Sécurité Informatique

Attaques Informatique

Jean-Olivier Gerphagnon*, Marcelo Portes de Albuquerque†
& Márcio Portes de Albuquerque‡

Centro Brasileiro de Pesquisas Físicas – CBPF/CNPq
Coordenação de Atividade Técnicas – CAT
Rua Dr. Xavier Sigaud 150
22290-180 Rio de Janeiro – RJ – Brazil

1 Introduction

La sécurité informatique est de nos jours devenue un problème majeur dans la gestion des réseaux d'entreprise ainsi que pour les particuliers toujours plus nombreux à se connecter à Internet. La transmission d'informations sensibles et le désir d'assurer la confidentialité de celles-ci est devenu un point primordiale dans la mise en place de réseaux informatiques. Ce document a pour but de présenter globalement la manière dont les "*Hackers*" opèrent afin de pénétrer les systèmes informatiques en espérant qu'il aide à pallier à ce type de problème de plus en plus fréquent...

2 Les mécanismes d'authentification

L'authentification est le premier rempart aux attaques informatique, il s'agit la plupart du temps du couple "nom d'utilisateur/mot de passe" (*login/password*). Cette première méthode constitue une sécurité relativement fiable lorsqu'elle est bien utilisée: mot de passe correct, confidentialité assurée, fichier protégé,... Elle pose tout de même certains problèmes comme par exemple le cas où un utilisateur a besoin de se connecter sur plusieurs stations différentes. Il va alors rapidement trouver cette méthode d'authentification relativement lourde (considérons que l'utilisateur est une personne "sérieuse" qui possède un mot de passe différent sur chaque station ou que le réseau utilise NIS (Network Information Service)).Il sera alors tenter d'utiliser une méthode moins contraignante: les services `r*`. Ces services (`rlogin, rsh, rcp`, ...) sont basés sur une authentification beaucoup moins lourde pour l'utilisateur mais également beaucoup moins sûre: l'adresse IP de la station source ainsi que le "nom d'utilisateur". Ce type d'authentification est, malgrès de nombreux problèmes de sécurité, encore très utilisé dans les réseaux locaux actuels.

Une autre méthode d'authentification permettant un niveau de sécurité plus élevé utilise un serveur d'authentification (*Authentification Server*) qui permet d'éviter les problèmes engendrés par l'authentification IP. Le serveur d'authentification reçoit une demande de connection et, afin de vérifier que la station est bien celle qu'elle prétend être, établit une seconde connection TCP avec la station source (qui ne sera donc pas sous le contrôle d'un éventuel attaquant).

*triton@softhome.net
†marcelo@cbpf.br
‡mpa@cbpf.br

Une méthode beaucoup plus sécurisée, mais plus contraignante et plus difficile à mettre en oeuvre, appellée *One-Time Password System* (OTP) permet d'éviter les problèmes de circulation des mots de passe sur le réseau (*Sniffing,...*). Se sont des pseudo mots de passe qui circulent alors sur le réseau et dont la validité est restreinte à la dite connection. Le fonctionnement des *One-Time Passwords* peut se résumer ainsi: le client demande une connection à un serveur, ce dernier renvoit un *Challenge* au client qui devra alors générer un *One-Time Password* à l'aide d'un mot de passe (ou d'une phrase) connu des deux stations ainsi que du *Challenge* reçu. Ce pseudo mot de passe sera alors retransmis au serveur qui pourra ainsi le vérifier et autoriser (ou non) la connection.

Actuellement, le programme *Secure Shell* est très apprécié, il peut apporter la facilité des authentifications par adresses IP tout en assurant une très bonne sécurité grâce à un système d'encryptage des données qui transitent sur le réseau. Il utilise pour cela un mécanisme d'encryptage assymétrique (principe des clés publiques et privées) bien connu. Il s'agit du système RSA (Rivest, Shamir, Aldeman). Cette solution peut permettre aux utilisateurs de ne pas être obligé d'entrer à chaque fois leur mot de passe tout en évitant les problèmes des services `r*` ou du `telnet` (*Sniffing*).

D'autres méthodes d'authentification existent également, comme par exemple *Kerberos* (gestion de tickets), SSL (Secure Socket Layer), S-HTTP (Secure HTTP), les signatures digitales (*Digital Signature*) et les certifications...

3 Principe de fonctionnement des *Firewalls*

En informatique, un *Firewall* est un périphérique ou un ordinateur qui protège la partie privée d'un réseau de la partie publique. C'est en réalité l'élément qui permet de distinguer la partie privée du réseau de celle que l'on nommera publique (Internet,...), lui seul peut donc en atteindre les deux extrémités. Il permet donc de protéger le réseau privé d'éventuelles attaques provenant d' Internet et peut également contrôler certaines actions effectuées de l'intérieur du réseau privé. Il y a en réalité deux principaux types de *Firewall*:

- *Firewall IP Filter* ou *Chokes* : ce type de *Firewall* travail au niveau des paquets transmis sur les réseaux, il permet de contrôler le flux de paquets en fonction de l'origine, de la destination, des ports et du type d'information contenue dans chacun d'eux. C'est un système relativement facile à mettre en place en instaurant un certains nombre de règles permettant de contrôler les paquets entrants ou sortants du réseau privé. Il s'agit soit d'un ordinateur soit d'un périphérique de communication permettant de restreindre le flux de paquets entre les réseaux.
- *Gates* : il s'agit d'un programme, d'un périphérique ou d'un ordinateur qui reçoit les connections des réseaux externes et les retransmet dans le réseau privé. Aucun utilisateur n'est autorisé à accéder à ce *Gate* pour des raisons de sécurité (seule une connection sur la console par l'administrateur est autorisée). Sur ce *Gate*, un ou plusieurs des programmes suivants doivent être utilisés:
  - *Network Client Software* : les utilisateurs doivent, dans cette configuration, avoir un compte sur le *Gate* pour pouvoir utiliser des applications tels que `ftp`, `telnet`, `netscape`... Les utilisateurs doivent donc se connecter sur le *Gate* afin d'accéder à l'extérieur. C'est un système facile à mettre en place mais très peu sécurisé du fait de l'accès d'utilisateurs sur l'élément protegeant la partie privée de la partie publique.
  - *Proxy Server* : un *Proxy* est un programme qui transmet les données qu'il reçoit à un autre programme situé sur une autre machine du réseau. Dans le cas d'un *Firewall*,

un *Proxy* sert à "forwarder" une requête (Web, FTP,...) à travers le *Firewall* du réseau interne au réseau externe.

– *Network Server* : il est également possible de lancer des serveurs (*daemon*) comme par exemple `Sendmail` sur le *Gate* afin de fournir d'autres services aux utilisateurs. Par contre, il n'est pas conseillé d'installer un serveur Web comme `Apache` sur un *Gate* pour des raisons de sécurité.

Voici quelques exemples de configurations pour l'utilisation d'un *Firewall*:

4 Techniques de *Hacking*

4.1 L'ingénierie sociale & l'irresponsabilité

Lorsque quelqu'un désire pénétrer dans un système informatique, sa première arme est le "baratin". Il n'y a généralement pas d'attaques réussies sans relations humaines. On appel ceci l'ingénierie sociale (*social engineering*), elle est basée sur quatres grands principes:

- **Le contexte** : en ayant une bonne connaissance de l'organigramme de l'entreprise cela permet à l'agresseur d'avoir d'ores et déjà un pied dans l'entreprise. Le but en général est de connaître qu'elles sont les personnes qui sont en droit de demander tels ou tels informations, et également à qui les demander, dans le but de se faire ultérieurement passer pour elles...
- **L'audace ou le bluff** : le bagout et l'art de la parole sont deux qualités indispensables lorsque l'on veut utiliser le "*social engineering*". lI s'agit ici d'avoir suffisament d'appoint et de connaissances techniques afin de faire croire à l'interlocuteur qu'il a affaire à un responsable technique de l'entreprise (ou d'un fournisseur de service). Tout ceci afin qu'il lui transmette les informations demandées sans aucun problème.
- **La chance** : la chance est également une part importante dans le "*social engineering*", cela ne marche pas à chaque fois ! Il faut de la pratique afin de bien maîtriser le séquencement du dialogue à établir.
- **La patience calculée** : il faut de plus savoir se montrer patient afin d'obtenir les informations désirées. Malgès tout, la méthode du "*social engineering*" demande une certaine rapidité (max. 1 heure) pour obtenir les informations voulues, passé ce délais, il est préférable de changer d'entreprise ou d'attendre quelques jours afin de ne pas éveiller les soupçons...

