Trabajo de Fin de Grado

Grado en Ingeniería de las Tecnologías de Telecomunicación

Fortificación del kernel para un sistema embebido

Autor: Diego Leñero Ramírez

Tutor: Germán Madinabeitia Luque

Dep. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo de Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Fortificación del kernel para un sistema embebido

Autor:
Diego Leñero Ramírez

Tutor:
Germán Madinabeitia Luque
Profesor Colaborador

Dep. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Trabajo de Fin de Grado: Fortificación del kernel para un sistema embebido

Autor: Diego Leñero Ramírez

Tutor: Germán Madinabeitia Luque

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A mi familia

A mis profesores

A ella

Agradecimientos

Con la realización de este trabajo se termina una etapa muy importante de mi vida en la que me he enfrentado a mi mayor reto personal hasta el momento. Es un buen momento para agradecer a todas las personas que me han ayudado en este camino plagado de dificultades. Empiezo por mi familia que desde siempre me han apoyado y han sufrido las dificultades tanto como yo. A los que están y a los que ya no están, que me han convertido en la persona que soy, sin los que no habría sido capaz de llegar a donde estoy.

También quiero agradecer a esa persona tan especial que me acompaña casi desde el principio de mi andadura en la Escuela Técnica Superior de Ingeniería y que ella más que nadie me ha visto progresar y me ha ayudado a superar todo lo que se ha puesto en mi camino.

Por último, agradecer a todos los profesores que he tenido a lo largo de toda mi vida. En Beas, en Huelva, en Trigueros y Sevilla. Algunos por su buena labor como docentes y otros por salirse de lo estrictamente docente y ofrecer en ocasiones una ayuda que hace que no vaya a poder olvidarme de ellos. Especialmente quiero mencionar a los profesores del Departamento de Ingeniería Telemática y a mi tutor de este trabajo, Germán Madinabeitia Luque.

Diego Leñero Ramírez

Grado en Ingeniería de las Tecnologías de Telecomunicación

Sevilla, 2016

Resumen

En este proyecto se ha trabajado en la fortificación de un kernel Linux. Se ha elegido un hardware Raspberry Pi porque se adapta al propósito de que el dispositivo pueda desplegarse en un entorno como una calle en una ciudad.

Se comienza describiendo las opciones de seguridad del kernel para luego centrarse en tres módulos principales tomoyo Linux, AppArmor y SELinux.

Con esto se ha logrado analizar las principales opciones de fortificación del kernel y describir los pasos para contruir con kernel con los módulos de seguridad descritos anteriormente.

Abstract

In this Project, we have worked on the fortification of a Linux kernel. We have chosen the Raspberry Pi hardware because we need a device that could be drop-down in an environment such a street in a city.

It begins by describing the kernel security options and then focus on three main modules (tomoyo Linux, apparmor and selinux).

With this, it has been posible to analyze the main fortification kernel options and describe the steps to build a kernel with the security modules described above.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 MOTIVACIÓN Y OBJETIVOS

El mundo le abre paso al hombre que sabe a dónde se dirige

- Ralph Waldo Emerson -

Existe hoy en día una tendencia a conectarlo todo a la red y exponerse a cantidades ingentes de información y contenidos. La aparición de dispositivos hardware con el tamaño de una tarjeta de crédito o menos con una alta capacidad computacional ha propiciado que en cualquier parte haya un servidor desplegado con cualquier propósito. Esto no solo afecta a las empresas sino también a usuarios. Estos, gracias a toda la información suministrada en la red, pueden programar estos dispositivos sin tener formación informática específica en servidores o seguridad.

Hay una gran variedad de dispositivos destinados a este propósito: estar conectados a la red y ofrecer un servicio. Que estén conectados no significa que ese servicio esté abierto al público, sino que también pueden tener motivaciones de vigilancia, captación de datos o cualquier aplicación que puedan llevar a cabo.

Este trabajo se centra en el uso de una Raspberry pi 2 B. Este dispositivo ofrece características similares a los que están siendo empleados en el despliegue de equipos en cualquier parte, concretamente en la calle o una carretera.

Aunque la tecnología tiende a conectarlo todo y hay una enorme variedad de aplicaciones y de manuales en internet para dispositivos Raspberry, la seguridad no es un punto muy tenido en cuenta por los usuarios. Por tanto, no existe tanta información sobre seguridad como sobre posibles aplicaciones.

La seguridad en un proyecto de despligue de un sistema embebido es un pilar fundamental. La funcionalidad debe estar muy bien especificada y el software debe hacer exactamente su propósito y no otras cosas; limitar esas “otras cosas” es una tarea complicada.

Un aspecto que debe tenerse en cuenta es que la seguridad que se aplica no debe reducir el rendimiento del sistema que se va a desplegar.

Para aplicar seguridad se tiene que dividir el sistema o el proyecto en capas, como una cebolla. Y aplicar en cada capa medidas de seguridad, aunque en este proyecto se dedica el esfuerzo a la seguridad del kernel es imprescindible analizar la seguridad en más capas.

En el ámbito de la seguridad de una Raspberry Pi desplegada en la calle hay múltiples amenazas que deben tenerse en cuenta: seguridad física, seguridad en el kernel, a nivel de red y a nivel de aplicación. Esto impacta en gran medida sobre qué tareas puede llevar a cabo. Para llevar a cabo una verdadera securización debe definirse muy precisamente qué se quiere hacer con el dispositivo y eliminar todas las demás funcionalidades innecesarias. Para así construir una fortificación en torno a lo que se quisiera proteger.

Los objetivos planteados en este trabajo son:

- Obtener información sobre los módulos del kernel que puedan servir para securizar el kernel para un dispositivo de las características de una Raspberry Pi.
- Aplicar esa securización a más capas como aplicación o red.
- Aportar ejemplos de configuraciones de seguridad.

2 HERRAMIENTAS

If it compiles, it is good, if it boots up it is perfect

- Linus Tolvalds -

Antes de empezar a desarrollar el núcleo del trabajo es conveniente comenzar con una descripción de los conceptos y herramientas. Esta presentación tiene el objetivo de simplificar la lectura e introducir al lector en la materia del trabajo sin profundizar demasiado en la aplicación en el proyecto. El manual de uso de las siguientes herramientas está en el anexo.

2.1 Raspberry Pi.

La Raspberry Pi 2 Model B es la segunda generación de Raspberry Pi. Existe una comparativa de tarjetas SD en la referencia [1] que pueden emplearse en estos dispositivos y su rendimiento. Esto es muy importante ya que puede cambiar significativamente el rendimiento general de la Raspberry. También cabe mencionar que las características técnicas y todo el potencial del hardware empleado pueden obtenerse de las referencias [2] y [3].

Las características más importantes para este proyecto son:

- Una CPU a 900 MHz quad-core ARM Cortex-A7
- 1 GB de RAM
- Puertos USB, Ethernet
- Un slot MicroSD
- Procesador de video e interfaces de cámara (CSI)

Esto se ajusta a la aplicación real para la que se realiza este trabajo, que se aplicará en dispositivos similares.

Por otro lado, permite ejecutar múltiples distribuciones de ARM GNU/Linux lo que la convierte en un buen banco de pruebas.

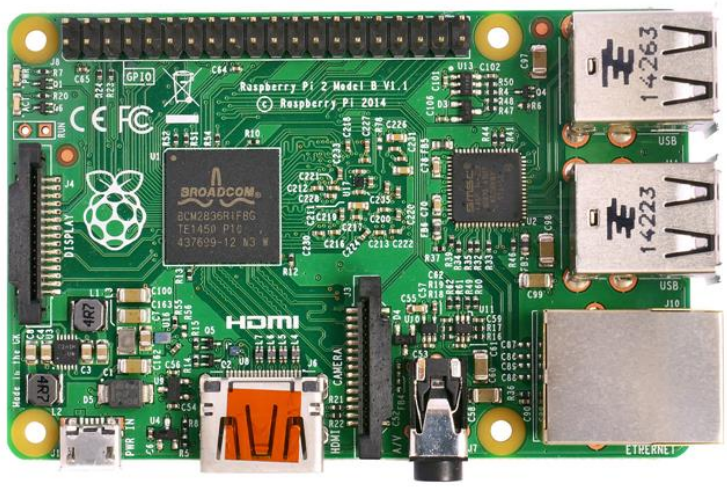

Figura 1 Raspberry Pi 2

2.2 Raspbian y kernel.

GNU/Linux es un sistema operativo que permite un alto grado de configuración y flexibilidad. Esta característica se empleará para compilar un kernel seguro. Tradicionalmente, las distribuciones implementadas en los servidores han sido Red Hat y Debian GNU/Linux. Ambos son un buen punto de partida para crear un sistema robusto.

La elección final es Raspbian, un sistema operativo basado en Debian optimizado para el hardware Raspberry Pi. Concretamente, se va empleando la versión lite porque no es necesario un entorno gráfico en el servidor ni tampoco gran cantidad de las herramientas y software que la versión completa trae. La imagen se obtiene de la referencia [4].

En cuanto al kernel, el trabajo se realizará sobre la última version disponible en el repositorio [5] lo que proporcionará un kernel 4.x. Además, para compilar se empleará el compilador y las herramientas proporcionadas en [6]. Dicho compilador es el gcc linaro arm Linux especializado para el dispositivo que se va a emplear.

Existe la posibilidad de realizar una compilación de un nuevo kernel en la Raspberry o hacer una compilación cruzada en otra máquina. Una compilación cruzada es la mejor opción en este proyecto puesto que puede realizarse en un tiempo mucho menor. Compilar en la Raspberry supone instalar todas las herramientas necesarias para la compilación y un tiempo de compilación que puede llevar muchas horas. El tiempo de compilación depende de las modificaciones realizadas. Por lo que es desaconsejable realizar una compilación en la Raspberry.

