'attaques est le *peer review*, c'est à dire, de faire vérifier ce qui est fait par d'autres personnes. Dans le cadre d'un logiciel utilisé en interne, il faut faire relire le code par un autre programmeur. Dans le cadre d'un administrateur, il faudrait surveiller chacune de ses actions, ce qui serait très difficile. Dans le cadre d'un logiciel utilisé par une tierce partie, seule la disponibilité du code source permet de vérifier l'absence de porte dérobée.

Les failles de programmation

Les failles de programmation sont des erreurs commises par un programmeur, et qui permettent à un attaquant de forcer un programme à effectuer des actions différentes de ce pourquoi il a été conçu.

La plus connue est le *buffer overflow* qui consiste à écrire dans une variable plus de données qu'elle ne peut contenir. Les données supplémentaires vont en écraser d'autres (par exemple l'adresse de retour de la fonction), et avec des données judicieusement choisies, on peut forcer le programme à agir comme on le souhaite. Par exemple, le programme `passwd` d'Unix permet à un utilisateur de changer son mot de passe. Il possède les droits du super-utilisateur, car bien sûr un utilisateur n'est normalement pas autorisé à modifier le fichier contenant les mots de passe. Si ce programme contient un *buffer overflow*, il est possible de lui faire effectuer autre chose (changer le mot de passe d'un autre utilisateur, ou donner un shell root, par exemple).

D'autres attaques du même type existent.

Il est possible de rendre plus complexe l'utilisation de ces attaques en interdisant, par exemple, l'exécution de la pile via la VM. Mais la seule véritable solution consiste à faire attention lors de l'écriture des programmes, et de faire des audits/relecture des différents programmes qu'on utilise en environnement critique.

1.5 Les mécanismes de protection

1.5.1 Les domaines de protection

Les domaines de protection est le mode de fonctionnement type des Unix. Un processus appartient à un utilisateur et à un ou plusieurs groupes, qui définissent un domaine. Dans chaque domaine, les objets du système (les fichiers, les segments de mémoire partagés, les appels systèmes, les ports TCP, ...) possèdent des droits, sous la forme d'une matrice.

1.5.2 Les ACLs

Les ACLs associent à chaque objet du système une liste d'utilisateurs (ou éventuellement de groupes) autorisés à y accéder et de quelle manière. Ils permettent un contrôle plus fin, mais au prix d'une complexité et donc d'un risque d'erreurs plus élevés. C'est ce qui est utilisé par Windows pour les accès aux fichiers (et disponible sous forme d'extensions pour la plupart des Unix).

1.5.3 Les capacités

La notion de capacités est la notion inverse à celle d'ACLs, elle consiste à donner à des utilisateurs, ou à des processus, des droits précis. Une capacité peut être “allouer un port sous 1024”, “monter un système de fichier” ou “accéder en écriture au fichier /etc/passwd”.

Les capacités permettent de définir finement des possibilités par utilisateur (et non par ressource comme les ACLs) ou par processus, et sont utilisés par des systèmes comme GNU/Hurd.

1.6 L'utilisation de cryptographie

1.6.1 Rappels sur la cryptographie

Le but de ce cours n'étant pas de faire de la cryptographie, ce rappel sera bref.

Cryptographie symétrique

Dans la cryptographie symétrique, la même clé permet de chiffrer et de déchiffrer le message. L'avantage principal de ce mode est qu'il existe des implémentations fiables et performantes.

Cryptographie asymétriques

Dans la cryptographie asymétrique, une paire de clés est utilisée. Tout message chiffré avec une clé de la paire ne peut être déchiffré qu'avec l'autre clé de la paire, et réciproquement.

L'une des deux clés est diffusée (nommée clé publique), et l'autre est conservée précieusement (clé privée).

Hachage et signature

Une fonction de hachage converti un message de taille arbitraire en une empreinte de petite taille (16 à 128 octets en général). Cette fonction est choisi pour que le procédé inverse (obtenir un message correspondant à l'empreinte) soit difficile et que trouver des collisions (deux messages ayant la même empreinte) le soit aussi.

Si cette empreinte est chiffrée avec une clé privée, le destinataire peut vérifier, avec la clé publique, que l'empreinte correspond bien au message, c'est le concept de signature numérique.

1.6.2 Protection des communications

La première utilisation de la cryptographie est de protéger les communications. Ce sujet dépasse le cadre d'un cours sur les systèmes d'exploitations, et est presque systématiquement effectué au niveau des programmes utilisateurs. L'exception la plus courante est IPSec ou les autres formes de VPN, qui se trouvent en général au niveau de la pile IP et donc dans le système d'exploitation.

Le fonctionnement général est le suivant :

1. Le serveur génère une paire de clés (une privée, une publique).
2. Le serveur envoie la clé publique au client.
3. Le client génère une clé pour un chiffrement symétrique.
4. Le client chiffre la clé symétrique avec la clé publique qu'il a reçue.
5. Le client envoie au serveur la clé symétrique chiffrée.
6. Le serveur décrypte la clé symétrique avec sa clé privée.
7. Les deux peuvent maintenant communiquer en utilisant la clé symétrique.

Un attaquant peut intercepter la clé publique ainsi que la clé symétrique chiffrée, mais pas la clé privée, et ne peut donc pas obtenir la clé symétrique non chiffrée.

La seule attaque possible est le « *man in the middle* » c'est à dire un intermédiaire qui intercepte l'envoie de la clé publique et la remplace par sa propre clé publique. D'où l'importance d'avoir toujours la même clé publique pour le même serveur, et de faire attention lors de la première connexion.