En général, les personnes ne sont pas formées à la notion de sécurité informatique ce qui entraîne des situations qui auraient facilement pû être évitées. A titre d'exemples, voici quelques cas fréquemment trouvés au sein des entreprises:

- *Disquettes ou sauvegardes* jetées à la poubelle sans être détruites au préalable. La plupart des *Hackers* sont friants des "poubelles" des entreprises qu'ils désirent pénétrer car elles recellent toujours d'informations très précieuses: fichiers de mots de passe, informations sur le systèmes, heures des sauvegardes, heures de connection des administrateurs,... Il faut donc toujours veiller à vérifier les informations stockées sur les disquettes et/ou les sauvegardes qui seront jetées à la poubelle. La meilleure solution consiste à les détruire afin de les rendre inutilisables.

Figure 1: *Firewall* utilisant une station UNIX équippée de deux ports réseau, un pour le réseau privé et un pour le réseau publique. Pour un maximum de sécurité, on interdira les comptes utilisateurs sur cette station et on installera un *Proxy* pour que les informations traversent le *Firewall*.

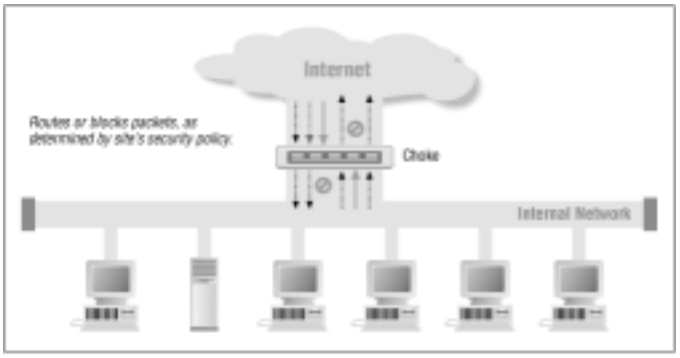

Figure 2: Il s'agit de filtrer les paquets à l'aide d'un *Choke*. On pourra ici utiliser un routeur pour sélectionner les paquets (TCP ou UDP) qui sont autorisés et pour quels services. Il s'agit d'une configuration facile à mettre en place et très utilisée actuellement.

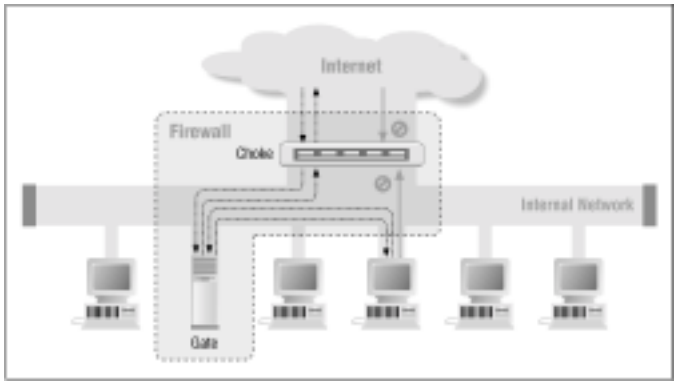

Figure 3: Il s'agit d'un type de *Firewall* plus sécurisé utilisant à la fois un *Choke* et un *Gate*. Le *Gate* doit être un ordinateur spécifique sur le réseau faisant tourner le serveur de mail ainsi que le *Proxy* (le serveur Web et le serveur FTP anonyme doivent être installés sur une autre machine devant le *Firewall*). Le *Choke* peut-être un routeur avec deux interfaces permettant de relier le *Gate* et le réseau interne.

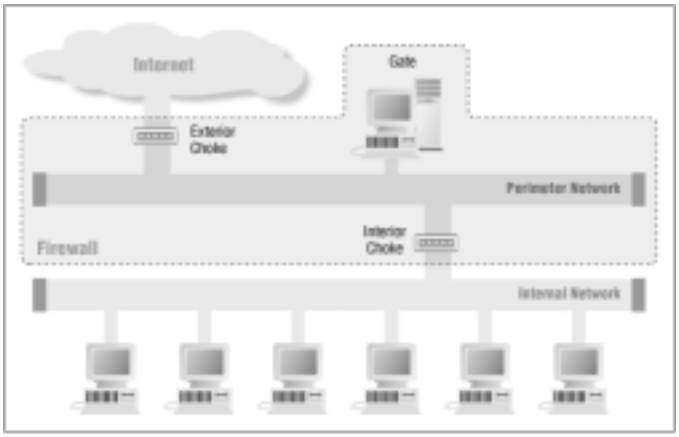

Figure 4: Pour une plus grande sécurité, ce type de *Firewall* est préconisé. Dans cette situation, si un intrus arrive à prendre le contrôle du *Gate*, il reste encore un troisième rempart permettant d'éviter qu'il ne puisse utiliser le *Gate* pour lancer une attaque contre le reste du réseau privé.

- *Papiers* ou *Post-It* où l'on note ses mots de passe qui seront par la suite jetés à la poubelle (même cas qu'au préalable), ou encore posés sur le bureau ou collés dans un tiroir afin de toujours l'avoir à porté de main. Il peut s'avérer utile d'inscrire son mot de passe sur un papier lorsqu'il est imposé par un administrateur mais il est préférable de pouvoir le fixer soit même en respectant un certains nombre de règles visant à rendre sa mémorisation facile mais sa découverte très difficile (il est très facile pour un *Hacker* de lancer un programme qui tentera de le "casser" en utilisant une liste de mots de passe pré-établie). Dans tous les cas, il faut veiller à conserver la confidentialité de ses mots de passe et donc d'en éliminer toutes traces facilement "accessibles" (lisibles).

- Les *échanges de mots de passe* sont également très fréquents. Ceci doit être évité au maximum car l'identité de l'utilisateur n'est plus assurée. Les actions qu'il engendre alors lui seront peut-être reprochées alors qu'il s'agissait d'un tiers à qui il avait eut le malheur de confier son mot de passe. Le couple "nom d'utilisateur" (*Login*) et "mot de passe" (*Password*) est personnel et ne doit en aucune manière être transmis à quelqu'un d'autre même pour "rendre service"...

Il s'agit ici du principe même du *Hacking*, les *Hackers* savent où trouver l'information et ensuite comment techniquement s'en servir afin de pénétrer les systèmes informatique. Les sections suivantes visent à présenter, de manière non exhaustive, ces différentes techniques permettant aux "*Hackers*" de passer outre les *Firewalls* et les systèmes d'authentification.

4.2 Les *Denial-of-Service* (DoS)

Les attaques de type *Denial-of-Service* ont pour but de saturer un routeur ou un serveur afin de le "crasher" ou en préambule d'une attaque massive. Ces types d'attaque sont très faciles à mettre en place et très difficile à empêcher (cqfd. à arrêter). Mais quelles sont les raisons qui peuvent pousser un attaquant à utiliser les *DoS* en sachant que cela peut mener à la "destruction" du routeur ou du serveur visé :

- *Récupérer un accès* : une attaque de type *Denial-of-Service* fait, la plupart du temps, partie d'une attaque visant à obtenir le contrôle d'une machine ou d'un réseau. Par exemple l'attaque de type "`SYN Flood`", très répendue, est souvent utilisée de paire avec une tentative de "*Spoofing*" (cf. ci-dessous).

- *Masquer les traces* : ce type d'attaque permet également de "crasher" une station qui par exemple aurait pû contenir des traces du passage d'un "*Hacker*". En détruisant cette station, il s'assure ainsi une certaine pérénité.

- *Se venger* : très fréquemment, ces attaques sont utilisées afin d'assouvir une vengeance personnelle contre une personne, un administrateur ou bien encore une entreprise...

- ...

Voici quelques exemples de programmes disponibles sur Internet permettant de réaliser ce genre d'attaque:

- *Ping 'O Death* : il s'agit de saturer un routeur ou un serveur en envoyant un nombre important de requêtes "`ICMP REQUEST`" dont les datagrammes dépassent la taille maximum autorisée. Des patches existent afin de se prémunir de ce type d'agression sous les systèmes MacOS, Windows NT/9x, Sun Solaris, Linux et Novell Netware.

