

SEGURIDAD EN SISTEMAS OPERATIVOS

4º Grado en Informática

Curso 2017-18

Práctica 1.- Administración de la seguridad en Linux

Sesión 4.- AppArmor.

1.- AppArmor

AppArmor es un Control de Acceso Obligatorio (MAC), o sistema de prevención de intrusiones, para Linux, basado en LSM, que confina programas individuales a un conjunto de archivos, capacidades, accesos de red y límites de recursos (*rlimits*), conocidos colectivamente como *política para el programa* o simplemente *perfil* (conjunto de reglas que determinan qué puede hacer un programa y qué no). Es decir, liga los atributos de control de acceso a los programas en lugar de a los usuarios.

La política de un programa puede modificarse en tiempo de ejecución. Esta confinación es selectiva en tanto en cuanto podemos confinar solo los programas seleccionados. Así, los programas sin confinar se ejecutan bajo la seguridad del sistema DAC (*Discretionary Access Control*) de Linux. AppArmor aumenta el sistema tradicional DAC para los programas confinados en tanto que estos son primero evaluados con DAC y si DAC permite el acceso entonces se consulta la política AppArmor.

Un perfil de AppArmor pueden estar en uno de los dos modos siguientes:

- *Enforcement* – un perfil cargado en este modo provoca que se aplique la política definida en el perfil y además informa de los intentos de violación (via *syslog* o *auditd*).
- *Complain* – no hace cumplir la política pero informa de los intentos de violación de la misma.

A diferencia de otros sistemas DAC: AppArmor está basado en *path* y permite la mezcla de perfiles *enforcement* y *complain*, utiliza archivos *includes* para facilitar el desarrollo, y es más fácil de utilizar que otros sistemas. Actualmente está integrado en Ubuntu, Novell/Suse y Mandriva.

AppArmor consta de (Figura 1):

- Una biblioteca de perfiles comunes para aplicaciones Linux, que describen a qué archivos necesitan acceder los programas.
- Una biblioteca de clases básicas de perfiles (bloques de construcción de perfiles) necesaria para las actividades de las aplicaciones comunes tales como búsquedas DNS o autenticación de usuarios.
- Un conjunto de herramientas para desarrollar y mejorar perfiles, de forma que podamos cambiar los perfiles existentes para adecuarlos a nuestras necesidades y crear nuevos perfiles para aplicaciones particulares.
- Varias aplicaciones específicamente modificadas habilitadas para AppArmor para suministrar una seguridad mejorada en la forma de confinamiento único de subprocesos, incluyendo Apache.

- En la distribución SUSE, módulos de carga AppArmor de Novell y guiones de control asociados para aplicar las políticas AppArmor.

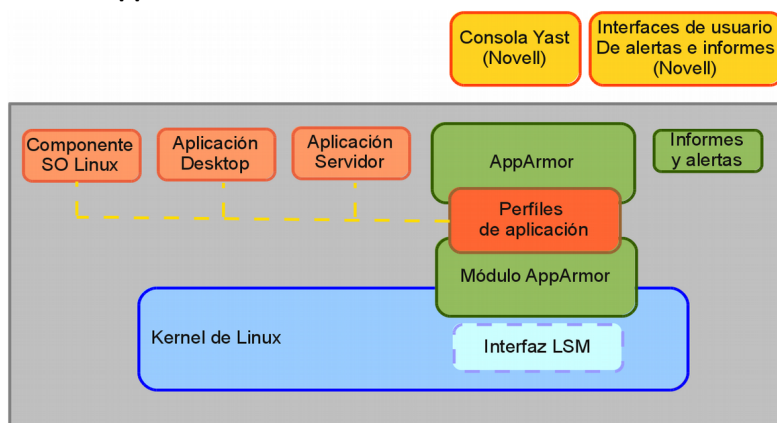


Figura 1.- Estructura de AppArmor.

Las órdenes que disponemos para controlar la seguridad, agrupadas por tareas, en AppArmor son:

