TP 4 : chroot

1 Introduction

Dans ce TP vous allez mettre en place un serveur `thttpd` (*tiny/turbo/-throttling HTTP server*) servant un script CGI (*Common Gateway Interface*). Vous exploiterez une vulnérabilité de ce script. Cette vulnérabilité permettera de voir l'intérêt de se placer dans un environnement cloisonné. Vous placerez alors le serveur HTTP dans un environnement cloisonné afin de contenir la vulnérabilité. À noter, dans ce TP, lorsque l'on parle d'environnement cloisonné on parle seulement de cloisonnement au niveau du système de fichier.

2 Environnement de travail

— Installez `thttpd` :

```
wget http://www.acme.com/software/thttpd/thttpd-2.29.tar.gz
tar xf thttpd-2.29.tar.gz
cd thttpd-2.29/
sudo /usr/sbin/groupadd www
sudo mkdir /usr/local/man/man1
./configure
make
sudo make install
/usr/local/sbin/thttpd -V
```

— Récupérez les fichiers qui seront servis par notre serveur `thttpd` (normalement envoyés par mail) :
  — index.sh
  — README.html
— Placez ces fichiers dans un répertoire de votre choix (dans la suite du TP ce répertoire sera `~/web_dir`).

3 Utilisation sans protection

— Lancez `thttpd` depuis le répertoire `~/web_dir` :

```
user@debian:~/web_dir$ sudo /usr/local/sbin/thttpd -D -u root -c index.sh \
  -nor
```

- À quoi servent les différentes options présentes sur la ligne de commande ?
- En quoi reflètent-elles correctement le titre de cette partie du TP ?
- Naviguez vers `index.sh` sur le serveur HTTP que nous venons de lancer avec l'outil de votre choix (`curl`, `wget`, `firefox`...).
- Faites une requête faisant afficher le contenu de `index.sh`. Cela vous donne-t-il une idée de faille ?
- Parcourez son code source et mettez en évidence la vulnérabilité. Exploitez-là.
- En tant qu'attaquant, quels fichiers intéressants consulteriez-vous grâce à elle ?
- Que feriez-vous pour y remédier ?

4 Construction d'une cage

- Construisez une *chroot jail* autour du daemon `thttpd` (pensez à utiliser `ldd` et `strace`).
- Relancez `thttpd` en vous plaçant à la racine de votre *chroot jail*.

```
sudo chroot . bin/thttpd -D -u root -c "index.sh" -nor -d \
        var/www -i /var/run/thttpd.pid
```

- En quoi cette nouvelle commande reflète le titre de cette partie du TP ?
- Essayez d'exploiter la vulnérabilité à nouveau. que remarquez-vous ?
- Le resultat attendu est-il celui escompté ? Était-ce prévisible ?
- Quels sont les avantages et inconvénients de cette méthode ?

5 La chroot jail applicative

- Relancez `thttpd` avec ces nouveaux paramètres (hors *chroot jail*) :

```
sudo thttpd -D -u root -c "index.sh" -r -d var/www
```

- Expliquez les nouveaux paramètres passés à `thttpd` en ligne de commande.
- Vérifiez que l'option `-r` fait effectivement ce qu'elle prétend faire. (indice : `chroot` n'est pas seulement une commande, mais aussi le nom du *syscall*)

— Essayez d'exploiter la vulnérabilité à nouveau. Que remarquez-vous alors ?
— Quels sont les avantages et inconvénients de cette méthode ?
— À quel(s) autre(s) daemon(s) appliqueriez-vous cette méthodologie ?

Chroot n'est pas parfait

Comme nous l'avons vu rapidement en cours, *chroot* présente plusieurs limitations du point de vue de la sécurité, notamment il est possible de s'échapper d'une *chroot jail*. Cependant, pour la vulnérabilité que nous avons vu dans ce TP (*directory traversal*) *chroot* est efficace.
Une solution plus simple (et plus "moderne") serait de faire tourner l'application dans un conteneur. Mais est-ce une bonne barrière de sécurité ?