- *Land - Blat* : cette attaque permet de geler la plupart des systèmes ayant plus ou moins un an. Il sera alors obligatoire de redémarrer la machine afin d'en reprendre le contrôle. Il

s'agit d'envoyer un paquet forgé (*spoofé*) contenant le flag `SYN` sur un port donné (comme 113 ou 139 par exemple) et de définir la source comme étant l'adresse de la station cible. Il existe un certains nombre de patches pour ce "bug" pour les systèmes UNIX et Windows.

- *Jolt* : spécialement destinée aux systèmes Microsoft (NT, 9x et 2000), cette attaque permet de saturer le processeur de la station qui la subie. La fragmentation IP provoque, lorsque l'on envoie un grand nombre de fragments de paquets identiques (150/sec), une saturation totale du processeur durant toute la durée de l'attaque. Des pré-patches existent déjà pour tenter de contrer ce type d'attaque.

- *TearDrop - SynDrop* : problème découvert dans le noyau du système Linux dans la partie concernant la fragmentation des paquets IP. Il s'agit d'un problème de reconstruction du paquet. Lorsque le système reconstitue le paquet, il exécute une boucle qui va permettre de stocker dans un nouveau "buffer" tous les paquets déjà reçus. Il y a effectivement un contrôle de la taille du paquet mais uniquement si ce dernier est trop grand. S'il est trop petit cela peut provoquer un problème au niveau du noyau et planter le système (problème d'alignement des paquets). Ce problème a également été observé sur les systèmes Windows (NT/9x) et des patches sont dès à présent disponibles.

- *Ident Attack* : ce problème dans le *daemon* `identd` permet aisément de déstabiliser une machine UNIX qui l'utilise. Un grand nombre de requêtes d'authorization entraîne une instabilité totale de la machine. Pour éviter cela, il faut installer une version plus récente du *daemon* `identd` ou alors utiliser le *daemon* `pidentd-2.8a4` (ou ultérieur).

- *Bonk - Boink* : même problème que le *TearDrop* mais légèrement modifié afin de ne pas être affecté par les patches fournis pour le *TearDrop*. Il existe de nouveaux patches mieux construits qui permettent également d'éviter ce nouveau type d'attaque.

- *Smurf* : ce programme utilise la technique de l'"`ICMP Flood`" et l'amplifie de manière à créer un véritable désastre sur la (ou les) machines visées. En fait, il utilise la technique du "Broadcast Ping" afin que le nombre de paquets ICMP envoyés à la station grandisse de manière exponentielle causant alors un crash presque inévitable. Il est difficile de se protéger de ce type d'attaque, il n'existe aucun patches mais des règles de filtrage correctes permettent de limiter son effet.

- *WinNuke* : il s'agit encore d'un programme permettant de "crasher" les sytèmes Windows NT/95 par l'envoie de données de type "`OOB`" (*Out Of Band*) lors d'une connection avec un client Windows. NetBIOS semble être le service le plus vulnérable à ce type d'attaque. Appraremment, Windows ne sait pas comment réagir à la réception de ce type de paquet et il "panique"... De nombreux patches existent contre ce type d'attaque et les versions récentes de Windows (98/2000) sont dès à présent protégées.

4.3 L'*IP Spoofing*

La technique de l'*IP Spoofing* est une technique dont le principe est relativement ancien (aux alentours de 1985) mais la première attaque connue l'utilisant ne remonte qu'à 1995. Kevin Mitnick, un célèbre "*Hacker*", l'a utilisé afin de s'infilter dans le réseau d'un expert en sécurité informatique, Tsutomo Shimomura. Le *Spoofing* n'est pas l'attaque en tant que tel, il s'agit d'une technique permettant de s'infiltrer dans un ordinateur en se faisant passer pour un autre en qui il a confiance (*Trusted Host*).
Avant de rentrer dans des détails plus techniques, voici un bref résumé du fonctionnement de cette technique: une station se fait passer pour une autre en envoyant un paquet dont l'adresse IP est "autorisée" par le serveur visé... La source IP envoyée trompe donc la cible qui accorde l'accès en pensant avoir affaire à une machine de confiance. Il existe différents types de *Spoofing*,

nous n'aborderons ici que les notions d'*IP Spoofing*, celles ayant attraits aux *DNS Spoofing*, *Web Spoofing*,... ne seront donc pas présentées. En ce qui nous concerne, il existe deux types d'*IP Spoofing* qui diffèrent par les moyens nécessaires à leur application.

4.3.1 *Non Blind Spoofing (NBS)*

Dans ce contexte, l'utilisation de la technique du *Spoofing* a pour but d'interférer avec une connection dont les paquets traversent un sous-réseau dont le "*Hacker*" a accès. Il peut donc aisément capturer et analyser les paquets qui sont échangés. En général, cette technique est utilisée lorsqu'il s'agit d'interagir entre deux machines du même sous-réseau ou alors il faut avoir un accès sur un routeur important (transatlantique par exemple) - la place rêvée pour un *Hacker*. Comme les paquets doivent impérativement traverser le sous-réseau, il est très facile de récupérer les paquets et d'obtenir les numéros de séquence (SEQ) et d'*acknowledgment* (ACK). Ce type de *Spoofing* est principalement utilisé pour les trois attaques suivantes:

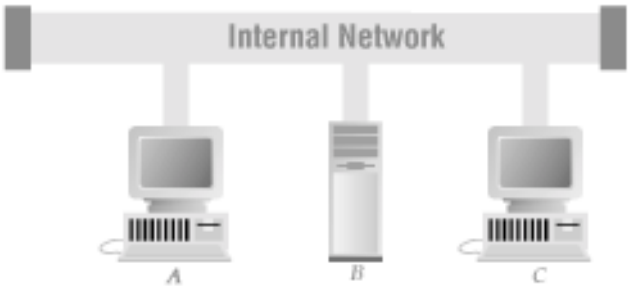

Figure 5: Configuration nécessaire pour le *Non Blind Spoofing*: l'ordinateur 'A' va demander une connection sur l'ordinateur 'C' alors que l'ordinateur 'B' espionne le traffic sur le réseau local. Il est donc possible pour l'ordinateur 'B' d'interrompre la relation entre 'A' et 'C' puis de se faire passer pour l'ordinateur 'A' car il a accès aux numéros de séquence (SEQ) et d'*acknoledgment* (ACK).

- `SYN Flooding` : le but du `SYN Flooding` est d'ouvrir "à moitié" un nombre de connection dépassant la taille du *backlog* (queue permettant de stocker les connections en attentes de fin d'ouverture, c'est-à-dire la terminaison du *3-way handshake* TCP) qui est limitée en général à 5 sous BSD (à 6 sous Linux). Cela a pour effet de bloquer totalement les nouvelles demandes de connection vers la station visée. Voilà brièvement comment cela s'avère possible: la station 'B' envoie un paquet de type `SYN` forgé (spoofé en tant que station A) vers la station 'C'. Lorsque la station 'A' est inaccessible (*unreachable*) la station 'C' va alors attendre une réponse (notion de *timeout*), dans le cas contraire (station 'A' accessible), ce type d'attaque n'a aucun effet sur la station 'C' car elle indiquera qu'aucune demande de connection n'a été effectuée de sa part. En exécutant cette opération un grand nombre de fois, la station 'C' ne va plus pouvoir répondre aux nouvelles demandes de connection et deviendra alors inacessible (et certainement instable).

- `Connection Killing` : dans ce cas de figure, l'objectif est de faire stopper une connection TCP, précédemment établie, entre deux stations (ici entre 'A' et 'C'). Il existe deux techniques pour effectuer ce type d'attaque, toutes les deux sont basées sur le *Spoofing* et sur les *flags* TCP. La première utilise le *flag* `RST` (Reset) alors que la seconde utilise le *flag* `FIN` (Fin). La seconde est plus difficile à mettre en oeuvre car elle exige que les

numéros de séquence et d'*acknolegment* soient tous deux corrects (ce qui n'est pas le cas de la première: seulement les numéros de séquence).
Voici donc comment fonctionne la méthode utilisant le *flag* `RST` (la seconde est basée sur le même principe avec le numéro d'*acknolegment* en plus): il faut tout d'abord attendre un paquet en provenance de la station 'B' à destination de la station 'A'. Ensuite, il faut calculer à partir du (ou des) paquets reçus le numéro de séquence à indiquer dans le paquet, contenant le *flag* `RST`, que nous allons envoyer à la station 'C' (rappel du calcul des numéros de séquence et d'*acknoledgment* sur la figure 6). Il ne reste plus qu'à envoyer le dit paquet et la connection sera logiquement rompue.