Aplicación de perfiles	
<code>apparmor-parser</code>	Carga un perfil en el kernel.
<code>aa-audit</code>	Establece el modo auditoria para un perfil.
<code>aa-enforce</code>	Pone un perfil de AppArmor que esta deshabilitado o en modo <i>omplain</i> , en modo <i>enforce</i> .
<code>aa-complain</code>	Pone un perfil en modo complain.
Herramientas de monitorización	
<code>aa-status</code>	Muestra información sobre la política actual de AppArmor.
<code>aa-notify</code>	Muestra información sobre los mensajes de log de AppArmor.
<code>aa-unconfined</code>	Muestra la lista de procesos con puertos tcp o udp que no tienen cargados un perfil.
Desarrollo de perfiles	
<code>aa-autodep</code>	Estima los requisitos de un perfil básico.
<code>aa-logprof</code>	Programa de utilidad para gestionar perfiles de seguridad AppArmor.
<code>aa-genprof</code>	Utilidad de generación de perfiles.
<code>mod_apparmor</code>	Confinamiento de grano fino para Apache.
<code>aa_change_hat</code>	Cambiar de “sombrero” dentro de un perfil.
<code>aa_change_profile</code>	Cambia un perfil de tarea.
<code>PAM plugin</code>	Módulo PAM (<code>pam_apparmor</code>) que puede utilizarse para ligar perfiles basados en una autenticación realizada a nivel de usuario o tarea. Puede utilizarse en varios casos, como por ejemplo, RBAC.

Ejercicio 4.1.- Determinar los perfiles activos en la distribución Linux de tu equipo. Elige uno de los perfiles y analiza sus características..

2.- AppArmor en Ubuntu

En las distribuciones actuales de Ubuntu, AppArmor está instalado y se carga por defecto. Algunos paquetes poseen sus propios perfiles *enforcing*. Podemos encontrar perfiles adicionales en el repositorio Universe instalando *apparmor-profiles*. Los perfiles no activados están en la carpeta */etc/apparmor.d/disable*.

Si no estuviese instalado el paquete *apparmor-utils*, deberíamos instalarlo para tener acceso a las órdenes *aa-logprof*, *aa-status*, *aa-autodep*, *aa-enforce* y *aa-complain*.

A continuación veremos las operaciones comunes, que debemos ejecutar en un terminal.

- Listar el estado actual de AppArmor

```
% sudo apparmor_status
```

- Poner un perfil en modo *complain*: *sudo aa-complain /path/to/bin*

Por ejemplo, *%aa-complain /bin/ping*

- Poner todos los perfiles en modo *complain*

```
% sudo aa-complain /etc/apparmor.d/*
```

- Poner un perfil en modo *enforce* con *aa-enforce path_to_bin*

```
% sudo aa-enforce /bin/ping
```

- Poner todos los perfiles en modo *enforce*

```
% sudo aa-enforce /etc/apparmor.d/*
```

- Poner todos los perfiles en modo *enforce* salvo uno

```
GLOBIGNORE="*profile.name"
sudo aa-enforce /etc/apparmor.d/*
```

Por ejemplo:

```
GLOBIGNORE="*bin.ping"
sudo aa-enforce /etc/apparmor.d/*
```

- Deshabilitando el marco AppArmor

Los sistemas no necesitan de forma general deshabilitar completamente AppArmor. Lo normal es que esté habilitado y si hay algún perfil problemático que sea el usuario quien puntualmente lo ponga en modo *complain*. Si debemos desinstalarlo, por cualquier otra causa, por ejemplo, para instalar SELinux, podemos:

```
sudo invoke-rc.d apparmor kill
```

```
sudo update-rc.d -f apparmor remove
```

En Ubuntu 9.10 y posteriores, podemos en cualquier caso:

- Ajustar el arranque del sistema en la línea de órdenes (ver */boot/grub/menul.lst* para Grub o */boot/grub/grub.cfg* para Grub 2) para incluir:
 - 'apparmor=0'
 - 'security=XXX' donde XXX puede ser "" para deshabilitar AppArmor o un LSM alternativo, por ejemplo, 'security="selinux"'
- Eliminar el paquete AppArmor con tu gestor de paquetes. No 'purgar' AppArmor si pensamos que podríamos utilizarlo en el futuro.

- **Habilitando el marco AppArmor**

```
sudo invoke-rc.d apparmor start
sudo update-rc.d apparmor start 37 S .
```

- **Recargando todos los perfiles**

```
sudo invoke-rc.d apparmor reload
```

- **Recargando un perfil**

```
sudo apparmor_parser -r /etc/apparmor.d/profile.name
```

Por ejemplo:

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

- **Deshabilitar un perfil**

```
sudo ln -s /etc/apparmor.d/profile.name /etc/apparmor.d/disable/
sudo apparmor_parser -R /etc/apparmor.d/profile.name
```

Ejemplo:

```
sudo ln -s /etc/apparmor.d/bin.ping /etc/apparmor.d/disable/
sudo apparmor_parser -R /etc/apparmor.d/bin.ping
```

- **Habilitar un perfil**

Por defecto, los perfiles estan habilitados, es decir, cargados en el kernel y aplicados a los procesos):

```
sudo rm /etc/apparmor.d/disable/profile.name
sudo apparmor_parser -r /etc/apparmor.d/profile.name
```

Ejemplo:

```
sudo rm /etc/apparmor.d/disable/bin.ping
sudo apparmor_parser -r /etc/apparmor.d/bin.ping
```

- Personalización de perfiles

Podemos encontrar los perfiles `/etc/apparmor.d`. Estos son archivos de texto que podemos editar con un editor de texto o con la herramienta `aa-logprof`.

