INSA – 5° R.T.

Sujet d'examen
12 janvier 2006

Sécurité des systèmes informatiques

2ème partie

Exercice 1 *(3 points)*

Question 1 *(1 point)* : Quelles valeurs le programme C ci-dessous va-t'il afficher ? Pourquoi ?

```
#include <stdio.h>

int main() {
      char buffer [10];
      char *ptr;

      buffer[0] = 'A';
      buffer[1] = 'B';
      buffer[2] = 'C';
      buffer[3] = 'D';

      ptr = buffer + 2;
      *ptr = 'Z';

      printf("%c %c %c %c \n", buffer[0], buffer[1],
                              buffer[2], buffer[3]);
}
```

Dans un premier temps, on stocke les caractères 'A', 'B', 'C' et 'D' dans les quatre premières cases du tableau `buffer`*. On stocke ensuite dans la variable* `ptr` *l'adresse de* `buffer` **+ 2***, c'est à dire l'adresse de* `buffer` *à laquelle on ajoute la taille de l'espace mémoire occupée par deux cases du tableau. Dans cet espace mémoire, qui correspond donc à la troisième case du tableau, on stocke le caractère 'Z'. Finalement, le programme affiche donc :* « `A B Z D` ».

Question 2 *(2 points)* : Une société est spécialisée dans la vente (légale) de fichiers musicaux sur Internet. Les personnes souhaitant acheter ces fichiers doivent avoir préalablement ouvert un compte sur le site de la société. Lorsqu'un client souhaite télécharger une nouvelle chanson, il exécute via son navigateur un script faisant appel à la fonction `buy()` ci-dessous. Cette fonction prend en argument l'identifiant du client `nickname`, son mot de passe `password`, son nom `name` ainsi que le numéro de la chanson désirée `song_num`. La fonction `debit()` comptabilise les fichiers que le client a téléchargés, la fonction `send()` retourne le fichier sur le navigateur du client et la fonction `authenticate()` permet de vérifier que le mot de passe indiqué est correct. On suppose que ces 3 fonctions, qui ne sont pas présentées ici, sont correctement implémentées.

```
void buy(const char* nickname, const char* password,
         const char* name,     const int song_num) {
      if (authenticate(nickname, password) == 1) {
                log_and_send(name, song_num);
                log_and_debit(nickname);
}
```

```
void log_and_send(const char* name, const int song_num) {
      char msg[100]="";
      strcat(msg, "Download initiated for ");
      strcat(msg, name);
      strcat(msg, ".\n");
      printf(msg);
      send(song_num);
}
void log_and_debit(const char* nickname) {
      debit(nickname);
      printf("Charged 3euros.\n");
}
```

a) Quelle vulnérabilité est fréquemment rencontrée dans les programmes, en particulier en C ?
b) Décrire comment cette technique peut être appliquée dans le cas présent afin qu'un client puisse récupérer un fichier sans être débité du montant de son achat.

a) Le débordement de tampon est une technique fréquemment utilisée, notamment en C. Cela consiste à écrire dans une partie de la mémoire une quantité d'information plus grande que l'espace allouée ne peut en contenir. Cette technique n'est utilisable que lorsque les paramètres acceptés en entrée par un programme ne sont pas correctement validés avant leur utilisation (le langage de programmation utilisé peut également avoir un effet sur les conséquences de ces problèmes). Elle peut avoir plusieurs effets, plus ou moins critiques selon les cas: par exemple l'écrasement de zones mémoires allouées dynamiquement peut conduire à la corruption des données du programme (qui conduit généralement à un comportement erroné, parfois exploitable pour prendre le contrôle du programme), ou à une corruption de la pile du programme en cours d'exécution (ce qui est généralement plus grave car plus facile à exploiter en influençant l'adresse de retour des procédures).
b) Dans le cas présent, on peut exploiter deux erreurs d'implémentation qui rendent le programme facile à exploiter: l'utilisation dans la procédure `log_and_send()` *d'une variable de pile de taille limitée (le tableau* `msg`*) sans vérification de la taille des données qui y sont copiées (le paramètre* `name`*), et l'appel de la procédure* `log_and_send()` *avant celui de la procédure* `log_and_debit()` *dans la procédure* `buy()`*.*
En s'appuyant sur la première erreur,il est facile de corrompre la pile d'exécution du programme au moment où il exécute `log_and_send()` *en passant un paramètre name de longueur supérieure à 100. Une technique sophistiquée d'exploitation du programme pourrait probablement ici être utilisée pour prendre le contrôle total du programme; mais elle n'est pas nécessaire: il suffit de remplir suffisamment la pile avec des valeurs aléatoires pour corrompre l'adresse de retour de la procédure* `log_and_send()`*, ce qui conduira très certainement à une violation de la protection mémoire et un arrêt brutal du programme à la fin de cette procédure. Compte tenu de la deuxième erreur, le programme va défaillir après que le fichier souhaité ait été renvoyé et avant que le montant de l'achat ne soit pris en compte.*