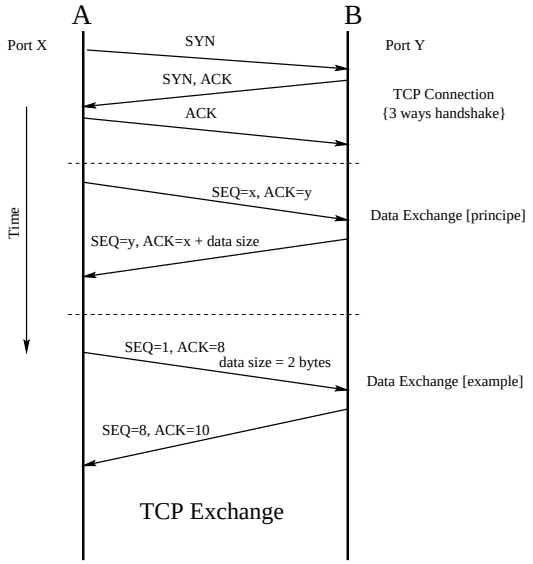

Figure 6: Connection et principe des échanges dans le protocole TCP (Transport Control Protocol).

- `Connection Hijacking` : ce type d'utilisation de l'*IP Spoofing* est relativement plus intéressante (en terme d'accès) que les deux précédentes. La `Connection Hikacking` permet de s'infiltrer dans une connection TCP existante dans le but de l'utiliser pour son propre intéret. Pour cela, il faut q'une connection TCP utilisant par exemple `Telnet` soit déjà établie entre deux stations ('A' et 'C' dans notre cas). Comme toujours, nous avons également besoin d'une troisième machine ('B') pour analyser les paquets émis sur le réseau. Cette technique nécessite plus de "précision" que les deux précédente car il faut savoir jongler avec les numéros de séquence (SEQ) ainsi qu'avec les numéros d'*acknoldgement*.

  Voici en quelques lignes comment fonctionne ce type d'attaque: l'objectif de cette manipulation est de récupérer une connection existante, on dit *Hijacker* une connection. Le protocole TCP sépare les bons des mauvais paquets à l'aide de leurs numéros SEQ/ACK. Si l'on arrive à faire en sorte que la station 'C' ne croit plus aux paquets de la station qui avait établie la connection (station 'A') mais par contre aux paquets forgés par la station 'B' (paquets spoofés en tant que station 'A') il est possible de continuer à "profiter" de la connection en utilisant des paquets de numéros SEQ/ACK valides pour la station 'C'. Une fois arrivé à ce niveau, nous avons récupéré la connection: la station 'A' est "perdue" et la station 'C' accepte sans problème nos données.

La question qui se pose alors est: "Comment faire en sorte que la station 'A' émette des paquets dont les numéros SEQ/ACK soient rejetés par la station 'C' ?". Il suffit simplement d'insérer, au bon moment, dans le fleau de données, un paquet supplémentaire que seuls les stations 'B' et 'C' connaitront. Dans ce cas, le serveur 'C' acceptera bien les données et mettra à jour le numéro d'*acknoledgment* (ACK). La station 'A' continuera de tenter d'envoyer des paquets avec l'ancien numéro de séquence (SEQ) qui ne seront donc plus acceptés par le serveur 'B'. La figure 7 présente à l'aide de paquets TCP le fonctionnement de cette technique.

Cette technique n'est utile que pour pénétrer les systèmes utilisant des systèmes d'authentification de type *One-Time Password*. L'utilisation de ces systèmes rend l'utilisation des techniques de *Sniffing* inopérantes car les mots de passe ne sont jamais transmis sur le réseau (notion de *Challenge*, cf. "Les mécanismes d'authentification").

4.3.2 *Blind Spoofing (BS)*

La technique du *Blind Spoofing* (aveugle) nécessite une tout autre configuration que pour le *Non Blind Spoofing* (cf. figure 8). L'avantage est qu'elle ne requière pas que l'attaquant (Hacker Server) soit capable de capturer les paquets émis par la station cible (Target Host). C'est pourquoi cette technique est appellée "aveugle". L'attaquant doit donc pouvoir prédire les paquets qui seront envoyés par la station qu'il désire attaquer. Afin d'utiliser le *Blind Spoofing*, il est nécessaire de connaître les adresses IP de quatre stations:

- La cible visée (Target Host).
- Une machine de "confiance" pour la cible visée (Trusted Host).
- Une adresse dont la station n'est pas accessible sur le réseau (*unreachable*).
- Un attaquant... (Hacker Server).

L'utilisation de cette technique est basée sur le principe de la relation de confiance qu'il est possible d'instaurer entre deux machines à l'aide par exemple des systèmes utilisant les fichier `/etc/hosts.equiv` ou les fichiers `~/.rhosts`. En utilisant ces mécanismes, l'authentification se fera uniquement par vérification de l'adresse IP (Internet Protocol) de la station qui demande la connection (pas de vérification d'identité à l'aide d'un mot de passe par exemple). Dans ce contexte, le *Spoofing* peut s'avérer intéressant....

L'attaque à proprement parlé va se dérouler en cinq étapes bien distinctes:

- Choix de la station cible (Target Host) selon des critères personnels: défis technique, vengeance,...
- Recherche et découverte d'une station de "confiance" (Trusted Host) (`showmount`, `rpcinfo`,...).
- Elimination de la station de "confiance" et "sampling" des numèros de séquence TCP.
- Tentative de forgeage de paquets IP en tant que station de "confiance" et connection sur la machine cible.
- Mise en place d'une *Backdoor* (cf. section suivante).

Le principe de fonctionnement est relativement simple, la station attaquante (Hacker Server) va tenter de se faire passer pour une station de "confiance" (Trusted Host) aux yeux de la station cible (Target Host). Il s'agit, une fois que la station de "confiance" a été trouvée, de rendre inaccessible (*unreachable*) cette dernière (à l'aide d'un *DoS* par exemple) afin qu'elle