Algunas personalizaciones pueden hacerse en `/etc/apparmor.d/tunables/`. Cuando actualizamos un perfil, es conveniente utilizarlas cuando sea adecuado. Por ejemplo, en lugar de usar una regla como:

```
/home/*/ r,
```

usar:

```
@{HOME}/ r,
```

Esto último expande la variable a un valor que puede cambiar sin cambiar el perfil completo.

Tras actualizar el perfil, debemos estar seguros de que lo re-cargamos.

- Perfiles

Antes de trabajar con perfiles, vamos a ver sus componentes. Para ello, utilizaremos un ejemplo, el perfil abreviado del programa `/usr/sbin/ntpd`, como muestra la Figura 2.

Las directivas `#include` permite utilizar perfiles establecidos de AppArmor para simplificar la construcción del nuestro y reducir así su tamaño. Por defecto, se toma como camino para los includes el `/etc/apparmor.d`.

AppArmor incluye dos clases de perfiles:

- **Abstracciones:** son perfiles que estan agrupados por tareas de aplicación comunes. Estas tareas incluyen mecanismos de autenticación, acceso a rutinas de servicio de nombres, requisitos gráficos comunes, y contabilidad del sistema. Los archivos listados en estas abstracciones son específicos de la tarea nombrada. Las abstracciones las encontramos en `/etc/apparmor.d/abstractions`.
- **Trozos de programas:** Contienen trozos de perfiles que son específicos para suites de programas y no suelen ser útiles fuera de esta suite, por lo que nunca se sugiere usarlos en el asistente de creación. Actualmente solo esta disponible la suite para el programa `postfix` en `/etc/apparmor.d/programa-chunks`.

En cuanto a los permisos de acceso del perfil, indicar que constan de una combinación de los siguientes seis modos:

- `r` – Permite la lectura del recurso
- `w` – Idem de escritura
- `px` – Modo de ejecución discreta del perfil (deniega el acceso si no hay perfil para el recurso).

- Px – Modo de ejecución limpia¹ (*clean exec*) (incompatible con Ux, ux, px o ix)
- ux – Modo de ejecución sin restricciones (ejecución del recurso sin perfil)
- Ux – Modo sin restricciones (*clean exec*).
- ix – Modo de ejecución heredado
- m – Permite `PROT_EXEC` con `mmap`
- l – Modo enlazado que media en acceso a enlaces duros. Cuando se crea un enlace, el archivo apuntado por él debe tener los mismos permisos de acceso que el enlace.