El kernel de Linux es robusto. Una de las razones es su estructura. Incluye un núclero monolítico y componentes modulares. La parte monolítica del kernel contiene aquellos controladores y opciones esenciales para el proceso de arranque del kernel. La parte modular del kernel añade todo aquello que es necesario para un correcto funcionamiento, incluyendo muchos controladores que habilitan la comunicación con el hardware instalado.

Por ejemplo, teniendo módulos para controlar una tarjeta de sonido o para añadir una funcionalidad extra que se cargarán si es necesario. Si ese módulo falla, eso no afecta a otras partes del sistema operativo Linux.

El kernel puede personalizarse o adaptarse a ciertos propósitos, como soporte de bases de datos, autenticación o cifrados. En caso de elegir modificar un kernel se debe tener claro qué se quiere hacer y qué se necesita para ello ya que cuando se abre la interfaz para añadir o quitar módulos se pueden encontrar más de 5000 opciones de configuración. Una forma recomendable de hacerlo es modificar un kernel ya compilado con la distribución y partir del fichero de configuración con el que se ha compilado ese kernel.

Si se necesita recompilar el kernel es necesario descargar el código fuente.

La siguiente tabla muestra las categorías en las que se organizan las opciones del kernel que, una vez compilado, involucarán a ciertos módulos del kernel de forma transparente al usuario. El kernel está en continuo desarrollo y mejora, por lo que las categorías pueden cambiar en el futuro. Además, algunas opciones pueden cambiarse en tiempo real, con las opciones de ayuda en el directorio `/sys/` y usando el commando sysctl y el fichero de configuración `/etc/sysctl.conf`.

Categorías básicas de configuración para el kernel de Linux	
CATEGORÍA	DESCRIPCIÓN
General setup	Incluye una variedad de opciones básicas.
Loadable module support	Establece condiciones sobre cómo se cargan los módulos en un kernel modular.
Block layer	Define cómo los dispositivos de bloques pueden cargarse.
Processor type and features	Configura el soporte para un conjunto de tipos de procesador y niveles de memoria.
Power management options	Define las características de gestión de energía.
Bus options	Especifica el soporte para tarjetas de hardware.
Executable file formats / emulations	Asociado con formatos binarios que pueden ejecutarse y enlazarse.
Networking	Incluye controladores de red, protocolos y otros relacionados.
Device drivers	Soporte para una amplísima variedad de hardware.
Firmware drivers	Soporte para sistemas no volátiles como el Basic Input/Output System (BIOS) y el Unified Extensible Firmware Interface (UEFI).
Filesystems	Incluye varios formatos de sistemas de ficheros.
Instrumentation support	Añade opciones sobre el comportamiento de los módulos.
Kernel hacking	Diseñado para la depuración del kernel.
Security options	Incluye opciones como Security Enhaced Linux (SELinux) y Application Armor (AppArmor). A esta categoría de módulos se dedica bastante trabajo en este proyecto.
Cryptographic API	Define los algoritmos de cifrado soportados por las application programming interfaces (APIs)

Virtualization	Añade código para máquinas virtuales basadas en hardware.
Library routines	Incluye módulos para verificación por redundancia cíclica.

Tabla 1 Categorías básicas de configuración para el kernel de Linux

2.3 El entorno de desarrollo del Proyecto.

En este punto es necesario empezar a mezclar la descripción de las herramientas y el uso de las mismas. No es necesario para el lector de este trabajo comenzar a familiarizarse con el entorno, por tanto, se hace referencia al Anexo A: Manual de configuración del entorno. Donde puede obtenerse una descripción detallada del proceso de configuración y compilación del kernel, así como de las herramientas empleadas para ese propósito.

3 SEGURIDAD EN EL KERNEL

The people who are crazy enough to think they can change the world are the ones who do.

- Steve Jobs -

Este capítulo se dedicará a la descripción de las opciones de seguridad del kernel sin profundizar demasiado en ninguna. Más adelante se procederá al desarrollo de documentación y pruebas de algunos de los mismos. La forma en la que se presentan es la que aparece en el menú de generación del fichero de configuración para la compilación del kernel. Dicho menú se despliega ejecutando el comando xconfig como viene descrito en el Anexo A: Manual de configuración del entorno.

3.1 Opciones de Seguridad.

3.1.1 Enable access key retention support

Habilitar el soporte de retención de clave de acceso.

Esta opción provee retención de tokens de autenticación y claves de acceso en el kernel. También proporciona métodos para que las claves se asocien, por ejemplo, a procesos o a cifrado. Además, también provee una clave especial que permite acceder al resto de claves. Cada proceso tendrá acceso a cinco llaveros estándar: específico de UID, específico de GID, sesión, proceso e hilo.

3.1.1.1 Enable register of persistent per-UID keyrings

Habilitar el registro de llaveros de claves persistentes por UID

Esta opción da soporte a un archivo de claves que permite a Kerberos usarlo como caché de credenciales. Es específico de Linux y lo usa para almacenar datos de credenciales en una memoria del kernel que no puede cambiarse ni modificarse, donde solo el usuario actual puede acceder. Además, esta memoria sobrevive cuando el usuario cierra sesión. Los archivos de claves se crean y añaden en el registro bajo demanda y se eliminan si expiran tras la cuenta atrás que se activa tras su creación.

3.1.1.2 Large payload keys

Claves grandes.

Mantiene claves grandes en el kernel, por ejemplo, cachés de tickets de Kerberos. Los datos pueden almacenarse en el área de intercambio por tmpfs.

3.1.1.3 Encrypted keys

Claves cifradas.

Esta opción permite crear, cifrar y descifrar claves en el kernel. Las claves cifradas son números aleatorios generados por el kernel que se cifran o descifran usando una clave maestra simétrica. La clave maestra puede

ser una clave de confianza o una clave de usuario. El espacio de usuario solo ve o almacena datos cifrados, no controla las claves.

3.1.2 Restrict unprivileged access to the kernel syslog

Restringir el acceso no privilegiado al syslog del kernel.

Esta opción refuerza las restricciones de acceso al syslog del kernel vía `dmesg` a los usuarios no privilegiados. Esta opción puede limitar la información proporcionada de dónde se ubican en memoria ciertas estructuras obtenidas a través del syslog del kernel que se utiliza maliciosamente cuando se realiza un ataque de escalado de privilegios.

3.1.3 Enable different security models

Habilitar diferentes modelos de seguridad.

Esta opción permite elegir entre diferentes módulos de seguridad para ser configurados en el kernel. Si no se selecciona se usará el modelo de seguridad por defecto en Linux.

3.1.4 Enable the securityfs filesystem

Habilitar el sistema de ficheros securityfs.

SecurityFS es un sistema de ficheros virtual cargado en memoria para los módulos de seguridad del kernel (LSM). Estos módulos sitúan sus políticas y otros datos aquí. El espacio de usuario ve securityfs como parte del sitema de ficheros montado en `/sys/kernel/security/`. Algunos de los módulos de seguridad leen y escriben ficheros de configuración en esta ubicación. Los LSM crean un directorio dentro del directorio raíz con su propio nombre. Por ejemplo, AppArmor o tomoyo Linux crearían un directorio llamado AppArmor o tomoyo, respectivamente, en `/sys/kernel/security`.

Sin embargo, hay sistemas que no lo usan como SELinux o SMACK.

3.1.5 Socket and Networking Security Hooks

Ganchos de seguridad de socket y red.

Si esta habilitado un módulo de seguridad podría usar este código de interceptación de llamadas a funciones o eventos para implementar controles de acceso. Esto último es necesario para SELinux.

3.1.5.1 XFRM (IPSec) Networking Security Hooks

Ganchos de seguridad para red XFRM (IPSec).

Permite implementar controles de acceso por paquete, basándose en las etiquetas definidas en la política de IPSec.

3.1.6 Security hooks for pathname based access control

Ganchos de seguridad para control de acceso basado en la ruta.

Si está habilitado, un módulo de seguridad puede implementar control de acceso basado en la ruta.

3.1.7 Low addess space for LSM to protect from user allocation

Espacio de bajo direccionamiento para módulos de seguridad de linux protegido del espacio de usuario.

Permite establecer un valor que representa la porción baja de la memoria virtual protegida del espacio de usuario. Evitando que un usuario escriba en las páginas bajas ayuda a reducir el impacto de los fallos de puntero nulo del kernel. Esta vulnerabilidad puede producir potencialmente denegación de servicio y escalado de privilegios. En un sistema con arquitectura arm no debe ser supeior a 32768. Necesitarán permisos especiales los programas que mapean en las páginas bajas.

3.1.8 NSA SELinux Support

Soporte para NSA SELinux.

La sección 3.4 de este trabajo está dedicada a este módulo.

3.1.8.1 NSA SELinux boot parameter

Parámetro de arranque de NSA SELinux.

Esta opción añade el parámetro de kernel '`selinux`', permitiendo desactivar SELinux al arrancar con `selinux=0`. Esto faculta que una imagen de kernel se distribuya con SELinux costruido en el núcleo, pero sin estar necesariamente habilitado.

3.1.8.2 NSA SELinux runtime disable

Deshabilitar NSA SELinux en tiempo de ejecución.

Permite deshabilitar SELinux en tiempo de ejecución hasta el siguiente arranque. Esta opción es similar a la anterior, pero adaptada a sistemas en los que los parámetros de arranque son difíciles de usar.

3.1.8.3 NSA SELinux Development Support

Soporte de desarrollo de NSA SELinux.

Este módulo es muy útil para experimentar con SELinux y desarrollar políticas. Con esta opción activa, el kernel arrancará en modo permisivo (registra todo, no deniega nada) a menos que se especifique '`enforcing=1`' en la línea de comandos.

3.1.8.4 NSA SELinux AVC Statistics

Estadísticas de caché de vectores de acceso de NSA SELinux.