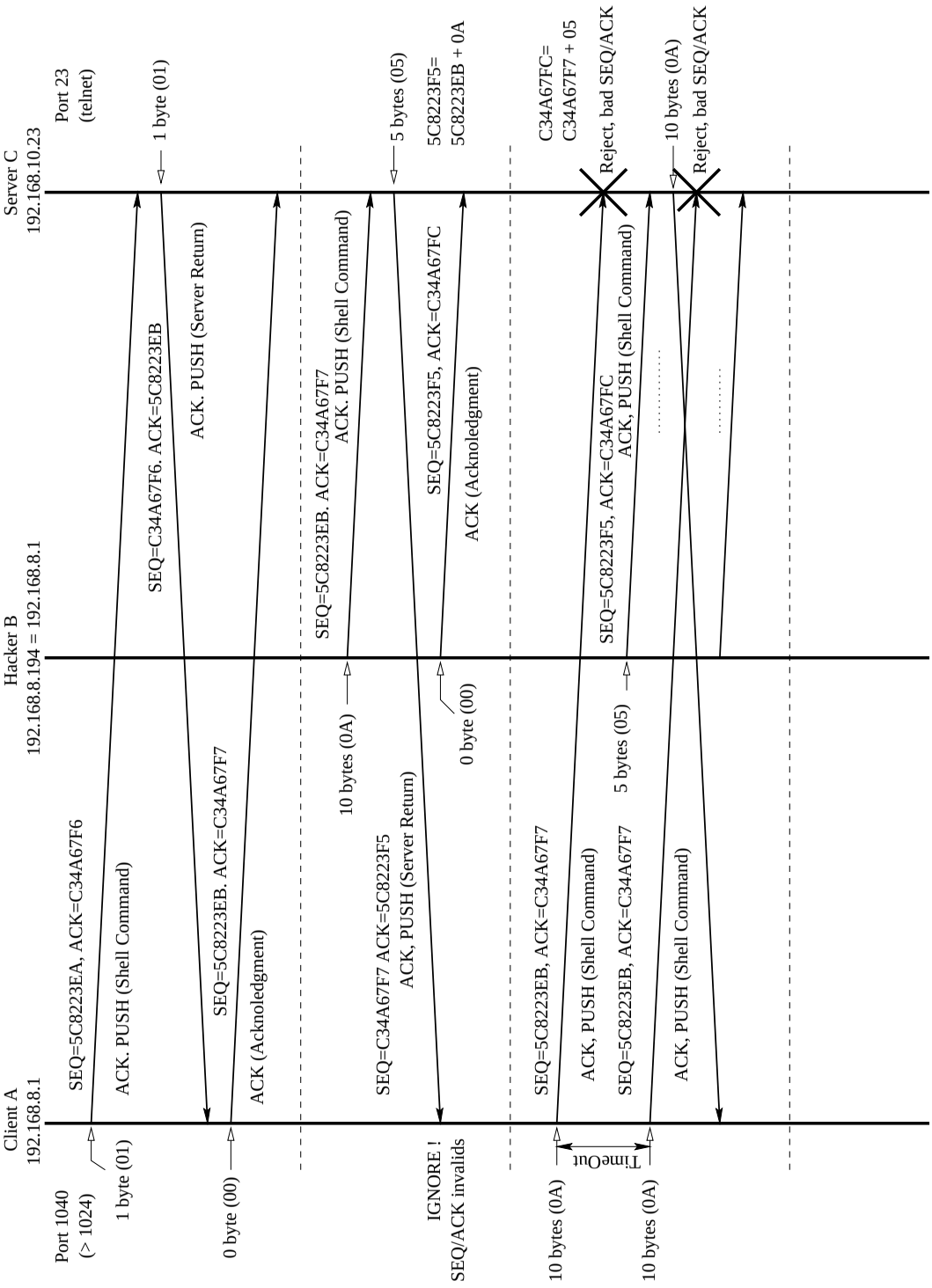

Figure 7: Présentation de la technique de **Connection Hijacking** à l'aide d'échanges de paquets TCP. Les trois premiers paquets échangés entre le Client (A) et le Serveur (C) montre le type de paquet transmis lors d'une connection en **telnet**. Le paquet suivant provient de l'attaquant (B) qui forge un paquet en se faisant passer pour la station 'A'. Le serveur va alors, dans le paquet suivant, retourner les informations à la station 'A', ce paquet passera par la station 'B' qui pourra alors renvoyer un *acknoledgement*. Les numéros de séquence (SEQ) et d' *acknoledgment* ont donc changés et le client (A) ne peut plus alors envoyer des paquets valides pour le serveur (C)...

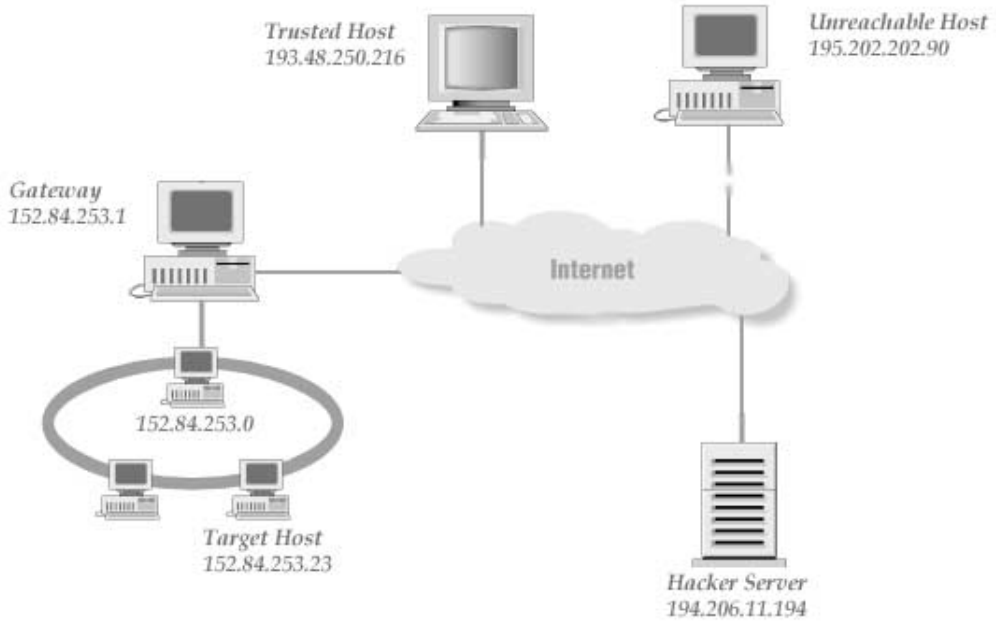

Figure 8: Exemple de configuration des stations lors d'une attaque de type *Blind Spoofing*. La station attaquante est le "Hacker Server" d'adresse IP 194.206.11.194, la station inacessible (permettant l'attaque de type *Denial Of Service*) est celle d'adresse 195.202.202.90, la station de "confiance" est 193.48.250.216 et enfin, la cible 152.84.253.23 (derrière une passerelle: 152.84.253.1). On pourra noter que les adresses IP sont bien de sous-réseaux différents.

ne puisse par interférer avec la tentative d'attaque. Ensuite, il est nécessaire d'établir une "première" connection avec la station cible afin de se faire une idée précise de l'état actuel des numéros de séquence (SEQ) et d'*acknoledgment*. Pour cela, une simple connection sur un port TCP quelconque (`SMTP` par exemple), juste avant de lancer une attaque, permettra d'obtenir le numéro de séquence initial fournit par la station cible (il sera judicieux d'effectuer cela plusieurs fois afin également d'obtenir un `RTT` (*Round Trip Time*) moyen). A partir de là, il est possible de tenter de forger un paquet IP (le plus rapidement possible) à destination de la station cible en se faisant passer pour la station de "confiance". Le problème étant que l'attaquant ne voit pas les paquets qui vont être émis par la station cible, il faudra donc qu'il arrive à les prédire afin de réagir en conséquence.

4.4 Les *Backdoors*

Depuis que les intrusions informatique existent, leurs adèptes ont mis au point un certains nombre de techniques leur facilitant l'accès aux systèmes pénétrés. La technique la plus connue, et sans doute la plus utilisée, est celle des *Backdoors* (portes dérobées ou portes de service). Elles permettent, à celui qui en connait l'existence et le fonctionnement, de revenir sur un système de façon détournée, c'est-à-dire sans passer par les méthodes d'authentification habituelles.

En règle général, les *Backdoors* permettent différents types d'actions sur le système où elles sont installée:

- se reconnecter sur la machine même après un changement de mots de passe ou d'ajouts de systèmes de sécurité.
- rendre invisible les connections et les actions réalisées.
- faciliter l'accès à la station sans avoir à utiliser des trous de sécurité existants (*security holes*).
- déranger le travail des utilisateurs par l'envoie de messages, la modification de fichiers, l'affichage d'images, la lecture de fichiers son...
- exécuter certaines commandes bien ciblées permettant d'avoir une vision de l'état de la station (processus, connections réseau, utilisateurs...) ou de modifier le contenu de certains fichiers de configuration (mots de passe, réseau,...).

Il existe différents types de *Backdoors*, certaines n'ont une utilité qu'une fois l'accès à la station accordé, d'autres permettent par exemple de contourner les différents types de *Firewalls*. Voici quelques exemples de *Backdoors* fréquemment trouvées sur les systèmes UNIX:

- **Password Cracking** : il est possible de faire "tourner" (on dit aussi "mouliner") un programme permettant de casser l'encryption des mots de passe du système (le logiciel `Crack` par exemple). Ainsi, l'intrus peut obtenir de nouveaux comptes sur la machine et tenter de s'en servir sur d'autres stations du réseau. Il est également possible de rechercher les comptes qui ne sont jamais (ou très rarement) utilisés et qui ont des mots de passe "trop" faciles. Il s'agit ensuite de modifier leurs mots de passe afin de s'assurer que l'administrateur ne les changera pas de sa propre initiative (en vérifiant les mots de passe à l'aide du programme `Crack` par exemple).
- **Rhosts + +** : les service `r*` utilisent les adresses IP comme moyen d'authentification (cf. section sur les méthodes d'authentification). Les fichiers `~/.rhosts` permettent d'indiquer quelles sont les machines et les utilisateurs autorisées à se connecter sans autres formes d'authentification. Il est donc facile de rajouter dans ces fichiers une entrée permettant d'autoriser toutes machines et tous utilisateurs à se connecter (l'entrée + +).

