INSA Bourges
4A STI 2014/2015

TP Sécurité Unix

Utilisez une machine virtuelle, idéalement *Debian* ou *Ubuntu*, sur laquelle vous vous connecterez en tant que simple utilisateur (et non *root*). Votre utilisateur devra cependant être *sudoer*.

Pour la préparation des TP, il faudra installer quelques packages :

- configurez l'interface réseau en mode *NAT* ;
- installation et mise à jour des packages:
  - **sudo apt-get update**
  - **sudo apt-get install aptitude libc6-dev-i386 make autopsy manpages openssh-server screen vim**
  - **sudo aptitude safe-upgrade**
- reconfigurez l'interface réseau en mode *Host-only*.

Récupérez et désarchivez les TP :

- depuis votre station hôte :
  - **TP_4ASTI.tgz**
  - **scp TP_4ASTI.tgz <user>@<adresse VM>:**
- sur la machine virtuelle :
  - **tar xvzf TP_4ASTI.tgz**

TP 1 : Buffer Overflow, Stack Overflow et protections système

Répertoire **TP_4ASTI/1.Buffer_Overflow**

Ce TP consiste à observer et comprendre le fonctionnement de la mémoire sous Linux afin de voir ensuite comment une mauvaise gestion de celle-ci peut mener à l'exécution de code arbitraire.

Il se découpe en 4 phases (répertoires Exo1 à Exo4).

Commencez par exécuter la commande **make** à la racine de ce TP, puis procédez répertoire par répertoire en suivant les instructions qui suivent.

1. Exo1
    1. Lisez le code de **vulnerable.c**
    2. Quel problème remarquez-vous dans la gestion de la mémoire dans ce code ?
    3. Compilez les 2 codes (**make all**).
    4. Lancez **vulnerable** avec un argument permettant d'afficher la chaîne de caractères "ENSIB" dans *buf1*.
    5. Lisez et analysez le code de **exploit.c** ; schématisez la construction de *large_string*.
    6. Utilisez les informations obtenues en étudiant le code des deux sources pour construire *large_string* de façon à pouvoir exploiter la vulnérabilité mise en évidence à la question 2.

Indication pratique : pour exploiter la vulnérabilité, lancez :
./vulnerable `./exploit <argument>`

2. Exo2
    1. Compilez les deux programmes avec la commande **make**.
    2. Comparez le **Makefile** de ce répertoire avec le précédent : qu'observez-vous ?
    3. Tenter d'exploiter la vulnérabilité comme précédemment : que se passe-t-il ?

3. Exo3
    1. Compilez les deux programmes avec la commande **make**.
    2. Comparez le **Makefile** avec celui du premier exercice : qu'observez-vous ?
    3. Lancez à plusieurs reprises le programme **vulnerable**, avec le même argument à chaque fois : qu'observez-vous, et que pouvez-vous en déduire ?

4. Exo4
    1. Lisez le code de **vulnerable.c** qui diffère des précédents.
    2. Tentez d'exploiter la vulnérabilité comme dans le premier cas : que se passe-t-il, et comment l'expliquez-vous ?

Résumez les protections génériques rencontrées au cours de ce TP.

TP 2 : Format Strings

Répertoire **TP_4ASTI/2.Format_Strings**

Le but de ce TP est d'exploiter une vulnérabilité dans un programme, en explorant la pile et en écrivant dans la mémoire du processus.

Compilez tout d'abord le code avec la commande **make**.

Lisez le code du fichier **vulnerable_format_string.c** : identifiez la vulnérabilité.

1. Exploration de la pile

Quelles informations sur le processus pouvez-vous extraire en choisissant un argument adapté ?

Lancez le programme avec l'argument *"%d %d"* :
./vulnerable_string_format "%d %d"

A quoi correspond la ligne renvoyée sous *"Affichage de la valeur de 'text' :*" ?

Affichez maintenant, via le même procédé, les variables du programme *a*, *b*, *c*, *d*.

Affichez ensuite uniquement l'adresse de la variable *c*, toujours avec la même méthode.

Indication pratique : dans une directive de formatage, l'élément à la position *n* dans la pile peut être spécifié avec le modificateur *n$*.
Exemple : *%5$s* permettra d'afficher sous forme de chaîne de caractères le 5ème élément de la pile. Dans *bash*, protégez le symbole *$* : il faudra écrire *%5\$s*.

2. Ecriture dans la pile

Quelle directive de formatage vous permet d'écrire dans la pile en exploitant ce genre de vulnérabilité ?

Remplacez la valeur de la variable *a* par une valeur arbitraire.

En utilisant la méthode précédente basée sur *n$*, attribuez la valeur *4* à la variable *b*.

Attribuez ensuite la valeur *1024* à la variable *c* : quelle est la taille du buffer *text* ? comment contourner cet obstacle ?

Indication pratique : la directive de formatage utilisée fait référence au nombre de caractères que l'on s'attend à afficher, et non au nombre de caractères réellement affichés.

Réactivez ensuite l'adressage aléatoire : **make random**.
Quelles conséquences cette protection a-t-elle sur l'exploitation de la vulnérabilité ?

TP 3 : Forensics

Répertoire **TP_4ASTI/3.Forensics**

L'objectif de ce TP est de pratiquer une étude sur un système de fichiers provenant d'une machine corrompue.

1. Lancez **autopsy** en tant que root, en autorisant la connexion depuis votre machine locale :
   sudo autopsy <@IP de votre station sur le réseau virtuel>

Connectez-vous à l'adresse renvoyée.

2. Dans l'interface web d'**autopsy**, vous allez créer le cas d'étude en suivant les étapes suivantes :
   1. Cliquez sur *New Case* ;
   2. Attribuez un nom à ce cas et cliquez sur *New Case* ;
   3. Cliquez sur *Add Host* ;
   4. Attribuez un nom d'hôte à la machine concernée puis cliquez sur *Add Host* ;
   5. Cliquez sur *Add Image* ;
   6. Cliquez sur *Add Image File* ;
   7. Entrez le chemin absolu du fichier *compromis.img* dans le premier champ, sélectionnez *Partition* pour le type d'image, et *Copy* pour importer l'image. Cliquez sur *Next* ;
   8. Choisissez ensuite *Calculate* pour la somme de contrôle du fichier, puis cliquez sur *Add* ;
   9. Cliquez sur *OK* ;
   10. Cliquez sur *Analyze*.
   11. Cliquez sur *File Analysis*.

3. Nous allons étudier le contenu du répertoire *willy*.

Indication pratique : la section *Help* vous sera très utile, notamment la partie *File Analysis*.

1. Que pouvez-vous dire des répertoires *adore* et *.adore* ?
2. Quels fichiers ont été supprimés ?
3. Dans le répertoire *.adore*, quand le programme *ava* a-t-il exécuté pour la dernière fois ? Quel(s) événement(s) observez-vous au même instant ?
4. A partir des observations précédentes, et du code contenu dans l'archive *adore-0.42.tgz* que vous analyserez, expliquez le rôle du fichier *.adore/133trul3z*. L'archive se trouve dans *3.Forenscis/tmp*.
5. Le code de ce rootkit avait été effacé par l'intrus. Que pouvez-vous dire de l'autre fichier effacé, en observant les informations fournies par **autopsy** ? Comment pouvez-vous obtenir des informations sur le fichier *a.out* si vous montez le système de fichiers sur une machine d'analyse ?
6. Quelles opérations pouvez-vous tenter sur *compromis.img* pour tenter de retrouver le code effacé ?