Exercice 2 *(3 points)*

On donne ci-après une description du virus CIH (aussi connu sous le nom de Tchernobyl en raison de sa date d'activation), apparu en 1998.
Le virus CIH infecte les fichiers exécutables Windows 32 bits au format PE, caractéristiques de Windows 95. D'une taille de 1003 octets dans sa version originale, il fragmente son code

pour le placer dans les zones vides (ou contenant des instructions nulles) des fichiers. La taille du fichier infecté n'est donc pas modifiée.
Chaque 26 du mois (pour la version originale de CIH), le programme entre dans sa phase destructive qui comprend deux aspects. D'une part, le virus corrompt le BIOS en modifiant des données dans la mémoire *flash* (cette procédure ne fonctionnant qu'avec certains types de BIOS). CIH empêche ainsi l'ordinateur de redémarrer puisque le BIOS sert à initialiser et à gérer les entrées/sorties des périphériques. D'autre part, le virus écrase avec des données aléatoires les 2048 premiers secteurs (1 Mo) de tous les disques durs de la machine infectée. Ce faisant, le virus détruit des informations essentielles au bon fonctionnement de la machine infectée (notamment la table des partitions et la table d'allocation des fichiers, ainsi que, éventuellement, sa copie, si celle-ci se situe par hasard dans ces 2048 secteurs).

On considère ensuite Jacques Dupond, l'administrateur d'un parc informatique à cette période, constitué de machines utilisant le système Windows 95 (généralement les postes de travail) et de machines utilisant le système Windows NT (généralement pour les serveurs). La plupart des serveurs sont sauvegardés sur bandes magnétiques, sur une base hebdomadaire.

Répondez aux questions suivantes en justifiant votre réponse.

A) Est-il nécessaire que l'administrateur nettoie l'ensemble du parc informatique avec un antivirus puisque CIH ne s'attaque qu'à Windows 95/98 ?
B) On suppose que le virus a déjà agi sur la machine faisant office de serveur de messagerie. En démarrant, la machine affiche le message « Insérer une disquette de démarrage ». Quelles données le virus a-t'il corrompues ?
C) On suppose maintenant que le virus a également agi sur le serveur HTTP. En démarrant la machine, plus rien ne s'affiche à l'écran. Quelles données le virus a-t'il corrompues ?
D) Peut-on espérer récupérer des données du disque dur du serveur de messagerie ?
E) Comment remettre en fonctionnement le serveur Web ?
F) Est-il possible de se prémunir des virus s'attaquant au BIOS ?

A) Même si le virus s'attaque spécifiquement aux systèmes d'exploitation Windows 95 ou Windows 98, il peut être présent dans des fichiers gérés par d'autres types de machines (par exemple des serveurs de fichiers utilisés par des clients sous 95 ou 98 qui ont pu y stocker des fichiers infectés). Il est donc préférable d'exécuter un antivirus sur l'ensemble des machines ayant été étroitement en contact avec les systèmes infectés, même si on est sûr qu'elles ne peuvent pas elles-mêmes être infectées.
B) Si la machine affiche le message indiqué, cela signifie que son programme d'amorçage (en BIOS) fonctionne encore; par contre, compte tenu du message, il est probable que son disque dur a été endommagé par le virus
C) Dans le cas du serveur HTTP, par contre, le virus a probablement réussi à altérer le programme du BIOS ce qui empêche alors la machine d'initialiser ses circuits d'affichage vidéo. Il est bien sûr quasi certain que le disque dur de ce serveur est également corrompu (comme pour le serveur de messagerie).
D) Les données situées sur disque dans la zone altérée (2048 premiers blocs) sont quasi définitivement perdues. Dans le cas d'un disque dur de grande capacité, il est possible que la copie de la table d'allocation ne soient pas située dans les 2048 premiers blocs, dans ce cas, on peut éventuellement récupérer cette information (avec un outil de réparation de système de fichiers) et récupérer dans ce cas les fichiers situés au-delà de la limite de 2048 blocs. Dans

tous les cas, il semble plus raisonnable de s'appuyer directement sur les sauvegardes hebdomadaires si c'est possible. Malheureusement, pour un serveur de messagerie, cela peut signifier dans le pire cas jusqu'à 6 jours de courrier électronique perdu. Avec de la chance ce sera peut-être moins. Dans tous les cas, une fois la restauration effectuée, ne pas oublier de commencer par utiliser l'antivirus pour éviter de tenter la chance une nouvelle fois l'année prochaine et augmenter la fréquence des sauvegardes pour ce type de données (volatiles) pour l'éviter également face à d'autres menaces.