- **Login** : en modifiant le programme de `login` (fichier *login.c*), il est possible d'ajouter un mot de passe ou un nom de login "spécial" permettant d'accéder au système sans avoir de vérifications réelles de l'identité de l'utilisateur.

- **Telnetd** : les administrateurs vérifient fréquemment le fichier binaire de `login` sur les stations UNIX à l'aide de la commande `strings` (affichage des chaînes dans un fichier binaire). De ce fait, il peut être préférable de modifier le fichier `in.telnetd` en ajoutant un compte "spécial" ou encore en évitant la procédure d'authentification si le terminal de connection est défini selon un certains type...

- **Network Services** : ce type de *Backdoor* est basé sur le même principe que pour la précédente. Il s'agit de modifier les démons pour les services tels que `FTP`, `rsh`, `rlogin`, `finger`,... Il est également possible de créer son propre démon avec un protocole réseau bien défini puis de le rajouter dans le fichier de configuration `/etc/inetd.conf`.

- **Cronjob** : la `crontab` permet de planifier l'exécution de certaines tâches à des heures bien définies. Il suffit donc de lancer un démon à une certaine heure donnée (durant la nuit de préférence) donnant alors l'accès au système. On peut également modifier le code source d'un des programmes déjà éxécutés par la `crontab` afin qu'il fournisse ce type de service.

- **Shared Library** : pratiquement tous les systèmes UNIX actuels utilisent des librairies partagées (*Shared Library*). Ces dernières permettent la réutilisation de routines sans avoir à les intégrer directement dans le fichier binaire. Il est donc possible de modifier certaines routines utilisées par les procédures d'authentification (la fonction `crypt()` par exemple) pour que dans certaines conditions elles accordent l'accès sans réelles vérifications.

- **System Kernel** : le noyau (*Kernel*) est la base définissant comment le système tout entier fonctionne. Il est donc également possible de tenter d'inclure certaines fonctions dans le noyau agissant en tant que *Backdoors*.

- **File System** : ce type de *Backdoor* ne vise pas à conserver le contrôle de la station mais y participe activement en permettant de cacher certains fichiers et répertoires où sont généralement stocké les programmes utilisés par les *Hackers*. Il faut donc modifier les programmes comme `ls`, `du`, `fsck`,... afin de masquer ces fichiers et répertoires.

- **BootBlock** : il est possible de rajouter, dans le *BootBlock* du système, des morceaux de code visant à donner accès au système sans aucune authentification après un redémarrage.

- **Process Hiding** : l'important pour un intrus est de ne pas se faire repérer par un administrateur un peu trop curieux. Les processus sont la première chose qui pourrait le trahir. Pour éviter cela, il faut modifier le programme `ps` (le programme `top` également) afin que certains processus n'apparaissent pas lors de l'affichage des processus sur le système. On peut, plus facilement, renommer les programmes ou encore modifier la variable `argv[]` dans un programme écrit en langage C.

Il est facile de trouver sur Internet des archives combinant plusieurs de ces *Backdoors*, se sont les `RootKits`. Chaque système en possède une dédié afin d'être assuré de son bon fonctionnement.
En plus de ces *Backdoors*, les *Hackers* tiennent à ne pas laisser de traces au niveau de leurs activités sur le rséau. Une connection sur le port `telnet` (23) est souvent "monitorée" et donc visible pour l'administrateur, il en est de même pour les routeurs qui définissent des régles de filtrage qui ne laisseront pas passer une connection de ce type de l'extérieur vers l'intérieur du réseau. Un certains nombre d'outils ont donc été développé afin de contourner ce type de

problème. Ils utilisent, souvent, le même principe que le remote login (`rlogin`) mais avec des ports supérieurs à 1024 (ports souvent moins contrôlés et ne faisant pas partie des ports dits sécurisés.) ainsi qu'un protocole réseau propriétaire. Voilà les trois grandes familles de *Shell Backdoors* que l'on peut actuellement trouver:

- **TCP Shell** : il est possible d'installer des "remote shells" sur des ports TCP suffisament élevés (*High Port Numbers*) pour ne pas être contrôlés par le routeur ou monitorés par le système de logs. Cela permet souvent de pouvoir éviter les *Firewalls*. Il est également possible de modifier un démon TCP (comme SMTP par exemple) afin qu'il puisse fournir d'autres services que ceux pour lesquels il était destiné: shell, exécution de commandes,...
- **UDP Shell** : en règle général le trafic TCP est relativement contrôlé, certains *Hackers* préfèrent donc contourner le problème en utilisant le protocole UDP (User Datagram Protocol). Par exemple, il est intéressant d'installer des *Backdoors* sur les services relevant du DNS (`named` ou `bind`) car ils utilisent le port 53 (`domain`) qui est rarement contrôlé tant par les filtres que par les systèmes de logs.
- **ICMP Shell** : la commande `ping` est sans aucun doute la plus utilisée lorsqu'il s'agit de vérifier la présence d'une station sur le réseau. Il est possible de modifier le contenu des paquets ICMP afin de permettre l'échange de données à travers ce protocole. N'étant pas, à l'origine, un protocole d'échange de données mais d'informations, l'ICMP est rarement stoppé par les *Firewalls* et encore plus rarement monitoré.

En plus de cela, il est très courant aujourd'hui d'utiliser des algorithmes de cryptage rendant encore plus difficile la découverte des *Backdoors* car l'administrateur ne peut pas comprendre les données qui sont échangées et donc réagir en conséquence.

Ce type de problème est très fréquent sous les systèmes UNIX mais aujourd'hui, avec la montée en puissance d'Internet, les systèmes qui n'étaient pas destinés aux réseaux les subissent également. Les programmes comme `BackOrifice` du CDC (*Cult of the Dead Cow*) ou encore `NetBus` permettent d'agir à distance sur les systèmes Microsoft (9x/NT/2000): messages, captures d'écran, redémarrage, ouverture du lecteur de CD, exécution de programmes,...

Bien entendu, un administrateur peut envisager d'installer un certain type de *Backdoor* afin, par exemple, de récupérer rapidement le contrôle de son système après une intrusion ou d'avoir une vue général de l'état d'une station. Dans ce cas, l'utilisation de *Backdoors* peuvent s'avérer utile mais il s'agit alors de bien les concevoir et de les garder secrètes afin que leur utilisation ne soit pas détournée.

4.5 Remote Buffer Overflow Exploits

La fonction principale d'un processeur est de traiter et de déplacer des données. Lors de ces traitements, le processeur a besoin d'un emplacement afin de sauvegarder rapidement les données traitées. La taille des registres ne permet pas à ceux-ci de jouer ce rôle. Ces informations sont donc sauvées, à l'aide de commandes spécifiques et plus rapides, dans une zone mémoire appellées *pile*. Elle est stockée en mémoire à une adresse spécifique (qui peut-être changée) et est de taille variable.