Esta opción reúne estadísticas en `/selinux/avc/cache_stats`, que pueden monitorizarse mediante herramientas como avcstat.

3.1.8.5 NSA SELinux checkreqprot default value

Valor por defecto del parámetro checkreqprot de NSA SELinux.

Esta opción determina si SELinux comprueba la protección solicitada por la aplicación o por el kernel. En el caso de que esta opción esté puesta a 0, SELinux comprobará que la protección aplicada sea la del kernel. Poniéndola a 1 comprobará la protección que solicita la aplicación. El flag checkreqprot puede cambiarse en el parámetro de arranque '`checkreqprot=`'. También puede cambiarse en tiempo de ejecución en `/selinux/checkreqprot` si la política lo permite.

3.1.8.6 NSA SELinux maximum supported policy format version value

Valor máximo de formato de política de NSA SELinux.

Esta opción permite establecer un valor concreto para la última versión del formato de política soportado por SELinux. El espacio de usuario obtiene dicho valor en el fichero `/selinux/policyvers.` SELinux comprueba este valor en el momento de cargar las políticas. Una opción en algunos casos es ajustar a un valor antiguo para evitar incompatibilidades. `checkpolicy -V` muesta el valor correcto. También aparece en las políticas instaladas en `/etc/selinux/$SELINUXTYPE/policy`, donde `SELINUXTYPE` está definido en `/etc/selinux/config`.

3.1.9 Simplified Mandatory Access Control Kernel Support

Soporte de kernel para control de acceso obligatorio simplificado.

SMACK es un módulo de seguridad del kernel que protege los datos y la interacción de procesos de manipulación maliciosa usando un conjunto de reglas MAC (Mandatory Access Control). Consta de tres componentes:

- Un módulo de kernel implementado como un LSM. Funciona mejor con los sistemas de ficheros que permiten atributos extendidos.
- Un script de arranque que comprueba que los ficheros de dispositivo tienen los atributos correctos y carga la configuración.
- Un conjunto de parches del paquete GNU Core Utilities y Busybox para que tenga en cuenta los atributos extendidos.

La principal crítica a este módulo es que proporciona una funcionalidad similar a SELinux, sin embargo, no se encuentra definido como un módulo de este.

3.1.10 TOMOYO Linux Support

Soporte para tomoyo Linux.

Un apartado completo será dedicado a la descripción de este módulo del kernel en la sección 3.2.

3.1.10.1 (2048) Default maximal count for learning mode

Número máximo por defecto para el modo aprendizaje.

Es el valor por defecto para el máximo número de entradas de ACL que generadas automáticamente en la política en el modo de aprendizaje. Algunos programas ejecutan o emplean cientos de objetos por lo que ejecutar dichos programas en modo aprendizaje puede consumir mucha memoria. Este valor es un seguro para estos programas.

3.1.10.2 (1024) Default maximal count for audit log

Número máximo por defecto de registros de auditoría.

Es el valor por defecto para el máximo número de entradas de log de auditoría que el kernel puede manejar en memoria. Pudiéndose leer en `/sys/kernel/security/tomoyo/audit`. Si no se necesita auditar los logs se pone a cero este valor.

3.1.10.3 Activate without calling userspace policy loader

Activar sin llamar al cargador de políticas de espacio de usuario.

Esta opción permite activar el control de acceso desde el momento en que la política se carga. Esta opción es útil en sistemas donde es posible modificar la secuencia de arranque antes de cargar la política.

3.1.11 AppArmor Support

Soporte para AppArmor.

El apartado 3.3 está dedicado a este módulo.

3.1.12 Yama support

Soporte para yama.

Esta opción selecciona Yama, que extiende el DAC con opciones de seguridad a nivel de sistema. `Ptrace` es un buen solucionador de problemas para desarrolladores que sirve para determinar como funciona un proceso. Yama puede limitar su uso pues es posible utilizarlo con intenciones maliciosas.

Es posible determinar el valor activo mediante el siguiente comando:

```
$ sysctl kernel.yama.ptrace_scope
kernel.yama.ptrace_scope = 1
```

Hay cuatro opciones válidas:

- `kernel.yama.ptrace_scope = 0`: todos los procesos pueden depurarse siempre que tengan el mismo uid.
- `kernel.yama.ptrace_scope = 1`: solo un proceso padre puede depurarse.
- `kernel.yama.ptrace_scope = 2`: solo `root` puede usar ptrace.
- `kernel.yama.ptrace_scope = 3`: ningún proceso puede trazarse con ptrace.

3.1.12.1 Yama stacked with other LSMs

Yama apilado con otros módulos de seguridad de linux.

Cuando Yama está compilado en el kernel, lo fuerza a trabajar junto con el LSM seleccionado.

3.1.13 Integrity subsystem

Subsistema de integridad.

Habilita el subsistema de integridad que se compone de diferentes partes:

- "Integrity Measurement Architecture" IMA
- "Extended Verification Module" EVM
- IMA-appraisal extension
- digital signature verification extension
- audit measurement log support.

Cada uno de los componentes puede habilitarse o deshabilitarse independientemente.

3.1.13.1 Digital signature verification using multiple keyrings

Verificación de firma digital empleando múltiples juegos de llaves.

Esta opción permite el soporte de verificación de firma digital empleando multiples juegos de claves. Define diferentes claves para cada caso de uso (evm, ima…). Emplear varias claves mejora la búsqueda, pero también permite bloquear ciertos juegos de claves para prevenir que se añadan nuevas claves. Esto es útil para evm, donde normalmente las claves solo se añaden desde initramfs.

Cuando se marca esta opción se despliega una nueva opción 'Enable asymetric keys support' que permite la verificación de firmas digitales empleando claves asimétricas.

3.1.13.2 Enable integrity auditing support

Habilitar el soporte de auditoría de integridad.

Además de habilitar la auditoria de integridad, esta opción añade el parámetro del kernel "`integrity_audit`" que controla el nivel de los mensajes de auditoria de integridad.

- 0 – mensajes de auditoria de integridad básicos (por defecto).
- 1 – mensajes adicionales.

Los mensajes adicionales de información se habilitan especificando "`integrity_audit=1`" en la linea de commandos del kernel.

3.1.13.3 Integrity Measurement Architecture (IMA)

Arquitectura de medición de la integridad

IMA de TCG mantiene una lista de valores hash de los ejecutables y otros ficheros sensibles del sistema. Si un atacante modifica un fichero importante será detectado.

Si el sistema tiene un chip TPM, IMA también mantendrá un valor de integridad dentro del hardware de forma que añade otra capa más de seguridad sobre los ficheros críticos del Sistema.

3.1.13.4 Default template

Plantilla por defecto.

Selecciona la plantilla IMA por defecto. La lista original ‘ima’ contiene un hash de 20 bytes y una ruta limitada a 255 caracteres. La lista ‘ima-ng’ permite mayores hashes y mayores rutas.

3.1.13.5 Default integrity hash algorithm

Algoritmo de resumen de inegridad por defecto.

Selecciona el algoritmo de hash empleado por defecto para la lista de medidas, la evaluación de integridad y el registro de auditoría. Este algoritmo tiene la posibilidad de ser cambiado tras la compilación usando el comando del kernel `ima_hash= opción`.

Las opciones disponibles son:

- SHA1
- SHA256
- SHA 512
- WP512.

3.1.13.6 Appraise integrity measurements

Evaluación de las mediciones de integridad.

Esta opción permite la evaluación de la integridad local. Se require que el sistema se etiquete con un atributo extendido que contiene el hash del fichero. Para proteger la seguridad de los atributos extendidos de un ataque offline se habilita EVM.

3.1.14 EVM support

Soporte para EVM.

Protege los atributos extendidos ante ataques a su integridad.

3.1.14.1 FSUUID (version 2)

Identificador único universal del sistema de ficheros.

Incluye el UUID del sistema de ficheros para el cálculo del HMAC.

3.1.14.2 Additional SMACK xattrs

Xattrs (extended file attributes) adicionales para el control de acceso obligatorio simplificado.

Incluye xattrs SMACK adiciones para el cálculo del HMAC. Además de los xattrs originales:

- `security.selinux`
- `security.SMACK64`
- `security.capability`
- `security.ima`

Con esta opción el kernel incluye nuevos xattrs:

- security.SMACK64EXEC
- security.SMACK64TRANSMUTE
- security.SMACK64MMAP

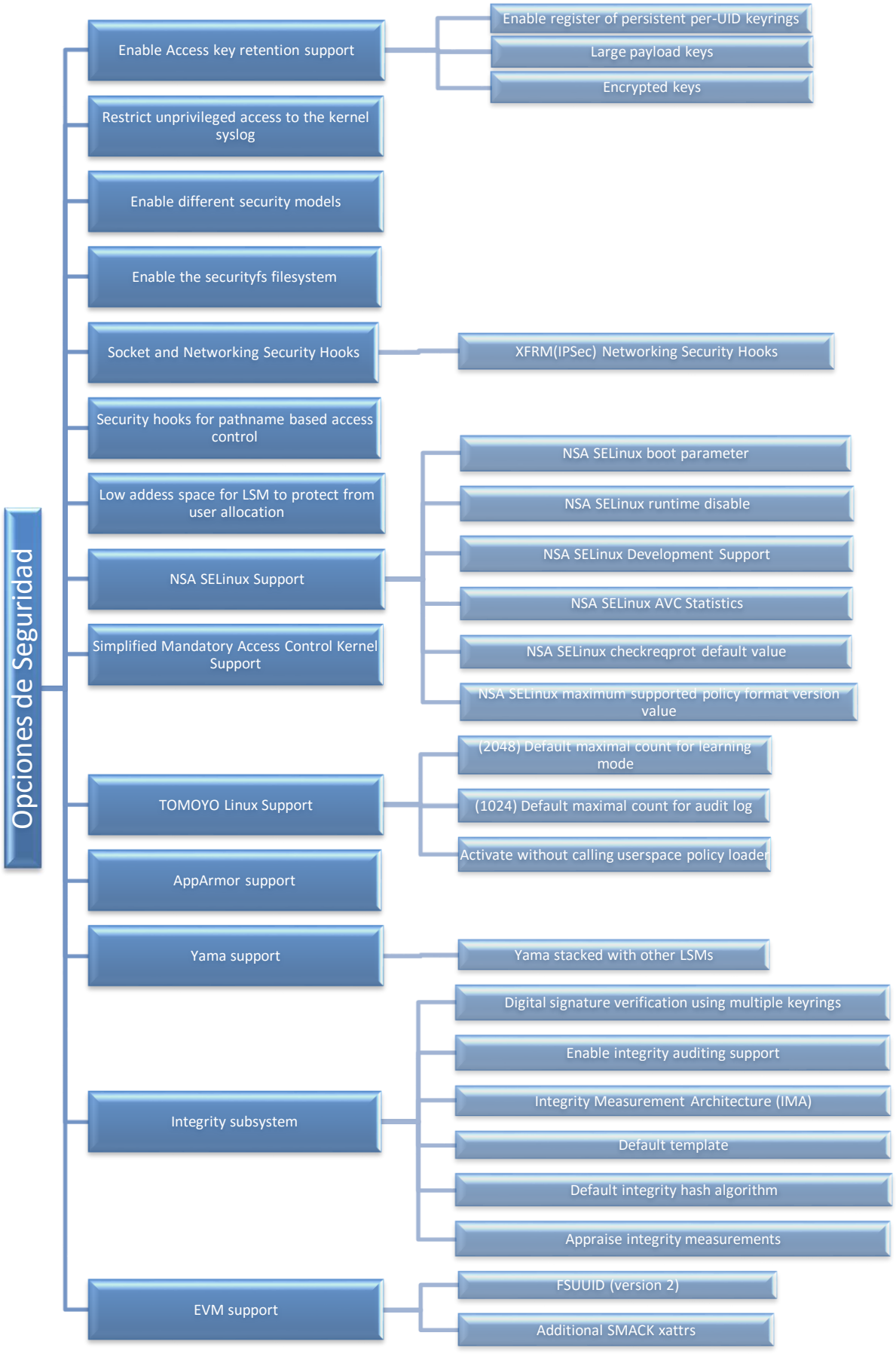

Figura 2 Árbol de opciones de Seguridad del kernel

3.2 TOMOYO Linux Support

Este apartado desarrolla la filosofía y la configuración de los paquetes y herramientas que interactúan con el módulo del kernel para tomoyo Linux.

Tomoyo es un sistema de control de acceso basado en la ruta, aunque también puede usarse simplemente como una herramienta de análisis. Tomoyo Linux está centrado en el comportamiento del sistema, todo proceso se crea con un objetivo y tomoyo permite a cada proceso declarar los comportamientos y recursos necesarios. Una vez habilitada la protección actúa como un guardián, restringiendo a cada proceso al comportamiento y recursos asignados por el administrador. Las principales características son:

- Análisis del sistema.
- Aumenta la seguridad del Mandatory Access Control (MAC).
- Contiene herramientas para ayudar a la generación de políticas.
- Sintaxis simple.
- Fácil de usar.
- Pocas dependencias.
- No requiere modificación de los ejecutables.

La sección 4.1 de este documento complementa esta descripción de tomoyo Linux. Dicha sección describe los módulos necesarios para su uso y se muestra una prueba de su funcionamiento.

Si el lector está interesado en tomoyo Linux puede consultar el artículo de la referencia [7].

En tomoyo Linux, la técnica empleada para reforzar el MAC es el uso de dominios. Este es un concepto importante. Cada proceso en un sistema pertenece a un dominio, que viene determinado por su historial de ejecución. Cada vez que un proceso se ejecuta, se crea un nuevo dominio. La concatenación de todas las rutas de ejecución previas de un ejecutable representa su dominio. Por ejemplo, se tiene un script llamado directamente por el kernel:

```
<kernel>/home/script
```

Si otro proceso que previamente ha ejecutado el kernel como por ejemplo `/sbin/init` llama a ese mismo script el nuevo dominio queda:

```
<kernel>/sbin/init  /home/script
```

De esta forma aparecen dos dominios diferentes para un mismo ejecutable, esto ocurre porque su historial de ejecución es diferente. Dicho de otro modo, en el primer caso el kernel es el que llama al ejecutable `/home/script` mientras que en el segundo caso el kernel llama a `/sbin/init` y es este último es el que llama a `/home/script`.

El kernel es siempre el primer dominio y tomoyo Linux lo representa con `<kernel>.`

El siguiente ejemplo ayudará a comprender el concepto de dominio. `/usr/sbin/apache2 es el` ejecutable de apache. Cada proceso que llame a dicho ejecutable tiene su dominio. Tomoyo crea un nuevo dominio cada vez que un proceso llama al ejecutable de apache si ese dominio no existe ya. Dicho nuevo dominio es el dominio del proceso que llamó al ejecutable seguido de un espacio y la ruta del ejecutable de apache. Por tanto, existen gran cantidad de dominios para ese ejecutable.

Los dominios permiten diferenciar diversos niveles de restricciones. Es decir, hay posibilidad de establecer una política para un ejecutable dependiendo de qué dominio lo llame. Parece una buena idea poder aplicar la

misma política de seguridad a apache independientemente de cómo se ejecute, es decir, independientemente de su dominio. También lo parece dar una política de seguridad diferente cuando apache se ejecute desde algún dominio concreto. Este tipo de configuraciones son parte del potencial de tomoyo Linux.

Un ejemplo real es la mejor forma de comprender qué es un dominio. `Domain transition editor` es una herramienta que proporciona tomoyo para analizar el sistema y configurar los dominios. La siguiente imagen se obtiene utilizando dicha herramienta. El dominio del ejecutable `/usr/sbin/apache2` en este ejemplo es

```
<kernel> /sbin/init /etc/init.d/apache2 /usr/bin/env /usr/sbin/apache2ctl
/usr/sbin/apache2
```

Eso significa que el kernel ha ejecutado `/sbin/init`, que dicho proceso ha ejecutado a su vez `/etc/init.d/apache2` y así sucesivamente. Esta es la forma en la que se generan dominios en el sistema.

```
136:  0 #       /sbin/init
137:  0             /bin/mount
138:  0             /bin/plymouth
139:  0             /bin/systemctl
140:  0             /bin/systemd-tmpfiles
141:  0             /bin/udevadm
142:  0             /etc/init.d/apache2
143:  0                 /bin/mktemp
144:  0                 /bin/plymouth
145:  0                 /bin/rm
146:  0                 /bin/run-parts
147:  0                 /bin/sleep
148:  0                 /usr/bin/env
149:  0                     /usr/sbin/apache2ctl
150:  0                         /bin/chmod
151:  0                         /bin/chown
152:  0                         /bin/mkdir
153:  0                         /bin/mktemp
154:  0                         /bin/mv
155:  0                         /bin/rm
156:  0                         /usr/bin/id
157:  0                         /usr/bin/stat
158:  0                         /usr/sbin/apache2
```

Figura 3 Ejemplo de dominio

Más adelante se verá cómo manejar los dominios.

Un perfil es el modo de un dominio y se aplica al conjunto de reglas o directivas que limitan el comportamiento del dominio. El sistema, mediante aprendizaje o manualmente, genera un conjunto de permisos y recursos a los que puede acceder un dominio. El perfil asignado a un dominio impide o permite violaciones de la política. También existe la posibilidad de asignar un perfil de aprendizaje para que el sistema genere las reglas que permitan que el sistema funcione correctamente. En la Figura 4 están descritas las opciones y características de los perfiles. Los perfiles se asigan a cada dominio de forma que dominio a dominio se va fortificando el sistema completo.

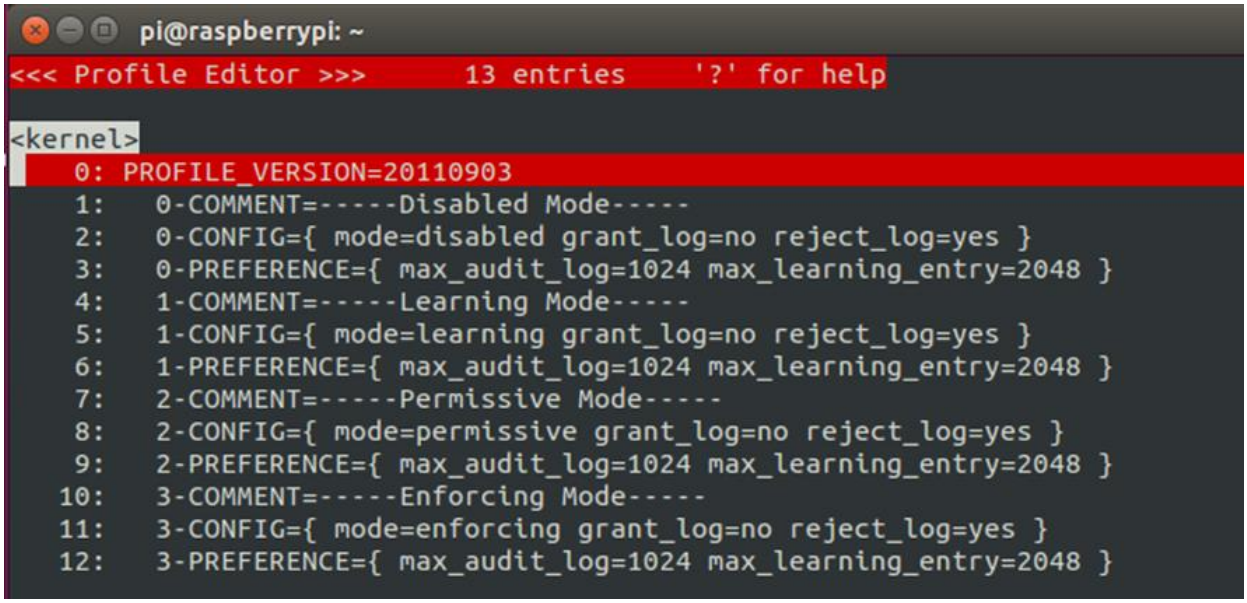

Figura 4 Editor de perfiles

Cada perfil consta de 3 campos:

- COMMENT: Descripción.
- CONFIG: Configuración del modo de operación.
- PREFERENCE: Configuración de otras opciones.

