TP - ASR5 système d'exploitation

Un peu de système

9 avril 2018

Pour ce TP, nous avons créé plusieurs machines virtuelles sur lesquelles vous allez vous connecter. Votre charger de TP vous indiquera l'adresse de celle que vous devez utiliser. Ces machine sont connues par leur adresse IP, dans le sujet, celle-ci est nomée `IP_VM`. Ces machines sont reliées à l'anuaire de l'université, pour vous connecter vous allez donc utiliser votre login de l'université et votre mot de passe. Dans la suite du TP, votre login sera noté `UTILISATEUR`.

Contrairement à ce qui était anoncé, il n'y aura plus de TP noté. cette séance et la suivante portera sur l'utilisation du sytème linux. Mais prenez des notes, lors du dernier contrôle, certaines questions porteront sur l'utilisation des commandes vues aujourd'hui et la prochaine fois.

I Connexion sur les machines

Le logiciel `ssh`, pour `secure shell`, permet de se connecter sur une machine distante afin de l'utiliser ou de l'administrer. Vous avez sans doute déjà utilisé la connexion en fournissant vos identifiants par exemple sur le serveur d'accès de l'université `linuxetu.univ-lyon1.fr`. Nous utiliserons `ssh` via *login/password*, mais `ssh` permet aussi une connexion utilisant un couple clef publique/clef privée, ce qui a deux intérêts :

- permettre la connexion sans utiliser de mot de passe (donc automatique) ;
- ne fournir que la clef publique à la machine cible (donc pas de transmission d'informations sensibles à une machine distante).

Les VMs sont générées à partir d'un modèle configuré, et vous pouvez utiliser vos login et mot de passe de l'université pour y accéder.

Q.I.1) - Connectez-vous en utilisant la commande `ssh p...@VM_IP`.

Q.I.2) - Il est directement possible de lancer une application graphique qui s'exécute sur la VM mais s'affiche sur votre PC. Il suffit d'ajouter deux options à la commande : `ssh -XC p...@VM_IP`. Testez la commande et lancez depuis la VM un logiciel comme `geany` ou `codeblocks`.
- Que sighifient les 2 options `-X` et `-C` ?
- Comment vérifier que le logiciel est exécuté sur la machine virtuelle ?

> **Solution:** −X export graphique et −C compression du flux ce qui accélère la réactivité des applications (le point le plus bloquant ici est le réseau).
>
> Il suffit de regarder les fichiers que peut ouvrir l'application, on s'appercoit que c'est sur le disque de la VM. Sinon, il y a bien sûr les classiques comme ps.

I.1 Si vous voulez aller plus loin :

Pour utiliser une clef d'authentification, vous devez d'abord en créer une. En CLI sous linux, vous devez utiliser la commande `ssh-keygen` : elle vous demandera un nom de fichier et une passphrase. Cela crée 2 fichiers, par défaut `${HOME}/.ssh/id-rsa` et `${HOME}/.ssh/id-rsa.pub` qui contiennent respectivement la clef privée et la clef publique. Il suffit d'écrire la clé publique sur les machines distantes, dans le fichier
`${HOME}/.ssh/authorized_keys` ou d'utiliser la commande `ssh-copy-id`.

I.2 Et sous windows :

Vous pouvez télécharger et utiliser `PuTTY`, qui permet de se connecter à distance. Le programme `PuTTYgen` permet de gérer et d'utiliser les clés. Par contre, pour afficher une fenêtre distante, vous devez avoir un serveur graphique compatible installé. C'est lui qui gèrera la gestion des fenêtres et renverra les évènements aux applications lancées sur la machine virtuelle. Vous pouvez utiliser `Cygwin-X` (qui est une collection d'outils du monde Linux fonctionnant sous windows) ou des outils comme MobaXterm qui a l'avantage d'être utilisable sans installation (donc sur les machines de l'université).

II Copie de fichiers

Vous devez apprendre à utiliser ssh pour copier des fichiers à distance. Par exemple en copiant le fichier `exemple.cpp` depuis votre PC vers la VM.

Q.II.1) - Faites le en utilisant la ligne de commande, via `scp` ou `pscp` (`PuTTY`).

Pour copier machin dans le repertoire /tmp/ de la machine virtuelle : scp −Cr machineacopier UTILISATEUR@VM_IP:/tmp/ l'option −C permet de compresser et l'option −r de copier de manière récursive un répertoire.

Q.II.2) - Faites le en utilisant une interface graphique :
- sous linux utilisez le gestionnaire de fichiers, demandez-lui d'aller à l'adresse `sftp://UTILISATEUR@VM_IP`.
- sous windows utilisez par exemple `WinSCP` (en version portable) ou `MobaXTerm`.