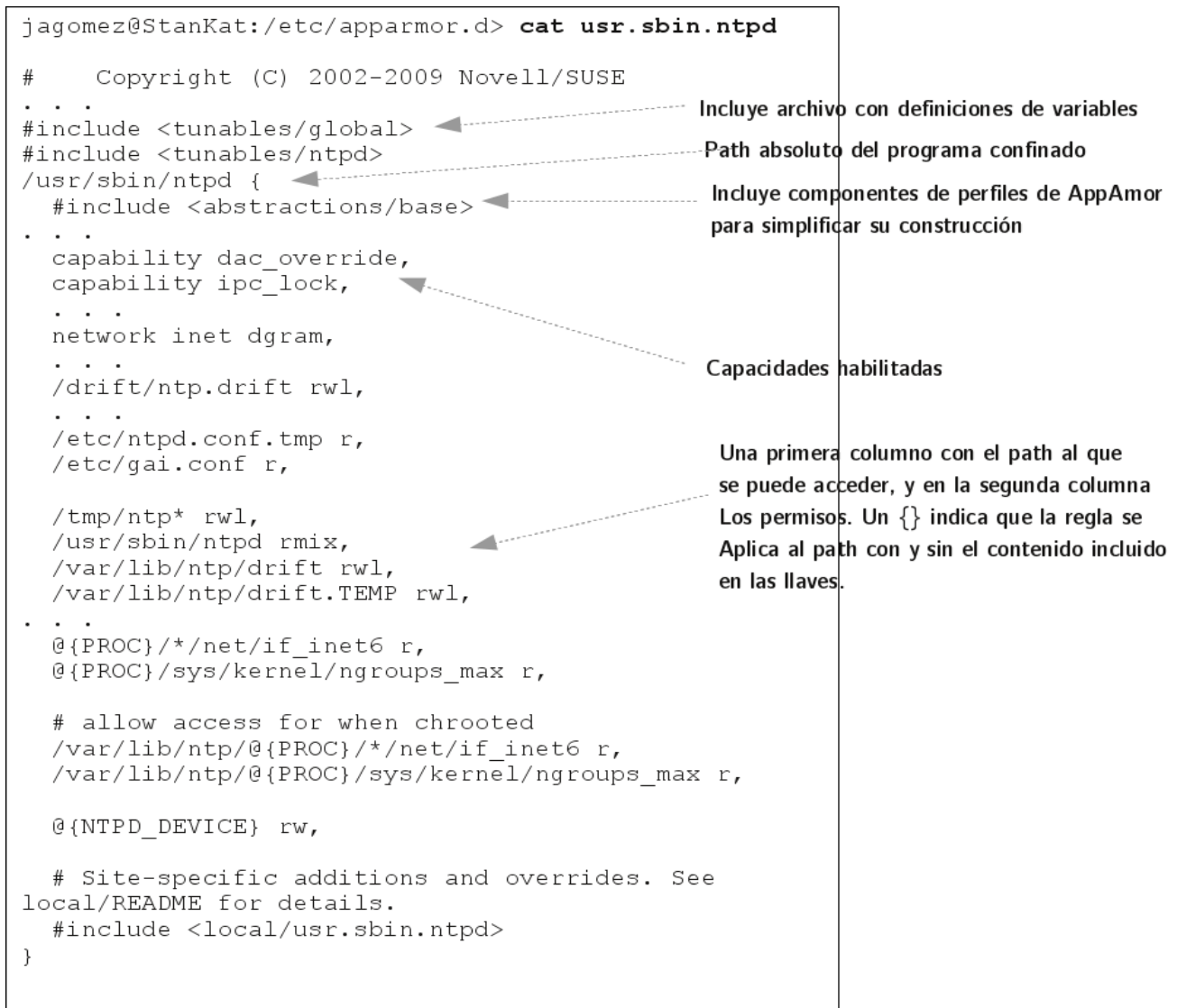


Figura 2.- Ejemplo de perfil AppArmor

- Creación de un perfil

Existen dos métodos o formas para crear un perfil junto con sus herramientas correspondientes.

A) *Perfil "stan-alone"* - aconsejable para pequeñas aplicaciones que tienen un tiempo de ejecución finito, tales como las aplicaciones de clientes. Se utiliza `aa-genprof` que se encarga de tomar todas los cuidados, pero que debemos ejecutar durante toda la duración del test de nuestro programa.

¹ https://www.novell.com/documentation/opensuse110/opensuse110_apparmor_admin_23/data/sec_apparm_profiles_exec.html

B) *Perfil Sistémico* – aconsejable para perfilar gran número de programas todos a la vez, o para aplicaciones que pueden ejecutarse durante días, semanas o continuamente entre rebotes, como aplicaciones de servidores de red, servidores web de correo, etc.

La automatización de perfiles es más manejable siguiendo los siguientes pasos:

- 1) Decidir el tipo de perfil a construir.
- 2) Realizar una análisis estático. Ejecutar bien `aa-genprof` o `aa-autodep`, dependiendo del método elegido.
- 3) Habilita el modo aprendizaje automático para todos los programas perfilados.