4.5.1 Structure de la pile

Voilà, brièvement, comment fonctionne le processeur a ce niveau: si le registe N est utilisé et qu'une sous-procédure est exécutée et qu'elle requière l'utilisation de ce même registre (N), le processeur doit sauver le contenu de ce registre dans la *pile* afin de pouvoir le restaurer après

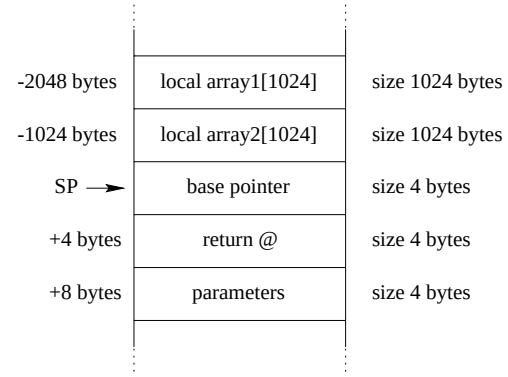

Figure 9: Exemple de pile contenant deux tableaux, le pointeur de pile (SP) et l'adresse de retour.

la terminaison de la sous-procédure. Pour cela, le processeur doit connaître l'adresse de retour lorsque la sous-procédure se termine. Cette adresse est donc également sauvée dans la pile avant son exécution. Lorsque la sous-procédure se terminera le processeur "sautera" (*jump*) à l'adresse de retour précédemment stockée dans la pile. La pile a une seconde utilitée, celle de stocker en mémoire les nouvelles données créées ou reçues par le programme.
NB: La gestion des entrées/sorties dans la pile utilise la méthode du LIFO (Last In - First Out)

4.5.2 Abuser l'adresse de retour

Lors de l'exécution de la procédure le processeur sauve l'adresse de retour dans la pile, lorsque la procédure se terminera le processeur retournera à l'adresse spécifiée et continuera son travail... Si (par hasard !) une procédure écrivait plus d'octets (bytes) dans une variable locale afin que la taille nécessaire à son stockage dans la pile dépasse celle de l'adresse de retour, on appellerait ceci un *Buffer Overflow*. En reprenant la structure de pile précédente et en inscrivant 1032 fois le caractère "X" dans le tableau local "array2", la procédure va alors dépasser sa propre adresse de retour (cf. figure 10).

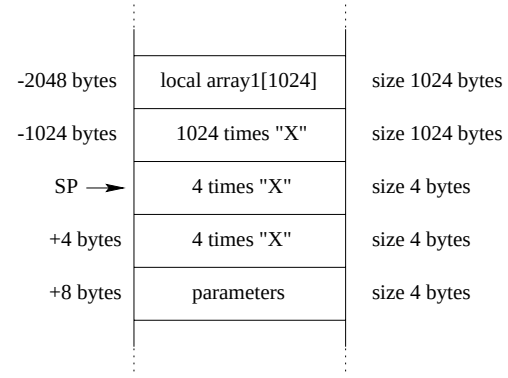

Figure 10: Exemple de *Buffer Overflow* en saturant le tableau "array2" ainsi que le *Stack Pointer* et l'adresse de retour de sous-procédure.

Au lieu de "stupidement" inscrire des caractères à l'adresse de retour, il serait plus judicieux (et plus intéressant) d'indiquer une nouvelle adresse mémoire dans la pile. Il sera alors possible d'indiquer au programme une nouvelle adresse de retour contenant un code totalement différent. Le programme sautera alors automatiquement à l'adresse spécifiée... Cela s'avère d'autant plus intéressant lorsque l'on arrive à forcer le programme à sauter vers une adresse où est situé un code assembleur destiné par exemple à exécuter un *shell* UNIX (`/bin/sh`).

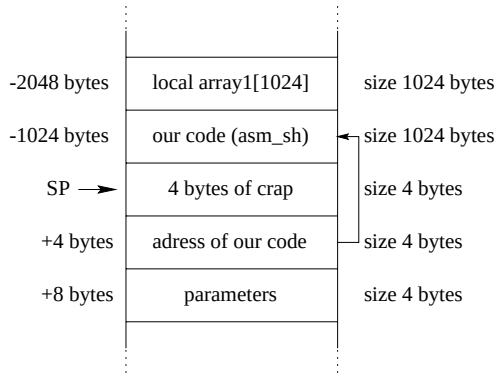

Figure 11: Utilisation d'un *Buffer Overflow* afin de récupérer un shell UNIX. Le tableau "array2" contient le code assembleur permettant l'appel au fichier "/bin/sh" donnant alors la possiblité d'utiliser le système à des fins détournées.

L'utilisation des *Buffer Overflow* s'avère très intéressante (pour un *Hacker*) lorsqu'elle est appliquée à des programmes où le *Sticky Bit* est activé (programmes SUID). Ce mode spécial de certains programmes UNIX permet leur exécution sous un autre nom d'utilisateur (souvent `root`). Il est alors possible de récupére un compte administrateur (`root`) ou encore d'exécuter une commande quelconque sur le système...

4.5.3 Pourquoi Remote ?

Ce qui a été exposé précédemment concerne le principe de fonctionnement des *Buffer Overflow*, ces derniers peuvent être utilisés localement ou par l'intermédiaire d'un réseau informatique. Ils permettent alors à des attaquants de pénétrer les systèmes informatiques sans avoir recours aux méthodes d'authentification conventionnelles. Ce type d'attaque n'est pas encore très fréquent car les programmes capables de les réalisés sont très difficiles à mettre au point (transmission des informations, durée, réactivité,...). Cependant, un certains nombre de programmes capables de réalisés ce type d'attaque existent. On pourra par exemple citer les programmes visant les services FTP ou DNS...

5 Et maintenant ?

Lorsqu'un intrus est arrivé à ce niveau, tous le réseau est compromis, c'est l'état de siège ! Que va-t-il faire à présent ? Les objectifs des intrus peuvent, comme nous avons pu le voir précédemment, être très variés. Dans (pratiquement) tous les cas, l'objectif final est de récupérer un contrôle total sur la, ou les, machines cibles et de le garder le plus longtemps possible. Il va

donc falloir évincer le personnel chargé de l'administration et de la gestion du réseau et s'assurer un retour aisé à la prochaine connection.

5.1 Récupération d'autres comptes

En règle générale, la première opération consiste à obtenir d'autres voies d'accès au système et cela par l'intermédiaire de différents comptes d'utilisateurs. Pour cela plusieurs techniques peuvent être utilisées, voici une présentation, non exhaustive, de quelques une:

- "Cassage" (*Cracking*) des mots de passe contenus dans le fichier `/etc/passwd` (version accessible aux utilisateurs classiques, dans le cas des fichiers `/etc/shadow` par exemple, l'accès au compte `root` est requis). Cette méthode a l'inconvénient majeur de nécessiter "beaucoup" de temps car il faut tester toutes les possiblités de mots de passe, de les encrypter et de les comparer avec ceux contenu dans le fichier précédemment récupérer. Des logiciels spécifiques permettent de réaliser cette tâche: Crack 5.0 ou encore John The Ripper 1.6. Une politique et un contrôle correct des mots de passe rend cette méthode sans fondement !

- Espionnage des connections d'utilisateurs (*TTY Snoop*), cette technique reprend le principe des *Backdoors*, il s'agit par exemple de changer le fichier binaire du programme de login afin qu'à chaque connection le couple login/password soit envoyé à une adresse ou stocké dans un fichier sur le système. Il est très facile de réaliser un programme permettant de réaliser cette opération. Ce type de problème est facilement évitable en utilisant des programmes calculant des *checksums* sur chaques fichiers "importants" du système et en les vérifiant fréquemment (ex: `TripWire`).

```
/* Login Trojan (/bin/login)                                          */
/* Inscrit dans /home/Ofh/toto le login/password de celui qui l'utilise */
/* Triton(c) - 1998/2000                                              */
#include <stdio.h>

void main(void)
{
        char *name[80];
        char *pw[20];
        char *prompt[20];
        FILE *strm;

        printf("\033[2J")
        printf("Welcome to Nasca...\n");
        printf("On a Linux 2.2.13 Powered.\n\n");

        printf("Nasca login: ");
        scanf("%s", &name);

        strcpy( prompt, name );
        strncat( prompt, "'s Password: ", strlen("'s Password: "));
        strcpy( pw, ( char * )getpass(prompt));

        strm = fopen("/home/Ofh/toto","a");
        if( strm == NULL )
        {
                strm = fopen("/home/Ofh/toto", "w" );
```

```
        }
        fprintf( strm, "User: (%s), Password: [%s]\n", name, pw );
        fclose( strm );
        printf("Bus Error - core dumped\n");
        exit(1);
}
```