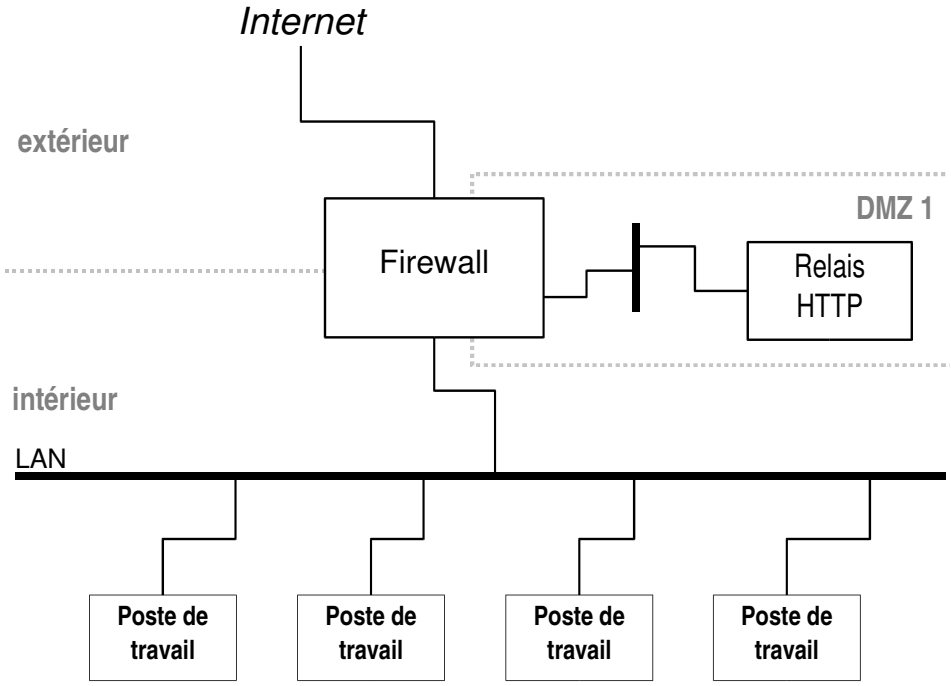
E) A priori, il faut commencer par restaurer le BIOS du serveur. Ceci peut être plus ou moins difficile à réaliser suivant la configuration matérielle de la machine: à cette époque, certains BIOS pouvaient être réinitialisés à une valeur constructeur en manoeuvrant assez simplement un jumper sur la carte mère, d'autres ne pouvaient pas sans remplacement d'un circuit. Dans tous les cas, si cette étape est franchie, on se trouve dans une situation similaire à celle du serveur de messagerie précédent. Compte tenu de l'ampleur du travail nécessaire, pour des données peu volatiles, l'utilisation d'une sauvegarde datant d'au maximum 6 jours peut être préférable. Hors tout paramètre financier, dans ce cas, il peut également être aussi simple de déployer un nouveau serveur. Dans tous les cas, penser à utiliser l'antivirus une fois la restauration effectuée.

D&E) Dans les 2 cas, si les données situées sur le disque dur des machines corrompues revêtent une importance critique et n'ont pas été sauvegardées, il faudra envisager de faire appel à des sociétés spécialisées. Par contre, les coûts associés à ce type de prestation (sans obligation de résultat) sont approximativement de l'ordre de grandeur des machines complètes elles-mêmes (neuves).

F) Il est effectivement assez difficile pour un administrateur de se prémunir spécifiquement contre les codes malveillants visant les BIOS des machines. Cette situation s'est d'ailleurs compliquée avec les systèmes récents qui permettent communément des mises à jour dynamiques (légitimes ou illégitimes) des BIOS et même plus (BIOS des cartes additionnelles, voire microcodes des processeurs). Il appartient surtout aux constructeurs de prévoir ces vulnérabilités dans la conception de leurs systèmes. Au niveau de l'utilisation, on peut surtout envisager de considérer ce critère de sécurité (parmi d'autres) pour le choix des plate-formes matérielles.

Exercice 3 *(2 points)*

On considère le réseau décrit dans la figure ci-après. Le relais HTTP (*proxy*) écoute sur le port 8080. Quels sont les traitements que doivent effectuer le pare-feu et le *proxy* pour offrir un service de *proxy* HTTP transparent ?