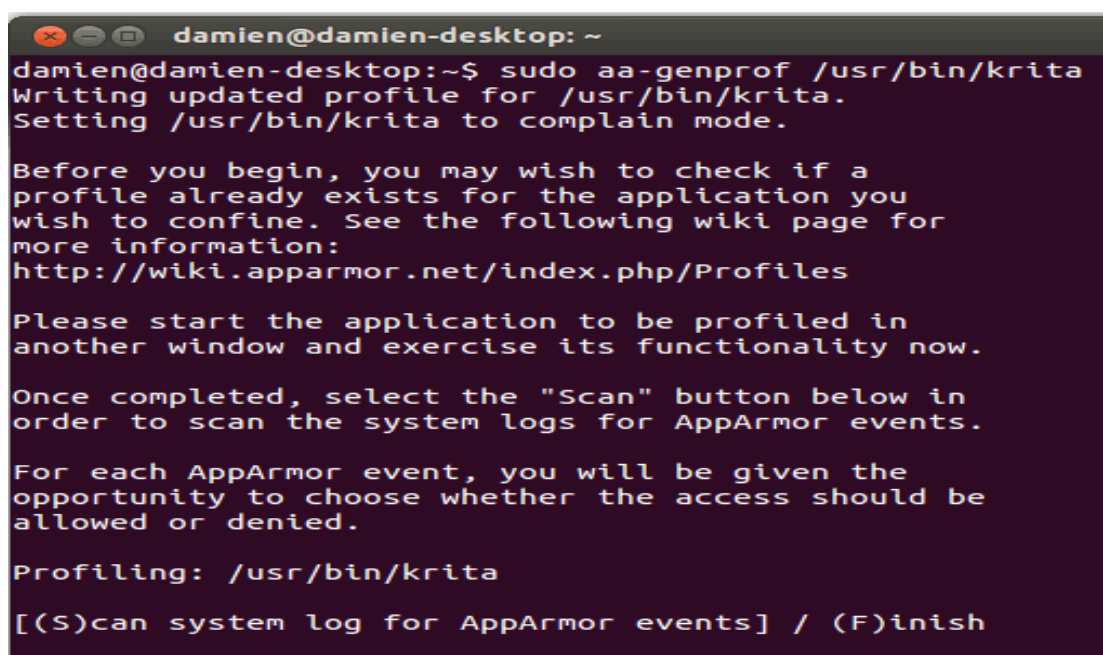
Vamos a ver el caso más simple de perfilado *stand-alone* para la aplicación `krita` (programa de edición). El caso de perfil sistémico se puede ver el Apartado 3.5.2 de la documentación de Novell sobre AppArmor (ver Bibliografía).

Si deseamos restringir una aplicación personal no protegida por Ubuntu, debemos crear nuestro propio perfil para confinarla. Esto es especialmente útil en servidores donde se ejecutan muchas aplicaciones en segundo plano y no las podemos ver. Antes de crear nuestro propio perfil será conveniente mirar en la biblioteca de perfiles para ver si hay uno para nuestra aplicación.

Como indicábamos antes, tras las instalación de *apparmor-utils*, debemos ejecutar:

```
% sudo aa-genprof path_to_application
```

donde `path_to_application` es el nombre de camino de la aplicación a la deseamos crear el perfil (Figura 3). El directorio por defecto es `/usr/bin`, pero puede ser diferente dependiendo de la aplicación.

A screenshot of a terminal window titled 'damien@damien-desktop: ~'. The user has entered the command 'sudo aa-genprof /usr/bin/krita'. The terminal output shows: 'Writing updated profile for /usr/bin/krita.', 'Setting /usr/bin/krita to complain mode.', followed by instructions to check for existing profiles, start the application, and scan system logs. It also shows the current profile is '/usr/bin/krita' and a prompt to scan system logs or finish.

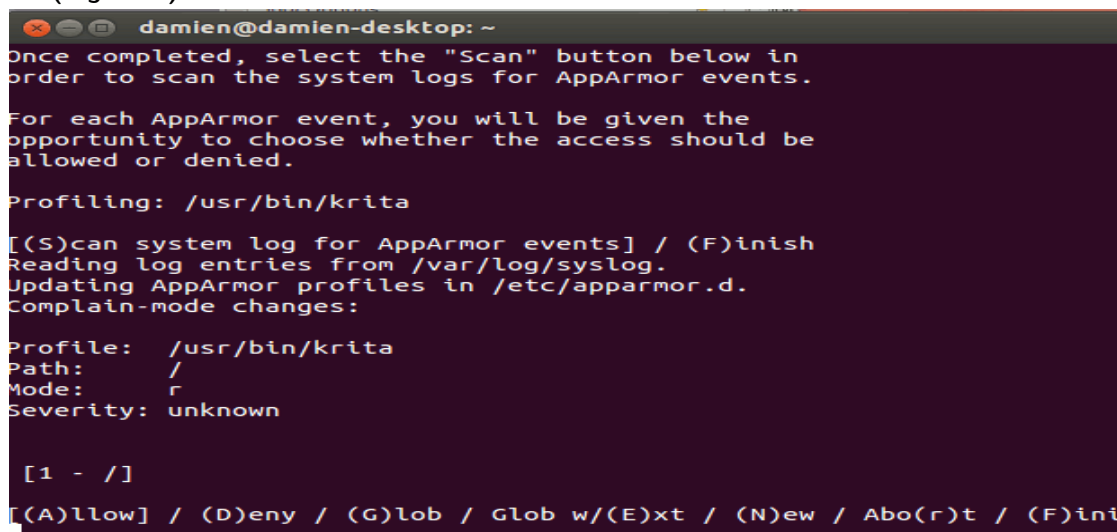
```
damien@damien-desktop: ~  
damien@damien-desktop:~$ sudo aa-genprof /usr/bin/krita  
Writing updated profile for /usr/bin/krita.  
Setting /usr/bin/krita to complain mode.  
  
Before you begin, you may wish to check if a  
profile already exists for the application you  
wish to confine. See the following wiki page for  
more information:  
http://wiki.apparmor.net/index.php/Profiles  
  
Please start the application to be profiled in  
another window and exercise its functionality now.  
  
Once completed, select the "Scan" button below in  
order to scan the system logs for AppArmor events.  
  
For each AppArmor event, you will be given the  
opportunity to choose whether the access should be  
allowed or denied.  
  
Profiling: /usr/bin/krita  
[(S)can system log for AppArmor events] / (F)inish
```