El parámetro "mode" de la linea CONFIG puede valer:

- `disabled`: actúa como un kernel normal.
- `learning`: No rechaza ningún acceso, aunque viole una política y añade ese acceso a la política.
- `permissive`: No rechaza ningún acceso, aunque viole una política.
- `enforcing`: Rechaza un acceso si viola una política.

Por tanto, estos son los cuatro perfiles existentes por defecto. Existe la posibilidad de asignarlos de forma independiente a cada dominio.

La linea PREFERENCE tiene dos parámetros:

- `max_audit_log`: Fija el número máximo de logs de auditoría que el kernel mantiene.
- `max_learning_entry`: Fija el número máximo de políticas añadidas en modo aprendizaje.

La política de seguridad del sistema es el conjunto de todos los perfiles asignados a cada dominio, reglas de transición de dominios y la configuración general de tomoyo. La política de dominio es el conjunto de reglas que afectan a un dominio concreto.

En este texto se va a hacer referencia a perfil de un dominio como el conjunto de las políticas que involucran a

un dominio y el modo en el que se encuentran esas políticas. Esto es así porque el perfil y la política de dominio tienen una relación muy estrecha y no es necesario separar una cosa de la otra.

La herramienta `Domain transition editor` muestra todos los dominios y el perfil que tienen asignado. Al seleccionar un dominio y pulsar Intro, tomoyo cambia a la pantalla `Domain policy editor` donde se muestra la política de dominio.

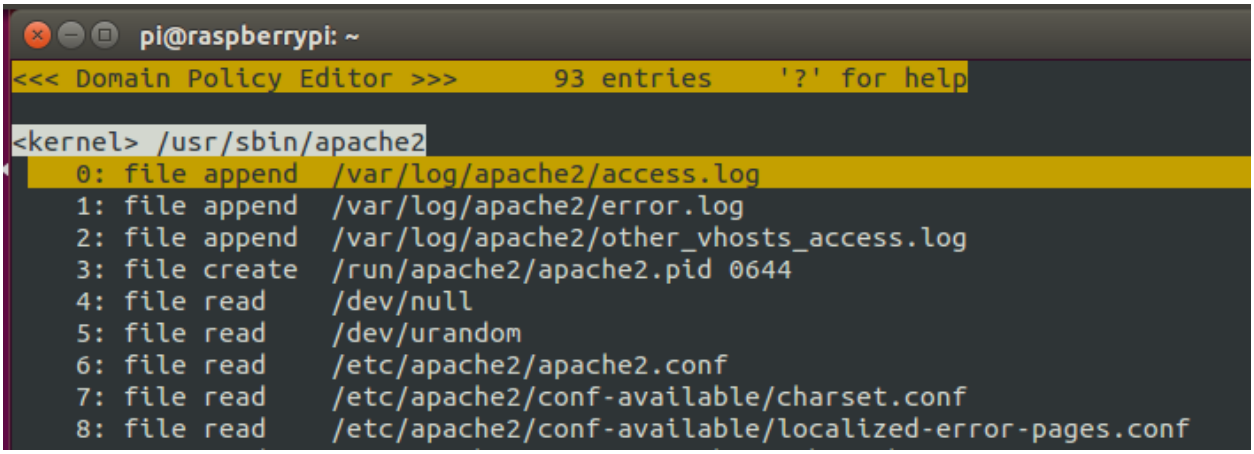

Figura 5 Domain policy editor

Quedan introducidos los conceptos de dominio, perfil y política de dominio. Es el momento de profundizar un poco más en ellos.

Primeros pasos.

Lo primero es tener una máquina configurada adecuadamente, es decir, con un kernel preparado para soportar tomoyo y con la instalación realizada correctamente. Para ello pueden seguirse los pasos descritos en los apartados 4.1.1 y 4.1.2. El funcionamiento aquí descrito es el resultado del hardware y el software mostrado en la sección 2. Una vez está el sistema preparado se puede proceder a arrancar las políticas de tomoyo mediante:

```
$ sudo /usr/lib/tomoyo/init_policy
```

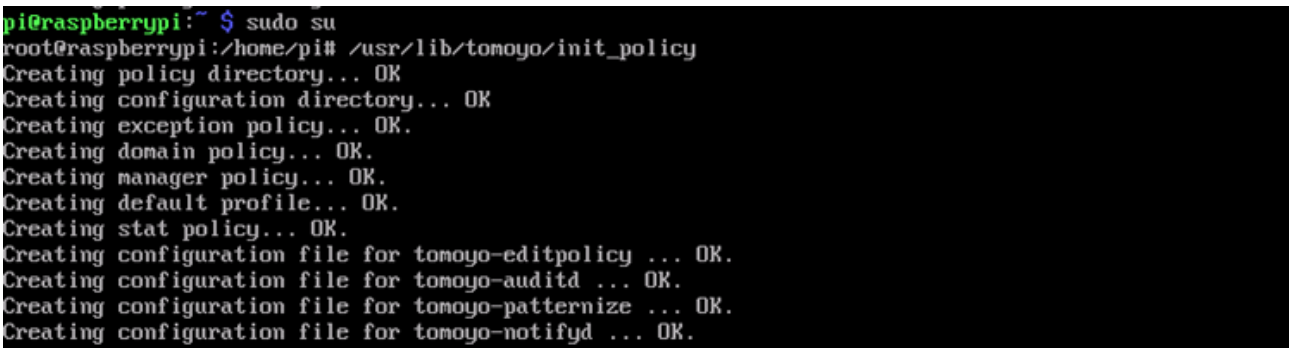

Figura 6 Ejemplo de ejecución de init_policy

A partir de este momento tomoyo almacena las políticas `en /etc/tomoyo/.`

Configuración de políticas, dominios y perfiles.

El editor de políticas se ejecuta mediante:

$ **sudo /usr/sbin/tomoyo-editpolicy /etc/tomoyo/**

Este editor provee algunas pantallas y cada una tiene un rol diferente. La pantalla por defecto es la de "`Domain Transition Editor`". En este editor al principio solo estará definido el dominio `<kernel>` y tomoyo detectará los dominios a medida que el sistema vaya ejecutando procesos. Tomoyo usa dos tipos diferentes de políticas, las leídas en el kernel y las guardadas en `/etc/tomoyo/`. Esto permite trabajar con las políticas mediante el editor de políticas y luego decidir si cargar en el kernel o desecharlas.

Ejecutando el siguiente comando aparecen los dominios creados en el kernel:

```
$ sudo  /usr/sbin/tomoyo-editpolicy
```

Pulsando '**w**' aparecen las distintas pantallas que proporciona el editor de políticas. La tecla '**p**' da acceso a la pantalla de edición de perfiles.

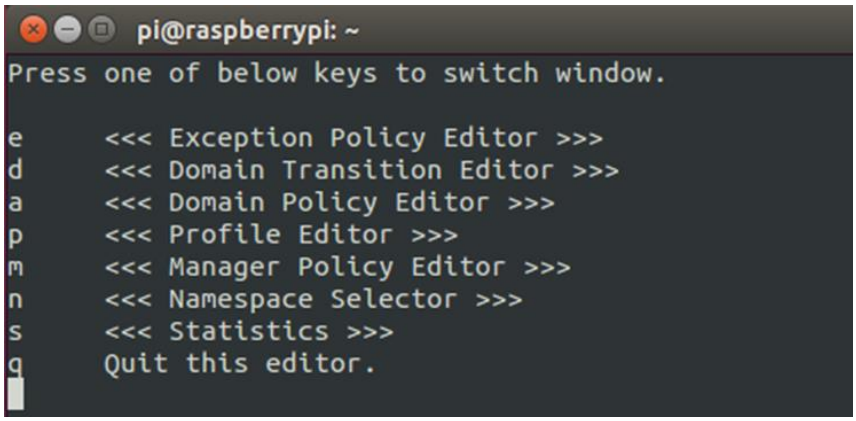

Figura 7 Editor de políticas de tomoyo

Cada dominio puede restringirse asignando un perfil. En la siguiente figura se observa cómo inicialmente en la pantalla mostrada por el comando anterior en la segunda columna todos los dominios tienen asignado un perfil 0. Esto significa que el dominio está deshabilitado, sin restricciones.