- Espionnage du réseau local (*Sniffing*), cette technique très utilisée actuellement ne peut-être mise en défaut que par l'utilisation du cryptage. Il s'agit de capturer toutes les trames qui circulent sur le réseau en configurant l'interface réseau de la station dans un mode spécial, *promiscious*, qui permet de recevoir toutes les trames qui circulent sans pour autant en être le destinataire. Il est alors possible de récupérer les comptes des utilisateurs utilisant FTP ou Telnet par exemple.

```
Connection Telnet de 192.168.9.194 vers 192.168.8.197
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
--------------------------------------
192.168.9.194[1093]->192.168.8.197[23]
--------------------------------------
BDEDBH
@M#L^@^@^@^@ACCyyu^CyuXyu_yu yuy!yu^@LINUXyXyLINUXyyAyuAfrank^M
@frank21^M@frank21^M@frank21^M@frank21^Mexit^@
(ici on peut trouver le compte et le mot de passe: frank/frank21)

--------------------------------------
192.168.8.197[23]->192.168.9.194[1093]
--------------------------------------
telnet (aix)

AIX Version 4
(C) Copyrights by IBM and by others 1982, 1996.
login: frank
frank's Password:
3004-610 You are required to change your password.
        Please choose a new one.

frank's New password:
Enter the new password again:
*******************************************************************************
*                                                                             *
*                                                                             *
*  Welcome to AIX Version 4.2!                                                *
*                                                                             *
*                                                                             *
*  Please see the README file in /usr/lpp/bos for information pertinent to    *
*  this release of the AIX Operating System.                                  *
*                                                                             *
*                                                                             *
*******************************************************************************
```

```
% exit
```

- *Buffer Overflow*, il est très fréquent d'utiliser des *Buffer Overflow* afin de récupérer le compte de l'administrateur du système (but ultime). Il existe un nombre impressionant de programmes permettants l'utilisation de cette technique malgrès toutes les mises à jour ou patches proposés par les constructeurs. Cette méthode est sans nulle doute l'une des plus utilisée et des plus difficile à détecter.

Maintenant que l'intrus est passé maître à bord, il lui faut s'assurer une certaine pérénité afin de ne pas être dérangé et surtout délogé de sa position. Pour cela il mettra en oeuvre les différentes techniques qui ont été présentées mais de plus, il lui faudra prêter attention aux différents mécanismes de traçage installés par les administrateurs (`syslog`, `accton`,...).

5.2 Laisser place nette

Rentrer, sortir, trafiquer le système, chercher des informations, utiliser des exploits,... peuvent être des occupations passionantes mais laissent des traces visibles par un administrateur consciencieux. En règle général les mécanismes de tracabilité sont souvent délaissés car les administrateurs voient simplement en eux des fichiers volumineux et inutiles en temps normal (ce qui n'est pas entièrement faux). Malgrès tout, ces fichiers sont très utils en cas de problèmes, que ce soit du fait d'une tentative d'intrusion ou d'un problème technique. Ils ont pour but d'aider l'administrateur à gérer correctement ses stations.

Le mécanisme le plus connu sous UNIX est sans aucun doute le démon `syslogd` qui permet d'enregistrer des informations relatives au système et aux différents processus tournant sur la machine. Cela peut aller de la simple tentative de login à des messages de *debugging* de certains démons. D'autres programmes permettent d'assurer des fonctions de *logging* plus poussés comme par exemple le programme d'audit `accton` très répandu sur les systèmes SUN mais rarement configuré. Sur certains systèmes il est également possible de trouver d'autres types de programmes de logs permettant par exemple d'analyser en temps réel le trafic réseau ou les différentes tentatives de connection. En plus de ces divers mécanismes, le système UNIX tient en règle général une base d'informations concernant les différentes tentatives et réussites/échecs de login sur le systèmes dans les fichiers `*tmp` (utmp & wtmp).

Les *Hackers* ont mis au point différents programmes permettant de retirer leurs traces dans les différents fichiers de log présent sur le système. Ces programmes sont plus ou moins éfficaces mais dans tous les cas il est préférable de maintenir une sauvegarde distante de tous ces fichiers sur une machine "non accessible" à partir du réseau.

6 Conclusions

Actuellement, le problème est de correctement définir les risques engendrés par la criminalité informatique. Il faut pour cela avoir une vision globale du problème et connaître globalement les techniques utilisés par les nouveaux flibustiers. Il s'agira ensuite d'analyser correctement les vulnérabilités propre à chaque site, de définir le niveau de sécurité requis et enfin de mettre en place une politique de sécurité acceptable. Lors de cette étape il faut bien veiller à examiner le problème tant du côté de l'administrateur que de celui du simple utilisateur afin de ne pas en créer de nouveaux...

Références

[1] Simson Garfinkel & Gene Spafford, *Practical UNIX & Internet Security*, O'Reilly & Associates, Second Edition, 1996, ISBN 1-56592-148-8.

[2] Lars Klander, *Hacker Proof - The Ultimate Guide to Network Security*, Jamsa Press, 1997, ISBN 1-884133-55-X.

[3] David Rudder, *Firewall and Proxy HOWTO*, 1997.

[4] Hans Husman, *Introduction to Denial-of-Service*, 1996.

[5] Philippe Leroy & Jérôme Thorel, *Les clefs pour une sécurité maîtrisée*, Planète Internet, 17 : 56-65, 1997.

[6] Ajay Kumar Gummmadi et *al*, *Advanced Networking Security - Final Report*, 1996.

[7] Chuck Semeria, *Internet Firewalls and Security - A Technology Overview*, 3Com Corporation, 1996.

[8] Brecht Claerhout, *A short overview of IP spoofing*, 1996.

[9] Deamon9, Route, Infinity, *IP-spoofing Demystified*, Phrack Magazine, 7, 1996.

[10] CERT Coordination Center, *TCP SYN Flooding and IP Spoofing Attacks*, CA-06.21, 1998.

[11] Rick Farrow, *Sequence Number Attacks*, 1997.

[12] Steven M. Bellovin, *Defending Against Sequence Number Attacks*, Network Working Group - RFC1948, 1996.

[13] W. Richard Steven, *TCP/IP Illustrated - Volume 1*, Addison-Wesley, 1993, ISBN 0-201-63346-9.

[14] Steven M. Bellovin, *Security Problems in the TCP/IP Protocol Suite*, Computer Communications Review, 19.2 : 32-48, 1989.

[15] Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach, *Technical Report 540-96*, Department of Computer Science - Princeton University, 1996.

[16] Robert T. Morris, *A Weakness in the 4.2BSD UNIX TCP/IP Software*, AT&T Bell Laboratories, 1985.

[17] D. Brent Chapman, *Network (In)Security Through IP Packet Filtering*, Great Circle Associates.

[18] Steven M. Bellovin, *Packets Found on an Internet*, 1993.

[19] Steven M. Bellovin, *There Be Dragons*, In Proceedings of the Third Usenix UNIX Security Sympisum, 1992.

[20] CERT Coordination Center, *IP Spoofing Attacks and Hijacked Terminal Connections*, Advisory CA-95.01, 1997.

[21] P. Ferguson & D. Senie, *Network Ingress Filtering : Defeating Denial of Service Attacks which employ IP Source Adress Spoofing*, Network Working Group - RFC2267, 1998.

[22] Laurent Joncheray, *A Simple Active Attack Against TCP*, Merit Network Inc., 1995.

[23] IDA Architecture Guidelines - Part IV, *Security Tutorial - Cryptography and Authentification services*, Version 3.2.1, 1998.

[24] N. Haller, C. Metz, P. Nesser, M. Straw, *A One-Time Password System*, Network Working Group - RFC2289, 1998.

[25] J. Kohl, C. Neuman, *The Kerberos Network Authenticatin Service (V5)*, Network Working Group - RFC1510, 1993.

[26] T. Ylonen, *The SSH (Secure Shell) Remote Login Protocol*, Network Working Group - Draft SSH Protocol 00, 1995.

[27] Craig Hunt, *TCP/IP Network Administration*, O'Reilly & Associates, Second Edition, 1997, ISBN 1-56592-322-7.

[28] Bill Cheswick, *An Evening with Berferd in which a Cracker is Lured, Endured, and Studied*, USENIX Proceedings, 1990.

[29] Christopher Klaus, *Backdoors*, Technical Report, 1997.

[30] Van Hauser, *Placing Backdoors Through Firewalls*, THC [The Hacker's Choice], 1998.

[31] Dan Farmer & Wietse Venema, *Improving the Security of Your Site by Breaking Into it.*

[32] J. Reynolds & J. Postel, *Assigned Numbers*, Network Working Group - RFC1010, 1987.

[33] THC [The Hacker's Choice], *Stack Overflow Exploits on Linux, BSDOS, FREEBSD, SunOS, Solaris, HP-UX*, 1996.

[34] Aleph One, *Smashing The Stack For Fun And Profit*, Phrack 49 file 14, 1999.