Figura 3.- Ejecución de *aa-genprof*.

A continuación arrancamos la aplicación a perfilar en el terminal y la utilizamos en la forma normal. La tarea más compleja es elegir entre los accesos de ejecución, es decir, establecer:

- inherit (ix)* El hijo hereda el perfil del padre, ejecutandose con los mismos controles de acceso. Utili cuando desde un programa queremos llamar a otro sin que gane privilegios.
- profile (px)* El hijo se ejecuta en su propio perfil. Si el perfil no esta cargado, fallará con permiso denegado. Esto es útil cuando el padre quiere acceder a servicios globales, como búsquedas DNS o enviar correo via MTA.
- Unconfined (ux)* El hijo se ejecuta de forma no confinada. Usar *Unconfined with clean exec (Ux)* para cancelar las variables de entorno o entorno que pudiera modificar la ejecución cuando se pase(n) al hijo. Esta opción introduce una vulnerabilidad que puede ser explotada, por lo que debemos usarla como último recurso.
- mmap (m)* Denota que el programa bajo el perfil puede acceder a los recursos usando mmap con el flag PORT_EXEC, es decir, lo datos mapeados pueden ser ejecutados.
- Deny* Evita que el programa acceda a las entradas de directorio especificadas.
- Abort* Aborta aa-logprof perdiendo las reglas anteriores.
- Finsih* Cierra aa-logprof salvando las reglas introducidas y modificando el perfil.

Para cada acción que ejecutemos en la aplicación, retornar al terminal con <Ctrl>+s para escanear los cambios (Figura 4).



```
damien@damien-desktop: ~
Once completed, select the "Scan" button below in
order to scan the system logs for AppArmor events.

For each AppArmor event, you will be given the
opportunity to choose whether the access should be
allowed or denied.

Profiling: /usr/bin/krita

[(S)can system log for AppArmor events] / (F)inish
Reading log entries from /var/log/syslog.
Updating AppArmor profiles in /etc/apparmor.d.
Complain-mode changes:

Profile: /usr/bin/krita
Path: /
Mode: r
Severity: unknown

[1 - /]

[(A)llow] / (D)eny / (G)lob / Glob w/(E)xt / (N)ew / Abo(r)t / (F)int
```

Figura 4.- Configuración del perfil.

Desde aquí podemos ver el path al que esta accediendo la aplicación y la severidad de sus acciones. Entonces, podemos permitir “Allow(A)” o denegar “Deny(D)” la acción. Hacemos esto para el resto de las acciones realizadas por la aplicación. Para obtener un resultado mejor, es mejor planear la lista de acciones antes de iniciar el perfilado. La opción “New” pregunta por nuestra propia regla para el evento y que puede ser cualquier expresión regular. La opción “Global” selecciona un parh específico o crea una regla general utilizando caracteres comodin que igualen un amplio abanico de paths. La opción “Global w/Exit” modifica el directorio actual mientras retiene las extensiones de los archivos.