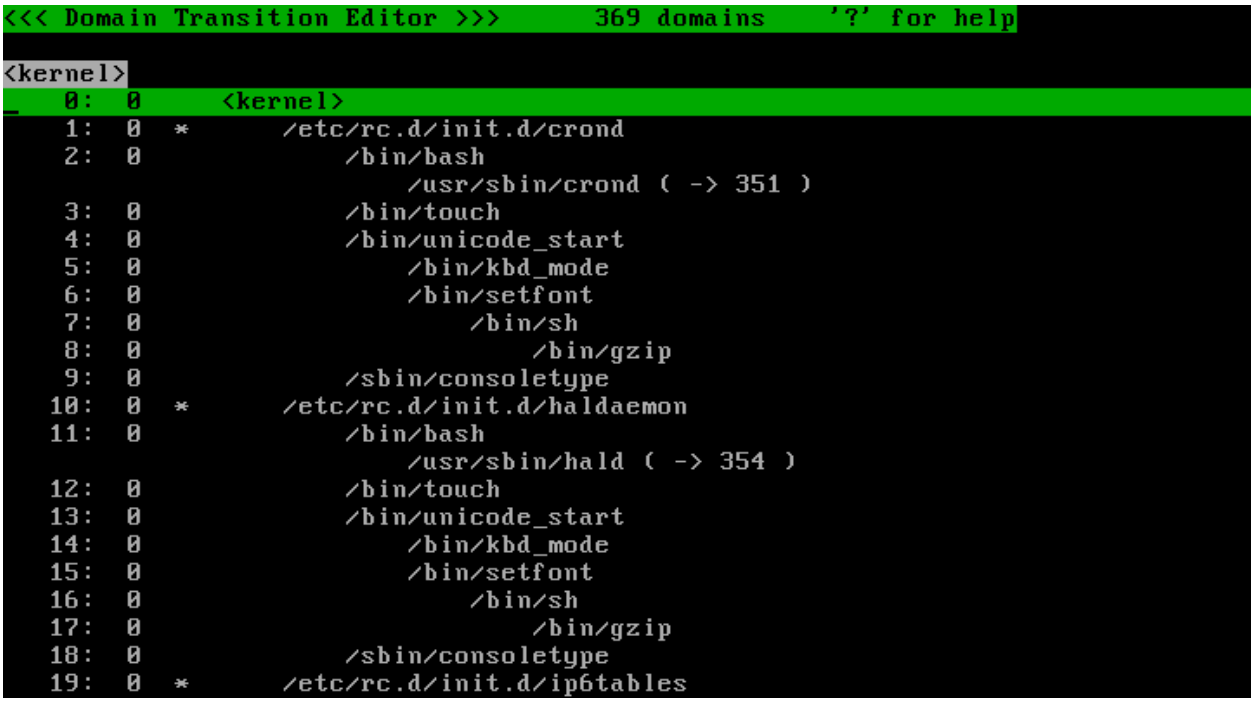

Figura 8 Dominios de tomoyo.

Por un lado, accediendo al editor de políticas de dominio se tienen las políticas para un servicio determinado. Por otro, también es posible usar el editor de políticas de excepción para habilitar algún acceso que se denegaría por la política de dominio.

Registrar logs de los accesos es muy importante para administrar el sistema; cuando las políticas están definidas y el sistema está en modo obligatorio puede permitir detectar un acceso que debe habilitarse en las políticas de excepción. Sin embargo, también puede ser útil mientras se emplea el modo aprendizaje para ayudar a desarrollar las políticas.

El fichero de configuración del demonio "`tomoyo-auditd`" que escribe los logs puede editarse en `/etc/tomoyo/tools/audit.conf`.

Para añadir una política de excepción, se ejecuta la pantalla de política de excepción. Una entrada se añade pulsando '**a**', introduciendo la nueva entrada y pulsando Intro.

Cuando una aplicación se ejecuta, tomoyo crea un nuevo dominio. Sin embargo, podría desearse dar los mismos permisos a una aplicación independientemente de cómo se haya ejecutado. Esto simplifica la gestión de políticas, para ello es necesario añadir una nueva política de excepción. Por ejemplo:

```
initialize_domain /usr/sbin/sshd from any
```

Tras añadir esta política, cada ejecución reinicializa el dominio `<kernel>/usr/sbin/sshd` independientemente de cómo se haya ejecutado la aplicación.

"`from any`" significa que tomoyo reinicializa el dominio de la aplicación cuando se invoque desde cualquier dominio. También puede hacerse la reinicialización desde un dominio específico con:

```
initialize_domain /usr/sbin/sshd from <kernel> /etc/rc.d/init.d/sshd
```

Esto sirve para poder limitar el comportamiento de esa aplicación cuando se ejecuta desde un dominio específico. Un ejemplo de uso es limitar el conjunto de acciones que puede llevar a cabo `/bin/bash` solo cuando se llama por el dominio de la aplicación ssh.

Si un dominio tiene un carácter ! en la tercera columna de la pantalla `"Domain Transition Editor"` significa que ese dominio es inalcanzable y puede eliminarse. El dominio que se ha inicializado aparece con un *.

También es posible diferenciar la ejecución de un proceso según un dominio concreto con:

```
initialize_domain /usr/sbin/sendmail.sendmail from any
no_initialize_domain /usr/sbin/sendmail.sendmail from /bin/mail
```

Lo que ocurre en este caso es que cualquier ejecución de sendmail ocurre bajo el mismo dominio, excepto si se llama desde `/bin/mail`.

Es preferible que algunas aplicaciones ejecutadas desde un dominio particular no experimenten una transición de dominio. Es decir, que actúen con los permisos del dominio que las invocó. Por ejemplo, aplicar la misma política a todas las aplicaciones ejecutadas en una sesión ssh.

```
keep_domain any from <kernel> /usr/sbin/sshd /bin/bash
```

De esta forma, la ejecución de `ls` o `cat` no resultará en la creación de un nuevo dominio. En su lugar quedarán en el mismo dominio especificado. Esto puede especificarse para cualquier shell mediante:

```
keep_domain any from /bin/bash
```

Esto podría precisarse aún más, causando la supresión de transición de dominio solo cuando se ejecute alguna aplicación en concreto:

```
keep_domain /usr/bin/xargs from <kernel> /usr/sbin/sshd /bin/bash
```

Lo que ocurre con la directiva que se acaba de presentar es que `/usr/bin/xargs` no genera un nuevo dominio cuando se ejecuta desde el dominio `<kernel> /usr/sbin/sshd /bin/bash`.

Puede haber casos donde se desee reanudar la transición de dominio. Por ejemplo, para permitir el acceso a ficheros de claves cuando se quiere cambiar la contraseña. Con las siguientes líneas tomoyo reanuda la transición de dominio solo para `cat`:

```
keep_domain any from <kernel> /usr/sbin/sshd /bin/bash
no_keep_domain /bin/cat from <kernel> /usr/sbin/sshd /bin/bash
```

Activando el modo aprendizaje.

El modo aprendizaje es una herramienta que permite la generación automática de permisos. Tomoyo es capaz de generar los permisos necesarios para una aplicación. Esto sucede cuando un proceso accede a recursos y lleva a cabo acciones que tomoyo registra. En este apartado se va a emplear un enfoque práctico para explicar este modo de funcionamiento.

En primer lugar, es necesario arrancar apache.

En `exception policy editor`, pulsando '**a**' para añadir una nueva línea de configuración e introduciendo la siguiente línea:

```
initialize_domain /usr/sbin/httpd from any
```

Ahora se pueden gestionar todas las instancias de apache desde el dominio `<kernel> /usr/sbin/httpd`

Pulsando '**s**' en `domain transition editor`, asignando perfil 1 y pulsando Intro.

Se pulsa '**@**' para cambiar a la lista de procesos. En esta nueva pantalla los procesos `/usr/sbin/httpd` tienen asignado el perfil 1. Así se comprueba que están en modo aprendizaje.

Guardando los permisos en el disco.

En caso de reiniciar el sistema los permisos se perderán. Para guardar las políticas en disco se emplea:

```
$ sudo /usr/sbin/tomoyo-savepolicy
```

Tras el último comando aparecen cuatro ficheros en `/etc/tomoyo/policy/YYYY-MM-DD.hh:mm:ss/` :

- `"exception_policy.conf"`
- `"domain_policy.conf"`
- `"profile.conf"`
- `"manager.conf"`

Para cargar la política actualmente en el disco en el kernel, se usa:

```
$ sudo /usr/sbin/tomoyo-loadpolicy -df < /etc/tomoyo/domain_policy.conf
$ sudo /usr/sbin/tomoyo-loadpolicy -ef < /etc/tomoyo/exception_policy.conf
$ sudo /usr/sbin/tomoyo-loadpolicy -p < /etc/tomoyo/profile.conf
$ sudo /usr/sbin/tomoyo-loadpolicy -m < /etc/tomoyo/manager.conf
```

"`-df`" significa sobreescribir "`/sys/kernel/security/tomoyo/domain_policy`".

"-ef" significa sobreescribir "/sys/kernel/security/tomoyo/exception_policy".

"-p" significa añadir a "/sys/kernel/security/tomoyo/profile".

"-m" significa añadir a "/sys/kernel/security/tomoyo/manager".

De esta forma configuración de seguridad permanece dentro del kernel.

Límites en el tamaño de las políticas.

El fichero de configuración "/etc/tomoyo/stat.conf" sirve para especificar límites en la memoria. Estos límites se establecieron en compilación, pero es posible modificados. El siguiente comando carga las modificaciones en el kernel:

```
$ sudo tomoyo-loadpolicy -s < /etc/tomoyo/stat.conf
```

Es importante tener esta opción en cuenta debido a que en modo aprendizaje todos los dominios del sistema a la vez pueden generar configuraciones muy grandes.

Desarrollar una política.

Modelos de archivos temporales

Un dominio en particular puede requerir diferentes permisos de acceso para diferentes recursos. Para ello los permisos pueden modelarse. Un permiso modelado permitirá que muchas entradas en la política de dominio se acorten en una sola.

Esto es especialmente importante con los ficheros temporales debido a la variedad de rutas únicas que involucran. Esta acción puede llevarse a cabo interactivamente en el editor de políticas de dominio por inspección y adición de reglas o usando tomoyo-findtemp. Que es lo que se va a hacer en este ejemplo.

```
$ sudo /usr/sbin/tomoyo-findtemp <
/sys/kernel/security/tomoyo/domain_policy
```

Es posible que al ejecutar el comando anterior aparezca /var/run/nscd/socket pero este fichero no puede considerarse temporal. Es decir, no hay que hacer nada con él.