Afin que les postes de travail internes accèdent à HTTP sur Internet en bénéficiant d'un relayage transparent de ce flux via le relais HTTP, le firewall doit rediriger les connexions TCP sortantes associées au port HTTP par défaut (80) vers le relais HTTP (normalement associé au port 8080). Le firewall doit donc effectuer à la fois une redirection mais éventuellement une translation de port pour les paquets sortants. Toutefois, pour que le système fonctionne bien, le relais HTTP doit également être prévu et paramétré de manière à prendre en compte ces flux particuliers. En effet, plusieurs scénarios sont envisageables:

- *le firewall peut envoyer les paquets au relais sans modifier les adresses IP source et destinataires contenues dans les paquets (en modifiant l'adresse MAC des paquets), mais dans ce cas le système d'exploitation du relais doit être spécifiquement prévu pour accepter des paquets IP ne correspondant pas à sa propre adresse et les envoyer au logiciel du relais HTTP ;*
- *le firewall peut effectuer une translation d'adresse et de port plus classique (du point de vue du relais), mais dans ce cas, il doit également modifier le contenu de la requête du client pour y inclure le nom du serveur demandé (en effet, la requête HTTP classique est du type GET /page.html HTTP/1.0 et doit être traduite en GET mon.serveur.fr/page.html HTTP/1.0 pour que le relais puisse l'honorer) ; ceci implique également que le relais sait qu'il est utilisé en mode transparent ;*
- *le firewall peut également encapsuler le paquet d'origine dans un autre paquet IP (de manière à préserver l'information d'origine).*

Ces différentes difficultés conduisent généralement à mettre en oeuvre un relais transparent (pour HTTP ou pour un autre protocole d'ailleurs) directement sur le firewall. Éventuellement, il peut s'agir d'un relais simplifié (dédié à la mise en oeuvre transparente)

s'appuyant sur une deuxième relais HTTP conventionnel (avec cache et filtrage d'URL par exemple).

Exercice 4 *(2 points)*

Les traces d'un serveur Web contiennent une série de lignes du type suivant (mentionnant l'adresse source, la date, la requête, le résultat et le nombre d'octets transférés) :

```
128.178.146.216 - - [24/Sep/2003:16:50:42 +0200] "GET /
default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXX%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%
u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%
uu531b%u53ff%u0078%u0000%u00=a %HTTP/1.0" 404 209
```

1. De quoi peut-il bien s'agir ?
2. Comment faudrait-il configurer un système de détection d'intrusion pour détecter ces attaques ?
3. En supposant que le serveur Web se trouve derrière un proxy inverse, comment le *proxy* pourrait-il aider à éviter ces attaques ?
4. Laquelle de ces solutions, IDS ou *proxy* inverse, est-elle préférable ?

1) Il s'agit visiblement d'un usage anormal du serveur Web. Il s'agit d'un appel à une page particulière correspondant à un script auquel est passé un paramètre anormal dans sa longueur et dans son contenu. La longueur du paramètre est délibérément rallongée par des caractères inutiles répétés un grand nombre de fois (les X). Ceux-ci sont suivis de caractères utilisant un encodage peu courant (probablement mal interprétés par 'default.ida'). Ce script est certainement associé à une vulnérabilité connue (consultez l'alerte Microsoft MS01-033), et il s'agit d'une tentative d'exploitation (peut-être celle d'un ver courant à une époque nommé CodeRed qui utilisait cette faille pour se propager).

2) Le nom du script, la séquence longue de caractères (qu'il s'agisse de 'X' ou peut-être d'autres caractères) ainsi que les codes de contrôle qui les suivent forment une signature assez spécifique permettant à un système de détection d'intrusion de repérer ce type d'attaque, s'il est en mesure de suivre et de surveiller un flux de requête HTTP.

3) Pour prévenir ce type d'attaque, un proxy inverse peut essayer de normaliser les requêtes qu'il relaye. Notamment, l'utilisation de paramètres anormalement longs pour des scripts peut faire l'objet d'un filtrage qui limite la faisabilité de ces attaques. (NB: Limiter la longueur totale des URL est une autre possibilité, plus simple, mais qui s'avère souvent inutilisable dans la pratique car de nombreux environnement de développement Web actuels utilisent des URL de grande taille. Toutefois, ceux-ci sont généralement dus à la présence de nombreux paramètres et non de paramètres de grande taille.)

4) L'avantage d'une solution basée sur un proxy est qu'elle empêche l'attaque d'être effective, même sur un serveur vulnérable. Elle semble préférable dans l'absolu, sans contrainte de mise en oeuvre. Il faudrait d'ailleurs lui préférer la mise en oeuvre d'un serveur HTTP intégrant nativement lui-même de telles protections. Enfin, on notera que l'utilisation d'un IDS peut être envisagée sans modifier l'architecture fonctionnelle du service HTTP et cette solution peut être plus facile à mettre en production.