III Exécution de processus à distance

Assez souvent, lorsqu'on lance un programme sur un serveur distant, c'est dans le but de le laisser s'exécuter sans devoir rester connecté. Cela n'est pas si facile.

En effet, vous constaterez que la fermeture d'une connexion distance provoque souvent la fermeture des processus qui ont été lancé par cette dernière.

Si ce n'est pas déjà fait, copier le fichier exemple.cpp sur la machine. Compilez le code par la commande :

```
g++ -g -Wall exemple.cpp -o exemple.exx
```

Q.III.1) - Lancez le programme en redirigeant la sortie vers un fichier (`./exemple.exx > res &`).

Vérifiez qu'il fonctionne bien (il écrit régulièrement dans le fichier `res` ce que l'on peut voir par la commande `tail -f res`.

Q.III.2) - Déconnectez-vous puis reconnectez-vous. Le processus tourne-t-il toujours ? Que lui est-il arrivé ?

> **Solution:** En regardant dans le fichier `res` on s'appercoit que le procesus a ecu un signal : `SIGHUP`. Il est soit envoyé par `ssh` lorsque ce dernier se termine (après suppression de la session ssh), soit envoyé par `bash` lorsque vous tapez sur exit.
>
> *Attention, le comportement de bash ici n'est pas standard. Cela a été modifié pour les besoin du TP.*

Q.III.3) - Comment assurer que le processus se poursuivent malgrès la déconnexion. Vous pouvez regarder la documentation des commandes `nohup`, `screen` ou `tmux`.

> **Solution:** `nohup` est une commande simple qui masque juste le signal `SIGHUP`. `screen` et `tmux` sont plus évoluées. Elles permettent de lancer des terminaux virtuelles qui peuvent être détachés et réattachés plus tard.

Q.III.4) - Lancez un terminal `screen` dans lequel vous lancez le programme `exemple.exx`. Déconnectez-vous de la machine virtuelle. Reconnectez-vous depuis la machine de votre voisin (avec votre login) et réattacher le terminal virtuel pour voir son évolution. Screen permet en effet de partir et de se reconnecter plus tard depuis un autre endroit.

Q.III.5) - Donnez un exemple d'utilisation des terminaux virtuels.

IV Débordement de mémoire

On parle de débordement de buffer (`buffer overflow`) lorsque les données fournies à un logiciel sont trop volumineuses pour être contenues dans le buffer qui est censé les contenir. Ce type d'erreur cause souvent une erreur de segmentation, mais ce n'est pas toujours le cas. La mémoire est modifiée d'une manière que le programmeur n'a pas prévu. En choisissant judicieusement la modification, l'utilisateur peut détourner un programme de sont but et peut ainsi acquerir des droits qu'il ne devrait pas avoir.

Pour ce genre d'attaque, il suffit de connaître un problème de dépassement de capacité qui a lieu sur une commande du système. Si cette dernière a des droits particuliers (par exemple le SUID bit), ce qu'on fait faire à cette commande est fait avec des droits étendus. Une commande système en général fait des vérifications avant d'autoriser une action quelconque, mais si le débordement de buffer permet d'ignorer ces vérification, tout est possible.

Aujourd'hui, vous allez détourner une commande proche de la commande `su` qui permet de changer l'utilisateur courant. Normalement, cette commande vérifie le mot de passe, mais vous allez faire en sorte que cette vérification soit inutile.

Sur la machine virtuelle, nous avons installé la commande `/usr/bin/ChangeUser.ex` dont le code aussi fourni avec le sujet sur le site de l'UE. Comme `su`, cette commande permet de changer l'utilisateur courant en fournissant son mot de passe. Elle est mal programmée et permet via un débordement de buffer de changer l'utilisateur sans connaître le mot de passe.

Le code c fourni doit être compilé avec la commande

```
gcc -g -Wall ChangeUser.c --param ssp-buffer-size=200 -fno-stack-protector
                                                      -lcrypt -o ChangeUser.ex
```

Il ne peut pas être réellement fonctionnel sans appartenir à l'administrateur et avoir le fameux bit *setuid*. Par contre, on peut étudier son fonctionnement avec un débugger comme kdbg ou gdb.

Certaines des commandes suivantes vont vous permettre de changer d'utilisteur. Pour vérifier le changement, vous pourrez remarquer que l'invite de commande change. Il est aussi possible d'utiliser la commande whoami pour cela.

Q.IV.1) - Vérifiez le fonctionnement de la commande ChangeUser.ex :

- Utilisez la commande que vous avez compilé `./ChangeUser.ex chaprot`, cela ne doit pas fonctionner avec une erreur `Permission denied`. Qu'est-ce qui est interdit ?
- Utilisez la commande installée sur la machine ChangeUser.ex chaprot, cela doit fonctionner (mot de passe gotlib).
- Copiez la commande qui fonctionne et réessayer, est-ce que cela fonctionne ?