Al final, presionamos <Ctrl>+F para acabar el perfil y <Ctrl>+s para salvarlo (Figura 5).


```
damien@damien-desktop: ~  
[2 - /usr/share/create/brushes/gimp/15fcirclef16.gbr]  
[(A)llow] / (D)eny / (G)lob / Glob w/(E)xt / (N)ew / Abo(r)t / (F)inish / (C)ancel  
Adding /usr/share/create/brushes/gimp/15fcirclef16.gbr r to profile.  
Profile: /usr/bin/krita  
Path: /usr/share/create/brushes/gimp/17circle.gbr  
Mode: r  
Severity: unknown  
  
1 - #include <abstractions/evince>  
[2 - /usr/share/create/brushes/gimp/17circle.gbr]  
[(A)llow] / (D)eny / (G)lob / Glob w/(E)xt / (N)ew / Abo(r)t / (F)inish / (C)ancel  
= Changed Local Profiles =  
The following local profiles were changed. Would you like to save them?  
[1 - /usr/bin/krita]  
(S)ave Changes / [(V)iew Changes] / Abo(r)t
```

Figura 5.- Guardamos los cambios del perfil.

Una vez creado, lo salvamos en `/etc/apparmor.d` y será cargado en modo *enforce*.

- Editar un perfil

Podemos editar un perfil con la orden `aa-logprof`:

```
% sudo aa-logprof path_to_application
```

Entonces, AppArmor escanea las entradas y nos permite hacer los cambios deseados.

3.- Inmunización de programa

AppArmor suministra tecnologías de inmunización que protegen a las aplicaciones Linux de las vulnerabilidades inherentes. Denominamos inmunizar a la protección de los programas suministrada por la política de AppArmor.

- Inmunizar programas que conceden privilegios

Los programas que necesitan un perfil son aquellos programas que median en los privilegios. Los siguientes programas tienen acceso a recursos que la persona que los utiliza no tiene, de forma que conceden privilegios al usuario que los usa:

- | | |
|------------------|--|
| Trabajos cron | – programas ejecutados periódicamente por cron y que se ejecutan con privilegios, normalmente de root. |
| Aplicaciones web | – programas ejecutados a través de un navegador web, que incluyen guiones CGI, páginas PHP, etc. |
| Agentes web | – programas (clientes o servidores) que tienen puertos abiertos, por ejemplo, clientes de correo o web. Estos programas se ejecutan con privilegios para escribir en el home del usuario y procesan peticiones de sitios potencialmente hostiles, como servidores web comprometidos o e-mail con código malicioso. |

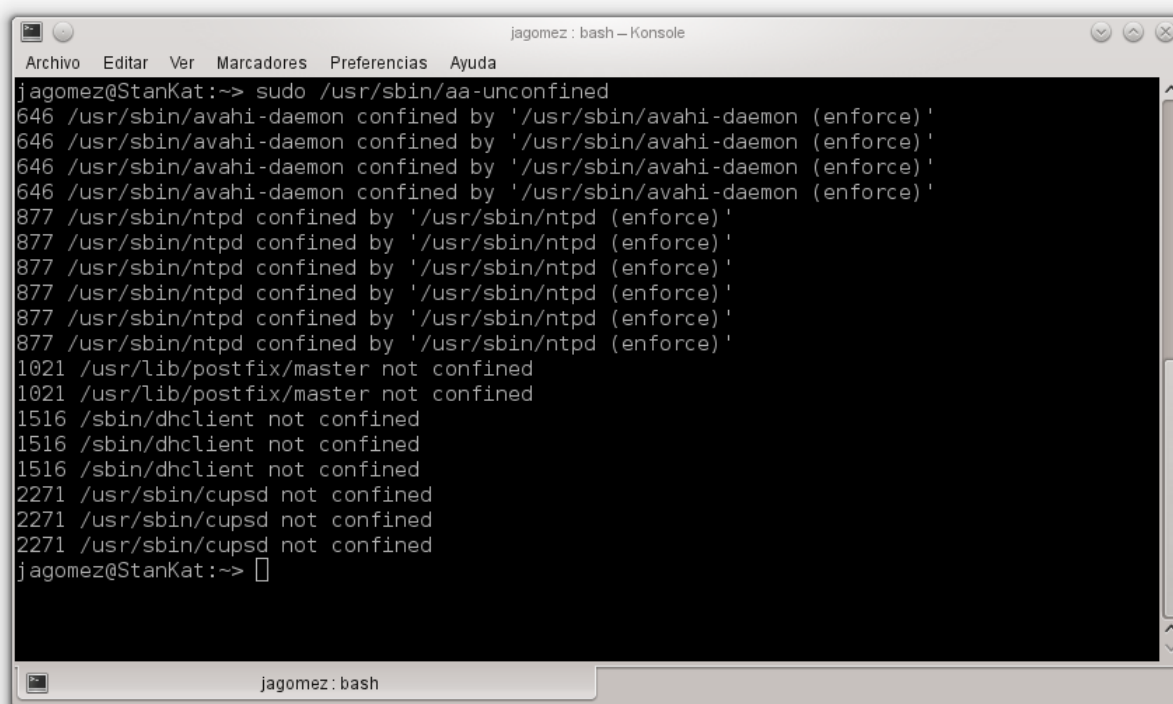
Por contra, programas sin privilegios no necesitan un perfil. Por ejemplo, un guión shell puede invocar a la orden `cp`, pero dado que `cp` no tiene su propio perfil, esta hereda del perfil del padre, de forma que `cp` puede copiar cualquier archivo que pueda leer o escribir el perfil de su padre.