1.6.3 Systèmes de fichiers chiffrés

Le chiffrement des données sur le disque dur permet de les protéger y compris contre des attaques physiques (insertion du disque dur dans une autre machine, par exemple).

Il y a deux manière de l'implémenter : soit au niveau du système de fichiers, où le chiffrement peut être fait fichier par fichier via un attribut spécial, soit à un niveau intermédiaire entre le driver disque et le système de fichier. Dans ce cas une partition entière est chiffrée, et la clé est demandée lors de l'accès (montage de la partition par exemple).

Le chiffrement par fichier permet plus de finesse car il permet de ne chiffrer que certains fichiers, mais le chiffrement au niveau intermédiaire a l'avantage de permettre de choisir le méthode de chiffrement et le système de fichiers de manière totalement indépendante.

1.6.4 Les programmes signés

Pour se protéger (partiellement) des virus et des chevaux de Troie, il est possible de signer les programmes avec une signature numérique, et de demander au système d'exploitation de vérifier la signature avant d'exécuter le programme. Il est aussi possible de faire une vérification régulière des signatures électroniques, afin de s'assurer que les programmes n'ont pas été corrompus.

Par exemple, sur les distributions GNU/Linux comme Debian (mais beaucoup d'autres le font), tous les paquets (contenant les logiciels et les bibliothèques, du système comme des couches au-dessus) sont signés numériquement via GnuPG, et contiennent une liste des sommes MD5 de tous les fichiers contenus dans le

paquet. Le logiciel `apt-get` effectue une vérification de la signature numérique lors de l'installation ou la mise à jour, et émet un avertissement en cas de problèmes. Il est aussi possible (via l'outil `debsums` ou à la main) de vérifier les sommes md5 a posteriori. Pour plus de sécurité, il est conseillé de faire la vérification depuis un *live cd* dans lequel on a confiance, car si le programme de vérification lui-même est corrompu, ...

1.6.5 Les DRM

L'utilisation de la cryptographie pour la sécurité peut prendre différentes formes. L'une d'entre elles est le DRM, pour « Digital Rights Management » (« gestion des droits numériques ») ou « Digital Restriction Management » (« gestion des restrictions numériques ») suivant la personne.

Les DRM consistent à chiffrer des données (musique, film, livre, ...) avec une clé (qui peut être rendue publique), afin d'interdire leur lecture par un lecteur qui ne possède pas la clé privée. La clé privée est possédée par un éditeur ou un consortium d'éditeurs, qui ne la donnent qu'aux fabriquants de lecteurs dont ils ont pu vérifier l'authenticité.

L'idée est la suivante : à côté des données chiffrées, un certains nombres de paramètres (soit dans le fichier lui-même, soit récupérés par le réseau) spécifient à quelles conditions le fichier peut être lu (uniquement sur tel ordinateur, uniquement les dimanche, affiché à l'écran mais pas imprimé, ...). Seuls les lecteurs qui appliquent ces restrictions se voient accordées la clé privée.

Le but affiché est d'éviter la copie illicite (au prix d'une interdiction de la copie licite, comme le fait de recopier une chanson qu'on a acheté sur un baladeur MP3, et de l'interdiction de lire le contenu via des logiciels libres). Les utilisations possibles sont beaucoup plus vastes : limiter le nombre de fois qu'on peut lire le contenu, sa durée dans le temps, interdire de sauter les pubs, ...

1.6.6 Le TC

Les DRM ont cependant une grosse faille : il est toujours possible d'analyser le fonctionnement d'un lecteur autorisé, et de réussir à extraire la clé privée, ou la manière de l'obtenir. En pratique, peu de méthodes de DRM ont résisté longtemps aux tentatives de contournement, que ce soit pour des utilisations licites ("DVD Jon" souhaitant lire son DVD sous GNU/Linux) ou moins licites.

La solution proposée par certains se nomme TC pour « Trusted Computing » (« informatique de confiance ») pour certains, « Treacherous Computing » (« informatique traître ») pour d'autres.

Le concept est le suivant : le système d'exploitation n'autorise l'exécution que des programmes ayant été signés par une de clé privée (ou un petit nombre de clés). Il est alors impossible d'installer de nouveaux programmes, qui pourraient contourner les utilisations autorisées du contenu. Il est aussi impossible de créer des chevaux de Troie ou des virus.

Deux modes de fonctionnement sont possibles : soit l'utilisateur reste maître en définitive (il peut forcer l'exécution d'un programme même si le système d'exploitation le déclare "non vérifié", ou alors il peut désactiver la vérification), ce qui permet de protéger contre les virus mais pas contre les utilisations illicites (copie d'un DVD par exemple).

Soit l'utilisateur n'a pas la possibilité de désactiver le TC (ce qui est le cas dans certaines consoles de jeu (xbox) ou dans l'embarqué (iphone), par exemple), et dans ce cas, la protection est totale (sauf faille de sécurité dans un programme de confiance, chose qui en pratique arrive souvent).

Mais le prix est très élevé : l'utilisateur n'est plus maître de son ordinateur, ce n'est plus lui qui peut choisir quel logiciel exécuter. Le logiciel libre ne peut plus exister. Tous les auteurs de logiciels, individuels comme sociétés, doivent se mettre d'accord avec les détenteurs des clés (le fabriquant du système d'exploitation, en général), pour simplement pouvoir continuer leur travail.

Mais nous quittons là le sujet de ce cours, la technique des systèmes d'exploitation, pour entrer dans l'éthique et la politique.

Informations légales

La version PDF et le code source LaTeX sont disponibles sur `http://insia.pilotsystems.net`.