  Solution: Non, on retrouve les même erreurs qu'avec le code compilé.

- Pourquoi cela fonctionne-t'il dans un cas et pas dans l'autre ?

  Solution: La différence est le SETUID bit et l'utilisateur. Dans la commande installée, le programme peut être lancé par tout le monde mais ses droits sont ceux de l'administrateur. La commande peut donc changer lire le mot de passe, changer l'utilisateur... Le même exécutable sans les mêmes droits ne le peut pas. Or quand vous copier la commande, vous modifiez le propriétaire.

Q.IV.2) - En étudiant la commande grâce au débugger, devenez root. Dans la fonction `verif()`, vous pouvez remarquer :

- que l'utilisateur entre un mot de passe et que la lecture (par un simple scanf()) n'est pas sécurisée ;
- que si le mot entré est trop long, ce que l'utilisateur tape va s'écrire sur les variables locales de la fonction `verif()` ;
- que si l'utilisateur parvient à modifier la valeur de la variable `res`, le résultat de cette fonction sera vrai *quoi qu'il se passe ensuite.*

Solution: Avec le débugger, on peut voir l'adresse du tableau pass[] et celle de la variable res. Si on écrit un mot de passe suffisamment grand, les données lues vont remplir le tableau pass[] puis toutes les cases mémoire jusqu'à celle de la variable res. Comme res est mis à 0 au début de la fonction et qu'elle n'est pas modifiée ensuite, il suffit de lui donner n'importe quelle valeur non nulle pour que le programme la considère comme vraie. Donc il suffit de donner un mot de passe de la bonne taille, contenant n'importe quel caractère et sans espace. Attention, il faut que la taille soit assez précise. Trop long, il va changer l'adresse de la variable de retour de la fonction et le programme va faire une erreur. Pas assez long, la variable res ne sera pas changée.
Sur ubuntu 16.04 avec des machines 64 bits, il faut 75 carractère dans le mot de passe. Un bon moyen pour trouver le bon nombre est de le faire par dichotomie. En effet, l'erreur si le mot de passe est trop court n'est pas la même que celle si le mot de passe est trop long.

Q.IV.3) - Pour garder la possibilité de passer administrateur même si le bug est corrigé, installer une *backdoor*. Sous ubuntu, les utilisateur du groupe `admin` sont autorisés à taper n'importe quelle commande via la commande `sudo`. Pour conserver votre accès pour le prochain TP, ajouter votre utilisateur au groupe admin (voir la commande `addgroup`).

Solution: Il y a plusieurs moyen d'installer une backdor, par exemple vous pouvez :
- Ajouter votre utilisateur au groupe `admin` addgroup UTILISATEUR admin
- Modifier la configuration de sudoers pour vous octroyer le droit de tout faire (regarder par exemple le fichier /etc/sudoers.d/90−cloud−init−users.
- Créer un code `ChangeUser` sans aucune vérification, le compiler et changer le propriétaire du fichier pour root ainsi que le *Setuid bit*

La denière solution a l'avantage d'être plus discrette, mais certains utilitaire de sécurité vont justement rechercher les fichier avec le *Setuid bit* activé.

IV.1 Note à ceux qui souhaitent utiliser leur propre machine

Le code utilisé est très simple et donc très sensible aux contres-mesures mises en place par les compilateurs pour éviter ce genre de problèmes. C'est pourquoi, si vous souhaitez utiliser votre ordinateur personnel, il est probable que cela ne fonctionne pas. Notamment, certains compilateurs (gcc sous ubuntu par exemple) ajoutent des protections lorsqu'on utilise des buffers dans la pile. Il est nécessaire de les désactiver pour obtenir l'effet recherché. De plus, il faut avoir les droits d'administration pour mettre en place la commande ChangeUser.ex.

- Téléchargez le code et compilez le par la commande :

```
gcc -g -Wall ChangeUser.c --param ssp-buffer-size=200 -fno-stack-protector
                                                      -lcrypt -o ChangeUser.ex
```

- Pour qu'il fonctionne vous devez ensuite changer le propriétaire et les droits du fichier exécutable :
  - `su root` (ou sudo...)
  - `chown root:root  ChangeUser.ex`
  - `chmod a+x ChangeUser.ex`
  - `chmod u+s ChangeUser.ex`
  - `logout`
- La deuxieme condition est que l'utilisateur cible ait un mot de passe (par défaut root n'en a pas sous MacOS et Ubuntu).
- Vérifiez le bon fonctionnement du logiciel. La commande suivante devrait vous permettre de passer administrateur en donnant le mot de passe de l'administrateur :
  - `ChangeUser.ex root`

Attention, une fois installée, cette commande permet de passer root sans connaître le mot de passe. Pensez à l'effacer !