En ocasiones las aplicaciones generan ficheros temporales con nombre que son combinaciones de carácteres aparentemente aleatorios, estos nombres de fichero se modelan usando "\?" que identifica cualquier carácter excepto "/". También está el carácter "*" que identifica cero o más repeticiones de cualquier carácter excepto "/". Hay un amplio rango de opciones en la referencia [8].

El fichero /etc/tomoyo/tools/patternize.conf explica cómo introducir modelos. Por ejemplo:

```
rewrite tail_pattern /tmp/Rs\?\?\?\?\?\?
rewrite tail_pattern /var/spool/clientmqueue/\*
```

Y luego comparar la política antigua y la nueva.

```
$ sudo tomoyo-savepolicy -d > /tmp/old
$ sudo tomoyo-patternize < /tmp/old > /tmp/new
$ sudo tomoyo-diffpolicy /tmp/old /tmp/new
```

Si la conversión da el resultado esperado, los cambios pueden aplicarse a la política actual mediante:

```
$ sudo tomoyo-diffpolicy /tmp/old /tmp/new | /usr/sbin/tomoyo-loadpolicy -d
```

Lo que consiguen estos patrones es permitir el acceso a archivos temporales o a ficheros y directorios cuyo nombre es aleatorio o puede variar.

Es posible aplicar estos patrones solo a dominios concretos especificando el dominio como argumento en el comando. En el siguiente ejemplo se aplica al dominio `<kernel> /usr/sbin/httpd`:

```
$ sudo tomoyo-selectpolicy -r '<kernel> /usr/sbin/httpd' <
/sys/kernel/security/tomoyo/domain_policy > /tmp/old-httpd

$ sudo tomoyo-patternize < /tmp/old-httpd > /tmp/new-httpd
$ sudo tomoyo-diffpolicy /tmp/old-httpd /tmp/new-httpd | tomoyo-loadpolicy
-d
```

Modelos de permisos de acceso.

Para ficheros que no necesariamente van a ser accedidos en el modo aprendizaje es posible crear patrones de permisos de acceso. Por ejemplo, para modelar contenido exportado por un servidor web:

```
<kernel> /usr/sbin/apache2

file read /var/www/html/\*.html
file read /var/www/html/\{\*\}/\*.html
file read /var/www/html/\{\*\}/\*.jpg
```

Con estos permisos de acceso el servidor web podrá acceder correctamente a los ficheros html y jpg en los directorios adecuados.

Las políticas de excepción simplifan la la política de dominio:

```
path_group WEB-CONTENTS /var/www/html/\*.html
path_group WEB-CONTENTS /var/www/html/\{\*\}/\*.html
path_group WEB-CONTENTS /var/www/html/\{\*\}/\*.jpg
```

Con esas directivas de excepción el grupo WEB-CONTENTS contiene todos los ficheros exportados por el servidor web. Introduciendo ahora en la política de dominio una sola línea se da acceso de lectura a todos los ficheros y directorios que necesita el servidor web:

```
<kernel> /usr/sbin/httpd

file read @WEB-CONTENTS
```

Hay más formas de simplificar la política. `tomoyo-patternize` es una buena herramienta para esta tarea:

- Primero hay que añadir los `path_group` a la política de excepción como antes.
- En segundo lugar, añadiendo a `patternize.conf` las siguientes lineas:

```
domain.equals <kernel> /usr/sbin/httpd
acl.starts file read
rewrite path_pattern /var/www/html/\*.html @WEB-CONTENTS

domain.equals <kernel> /usr/sbin/httpd
acl.starts file read
rewrite path_pattern /var/www/html/\{\*\}/\*.html @WEB-CONTENTS

domain.equals <kernel> /usr/sbin/httpd
acl.starts file read
rewrite path_pattern /var/www/html/\{\*\}/\*.jpg @WEB-CONTENTS
```

Obteniendo el mismo resultado que en el caso anterior.

Modelos de permisos numéricos.

Los patrones también simplifican números, como puertos o direcciones IP. En el siguiente ejemplo se observa una configuración generada en el modo aprendizaje:

```
<kernel> /usr/sbin/httpd

network inet stream accept 0:0:0:0:0:ffff:c0a8:801 3810
network inet stream accept 0:0:0:0:0:ffff:c0a8:801 3829
```

Para permitir el acceso a un rango de puertos, por ejemplo del 1024 al 65535, la siguiente política de dominio permite que el domino `<kernel> /usr/sbin/httpd` acepte conexiones en dichos puertos.

```
<kernel> /usr/sbin/httpd

network inet stream accept 0:0:0:0:0:ffff:c0a8:801 1024-65535
```

Como ya se ha visto anteriormente las políticas de excepción simplifican aún más la política de dominio:

```
number_group WEB-CLIENT-PORTS 1024-65535
```

Ahora la política de dominio también queda simplificada al usar el grupo @WEB-CLIENT-PORTS:

```
<kernel> /usr/sbin/httpd

network inet stream accept 0:0:0:0:0:ffff:c0a8:801 @WEB-CLIENT-PORTS
```

Modelos de permisos de acceso a red.

También existen patrones para direcciones IP en la política de dominio. Los siguientes permisos son ejemplos de acceso a red que van a ser simplificados:

```
<kernel> /usr/sbin/httpd

network inet stream accept 0:0:0:0:0:0:0:1 @WEB-CLIENT-PORTS
network inet stream accept 0:0:0:0:0:ffff:a00:1 @WEB-CLIENT-PORTS
network inet stream accept 0:0:0:0:0:ffff:a00:a1 @WEB-CLIENT-PORTS
network inet stream accept 10.0.0.10 @WEB-CLIENT-PORTS
network inet stream accept 10.0.0.200 @WEB-CLIENT-PORTS
```

Es posible simplificarlos empleando rangos de direcciones:

```
<kernel> /usr/sbin/httpd

network inet stream accept 0:0:0:0:0:0:0:1 @WEB-CLIENT-PORTS
network inet stream accept 0:0:0:0:0:ffff:a00:1-0:0:0:0:0:ffff:a00:ff @WEB-
CLIENT-PORTS
network inet stream accept 10.0.0.1-10.0.0.255 @WEB-CLIENT-PORTS
```

Otra opción es seguir los ejemplos anteriores y simplificar las directivas mediante la política de excepción:

```
address_group WEB-CLIENT-ADDRESS 0:0:0:0:0:0:0:1
address_group WEB-CLIENT-ADDRESS 0:0:0:0:0:ffff:a00:1-0:0:0:0:0:ffff:a00:ff
address_group WEB-CLIENT-ADDRESS 10.0.0.1-10.0.0.255
```

El último paso para simplificar a través de la política de excepción es definir en la política de dominio:

```
<kernel> /usr/sbin/httpd

network inet stream accept @WEB-CLIENT-ADDRESS @WEB-CLIENT-PORTS
```

De esta forma la política acepta las IP definidas en @WEB-CLIENT-ADDRESS en los puertos definidos en @WEB-CLIENT-PORTS.

Por último, también puede emplearse `patternize` añadiendo las políticas de excepción y añadiendo a `patternize.conf`:

```
  domain.equals <kernel> /usr/sbin/httpd
  acl.starts network inet stream accept
  rewrite address_pattern 0:0:0:0:0:0:0:1 @WEB-CLIENT-ADDRESS

  domain.equals <kernel> /usr/sbin/httpd
  acl.starts network inet stream accept
  rewrite  address_pattern  0:0:0:0:0:ffff:a00:1-0:0:0:0:0:ffff:a00:ff  @WEB-
CLIENT-ADDRESS

  domain.equals <kernel> /usr/sbin/httpd
  acl.starts network inet stream accept
  rewrite address_pattern 10.0.0.1-10.0.0.255 @WEB-CLIENT-ADDRESS
```

Revisión de los permisos necesarios.

Una vez parece que ha sido habilitado todo lo que se necesita que apache pueda hacer es el momento de cambiar el perfil del domino de apache a 2. Hay que tener en cuenta que Apache puede haber ejecutado otros programas como `/bin/sh` o `/usr/bin/perl` y, por tanto, tiene algunos dominios descendentes. Hay que asegurarse de cambiar el perfil de todos los dominios descendentes también.

Ahora se debe continuar usando apache y comprobar `/sys/kernel/security/tomoyo/stat` para ver cuántas violaciones de política han ocurrido:

```
$ sudo cat /sys/kernel/security/tomoyo/stat
```

Logs de auditoria para desarrollar la política.

Si los logs de auditoría están configurados, el siguiente comando muestra los permisos necesarios:

```
$ sudo cat /var/log/tomoyo/reject_002.log
```

Son complicados de entender, por este motivo, es necesario usar las herramientas que proporciona tomoyo para verlos de forma más clara:

```
$ sudo cat /var/log/tomoyo/reject_002.log | /usr/sbin/tomoyo-sortpolicy
```

Existe la posibilidad de guardar estos logs resumidos en un fichero temporal. Editarlos si es necesario y

añadirlos a la política actual:

```
$ sudo cat /var/log/tomoyo/reject_002.log | /usr/sbin/tomoyo-sortpolicy > ~/rejected.log

$ sudo nano ~/rejected.log
$ sudo /usr/sbin/tomoyo-loadpolicy -d < ~/rejected.log
```

Por último, es importante recordar que los cambios en la política actual se borran al reiniciar. Sabiendo esto, la política debe guardarse con el siguiente comando:

```
$ sudo /usr/sbin/tomoyo-savepolicy
```

Habilitar el enforcing mode.

El modo obligatorio se activa una vez la política está lo suficientemente pulida. Esto se consigue accediendo al editor de política de dominio y cambiando el perfil de los dominios que se han trabajado a 3. Pulsando '@' para ver la lista de procesos y verificando que, por ejemplo, "`/usr/sbin/httpd`" y sus procesos hijos tienen perfil 3.