- Inspeccionar puertos abiertos para inmunizar programas

Un procedimiento automatizado para encontrar demonios de red que deben ser perfilados es la herramienta `aa-unconfined`. Esta utiliza la orden `netstat -nlp` para inspeccionar los puertos abiertos en nuestro equipo, detectando los programas asociados a esos puertos e inspeccionando los perfiles AppArmor que tenemos cargados. Entonces, informa que perfil tiene asociado cada programa o “none” si el programa no esta confinado.

La primera columna (Figura 6) refleja el PID del proceso que escucha el puerto. La segunda, la cadena que representa el path absoluto del programa, y la tercera, el perfil que confina el programa o “no confined” si no está confinado.

Nota: `aa-unconfined` requiere privilegios de *root* y no debería ejecutarse desde un shell que este confinado por un perfil AppArmor.



```
jagomez@StanKat:~> sudo /usr/sbin/aa-unconfined
646 /usr/sbin/avahi-daemon confined by '/usr/sbin/avahi-daemon (enforce)'
```

PID	Path	Profile
646	/usr/sbin/avahi-daemon	confined by '/usr/sbin/avahi-daemon (enforce)'
646	/usr/sbin/avahi-daemon	confined by '/usr/sbin/avahi-daemon (enforce)'
646	/usr/sbin/avahi-daemon	confined by '/usr/sbin/avahi-daemon (enforce)'
646	/usr/sbin/avahi-daemon	confined by '/usr/sbin/avahi-daemon (enforce)'
877	/usr/sbin/ntpd	confined by '/usr/sbin/ntpd (enforce)'
877	/usr/sbin/ntpd	confined by '/usr/sbin/ntpd (enforce)'
877	/usr/sbin/ntpd	confined by '/usr/sbin/ntpd (enforce)'
877	/usr/sbin/ntpd	confined by '/usr/sbin/ntpd (enforce)'
877	/usr/sbin/ntpd	confined by '/usr/sbin/ntpd (enforce)'
877	/usr/sbin/ntpd	confined by '/usr/sbin/ntpd (enforce)'
1021	/usr/lib/postfix/master	not confined
1021	/usr/lib/postfix/master	not confined
1516	/sbin/dhclient	not confined
1516	/sbin/dhclient	not confined
1516	/sbin/dhclient	not confined
2271	/usr/sbin/cupsd	not confined
2271	/usr/sbin/cupsd	not confined
2271	/usr/sbin/cupsd	not confined

```
jagomez@StanKat:~> □
```

Figura 6.- Ejecución de `aa-unconfined`.

Algunas consideraciones sobre los resultados de la herramienta:

- No distingue entre interfaces de red, se informa de redes que pueden ser externas o internas.
- Detecta aplicaciones que se están ejecutando en el momento en que ejecutamos la herramienta,

por lo que el análisis puede no detectar clientes de red que no se estén ejecutando en este momento. Recordar que AppArmor esta más pensado para servidores que para estaciones de trabajo, por ello, se deja al usuario la confinación de clientes.

- Para confinar agresivamente aplicaciones en desktop , la orden tiene una opción paranoica (`--paranoid`), que informa de todos los procesos del sistema que se estan ejecutando y sus correspondientes perfiles que pueden asociarse o no a cada proceso. Es el usuario el que decide si asociarlos.

Ejercicio 4.2.- Selecciona un programa de tu distribución que no tenga perfil asociado y crea y activa un perfil con los privilegios que estimes oportunos. Indica cómo se ha reflejado en el perfil.

4.- Bibliografía

- Suse, "Novel AppArmor 2.0 Administration Guide", Suse, 2006. Disponible en http://www.novell.com/documentation/apparmor/book_apparmor21_admin/data/book_apparmor_admin.html.

- AppArmor, "Apparmor wiki", disponible (19/10/2017) en http://wiki.apparmor.net/index.php/Main_Page.

- Ubuntu, "Ubunti wiki", disponible (19/10/2017) en <https://wiki.ubuntu.com/AppArmor>.