Ahora es el momento de probar el funcionamiento del servidor e intentar forzarlo tratando de llevar a cabo alguna acción no permitida y se comprueban los logs:

```
$ sudo cat /sys/kernel/security/tomoyo/stat
```

Si se tienen los logs de auditoría también pueden revisarse para saber qué ejecuciones han sido rechazadas:

```
$ sudo cat /var/log/tomoyo/reject_003.log
```

Demonio de notificaciones.

`tomoyo-notifyd` es un demonio que se usa para reportar la violación de una política. Su configuración viene especificada en `/etc/tomoyo/tools/nofifyd.conf`. En dicho fichero hay ejemplos de acciones a tomar en caso de violaciones de política. Por ejemplo, puede enviar un correo electrónico.

Gestión de violaciones de política en tiempo real.

`tomoyo-queryd` es un software que permite la gestión de violaciones de política en tiempo real. Hay que tener en cuenta que no todas las violaciones de la política son maliciosas. Por ello es especialmente útil en la instalación de actualizaciones. Si algún paquete se altera, sería necesario modificar la política en alguno de los siguientes casos:

- Los nombres o rutas de ficheros han cambiado.
- Las dependencias de los ficheros han cambiado.
- Los permisos de acceso han cambiado.

La forma ideal de actuar es reconstruirlo todo desde el modo aprendizaje. Sin embargo, en un sistema en producción eso no es viable. En este punto entra en juego `tomoyo-queryd`, que permite modificar la política sin deshabilitar el modo obligatorio. No tiene por qué funcionar en todos los casos, es decir, que pueden darse situaciones que `tomoyo-queryd` no pueda solucionar.

Este es un ejemplo de cómo gestionar una actualización. En primer lugar, se ejecuta el siguiente comando para detectar violaciones en la política:

```
$ sudo /usr/sbin/tomoyo-queryd
```

Ahora actualizando con `apt-get update`, durante la actualización pueden ocurrir violaciones de políticas. El terminal que ejecuta `queryd` mostrará las violaciones que vayan sucediendo, dando la oportunidad de permitir la violación de la política. Por ejemplo:

```
 profile=3 mode=enforcing granted=no (global-pid=11788) task={ pid=11788
ppid=11779 uid=0 gid=0 euid=0 egid=0 suid=0 sgid=0 fsuid=0 fsgid=0 } path1={
uid=0 gid=0 ino=753729 major=253 minor=0 perm=0755 type=file } path1.parent={
uid=0 gid=0 ino=753665 perm=0755 } exec={ realpath="/bin/sleep" argc=2 envc=6
argv[]={       "sleep"       "1"       }       envp[]={       "TERM=xterm"
"PATH=/sbin:/usr/sbin:/bin:/usr/bin"  "PWD=/"  "LANG=en_US.UTF-8"  "SHLVL=1"
"_=/bin/sleep" } }

 <kernel> /etc/rc.d/init.d/sshd

 file execute /bin/sleep

 Allow? ('Y'es/'N'o/'R'etry/'S'how policy/'A'dd to policy and retry):
```

Esto indica que el proceso que pertenece al dominio `<kernel> /etc/rc.d/init.d/sshd` ha intentado ejecutar `/bin/sleep`. El software proporciona varias opciones:

‘**Y**’ para permitir el acceso.

‘**N**’ para rechazar el acceso.

‘**R**’ para reintentar el acceso (por ejemplo, tras editar la política manualmente).

‘**S**’ para mostrar la política de dominio del proceso.

‘**A**’ para añadir el acceso a la política de dominio y reintentar (primero permite editar el acceso antes de añadirlo).

Si parece razonable permitir esa ejecución hay que introducir '**A**' y tomoyo modificará la política para poder realizar la acción.

Durante la ejecución de este ejemplo en un servidor en producción pueden producirse ataques que violen la política, por tanto, no debe autorizarse cualquier violación de la política incondicionalmente.

Otra cosa a tener en cuenta es que si el administrador cierra sesión mientras este programa está corriendo y ese cierre de sesión viola alguna política quedaría pendiente y podría quedar durmiendo para siempre. Por ello el administrador debe cerrar primero el gestor de violaciones y luego cerrar sesión. Una vez que la política está correctamente actualizada, `queryd` se para con Ctrl-C.

Por último, como siempre, hay que guardar la política.

```
$ sudo /usr/sbin/tomoyo-savepolicy
```

Para profundizar un poco más en la gestión de violaciones se plantea una situación diferente. En este caso, una violación de política relacionada con acceso de lectura a un fichero:

```
#2016/02/27 00:22:26# profile=3 mode=enforcing granted=no (global-
pid=11937) task={ pid=11937 ppid=1 uid=26 gid=26 euid=26 egid=26 suid=26
sgid=26 fsuid=26 fsgid=26 } path1={ uid=0 gid=0 ino=690602 major=253 minor=0
perm=0644 type=file } path1.parent={ uid=0 gid=0 ino=690114 perm=0755 }

<kernel> /etc/rc.d/init.d/postgresql /sbin/runuser /bin/bash
/usr/bin/postmaster

file read /usr/share/zoneinfo/posix/Pacific/Pohnpei

Allow? ('Y'es/'N'o/'R'etry/'S'how policy/'A'dd to policy and retry):
```

Pulando '**S**' muestra la política. Comparando el mensaje y la política puede saberse si la violación tiene relación con la política que se está aplicando bien porque se haya producido un cambio de ubicación, porque se haya añadido un nuevo fichero a un directorio etc. Esto se realiza por inspección.

Si es seguro permitir la lectura del fichero, tomoyo añade las reglas necesarias pulsando 'A' . En este punto aparecerá:

```
Enter new entry> file read /usr/share/zoneinfo/posix/Pacific/Pohnpei
```

Pero puede modificarse para que los ficheros y directorios de esa ruta no den más problemas. Concretamente \{*\}/ significa todos los directorios recursivamente y * todos los posibles nombres de ficheros.

```
$ Enter new entry> file read /usr/share/zoneinfo/\{\*\}/\*
```

```
Added 'file read /usr/share/zoneinfo/\{\*\}/\*'
```

Habilitar el modo obligatorio para todos los dominios.

El sistema completo va siendo fortificado dominio a dominio siguiendo los pasos descritos anteriormente aplicados a cada dominio.

Aspectos más avanzados.

Registrar una aplicación.

Para que una aplicación o dominio pueda modificar la política en `/sys/kernel/security/tomoyo` debe estar registrada en `/sys/kernel/security/tomoyo/manager`. El comando para registrar una aplicación es el siguiente:

```
$ sudo echo "/usr/sbin/tomoyo-editpolicy" | /usr/sbin/tomoyo-loadpolicy -m
```

Gestión como usuario no-root.

Por defecto solo UID=0 y EUID=0 pueden modificar la política. La ejecución de los siguientes comandos habilita a un usuario para editar la política:

```
$ sudo echo "manage_by_non_root" | /usr/sbin/tomoyo-loadpolicy -m
$ sudo chown -R $USER /sys/kernel/security/tomoyo/
```

Donde $USER es el usuario habilitado.

Los siguientes comandos deshacen la configuración anterior:

```
$ sudo echo "delete manage_by_non_root" | /usr/sbin/tomoyo-loadpolicy -m
$ sudo chown -R root /sys/kernel/security/tomoyo/
```

El fichero `/etc/tomoyo/tomoyo-post-init` sirve para crear un usuario que pueda modificar la política. Para ello debe contener lo siguiente:

```
#! /bin/sh
echo manage_by_non_root > /sys/kernel/security/tomoyo/manager
chown -R tomoyo /sys/kernel/security/tomoyo/
```

`/sbin/tomoyo-init` ejecutará este fichero en el arranque. Es necesario que este fichero sea ejecutable y que el directorio de políticas tenga permisos de lectura y escritura por el usuario 'tomoyo'.

```
$ sudo chmod 755 /etc/tomoyo/tomoyo-post-init
$ sudo chown -R tomoyo /etc/tomoyo/
```

Para los dominios en modo aprendizaje no es necesario porque pueden añadir permisos hasta llegar al número máximo.

Conclusión.

Lo descrito hasta ahora es lo básico de tomoyo. Hay mucho más por explorar, pero no es necesario recogerlo todo en esta guía. Ahora, tras haber hecho esta introducción puede pasarse a securizar un sistema con tomoyo. El usuario que necesite resolver cuestiones más específicas puede consultar la documentación. El apartado 4.1 es un ejemplo de securización con tomoyo.

3.3 AppArmor Support

En este apartado desarrolla la filosofía y la configuración de los paquetes y herramientas que interactúan con el módulo del kernel para AppArmor.

Introducción a AppArmor.

AppArmor permite al administrador del sistema asociar a cada programa un perfil de seguridad que limite las capacidades. Provee control de acceso obligatorio basado en nombres. Limita programas individuales a un conjunto de archivos y de capacidades. La principal característica es que asocia los atributos de control de acceso a programas en lugar de a usuarios. Los perfiles AppArmor tienen dos modos de ejecución:

- Aprendizaje o complain: las violaciones de perfil se permiten y registran.
- Confinada o enforcement: fuerza la política de perfil y registra el intento de violación.

Hay dos tipos principales de reglas usadas en perfiles:

- Entradas de ubicación: detalla a qué ficheros puede acceder una aplicación.
- Entradas de capacidad: determinan qué privilegios puede utilizar un proceso confinado.

Se creó como alternativa a SELinux. Fácil de entender y usar para las tareas más comunes buscando alcanzar la misma funcionalidad. AppArmor es selectivo en el confinamineto de procesos en el sentido de que el administrador puede confinar unos programas y otros no.

Los programas no confinados emplean el sistema DAC (Discretionary Access Control). Los programas confinados primero evalúan el sistema DAC y si la acción está permitida, entonces evalúan el perfil

AppArmor.

AppArmor está en la serie de versiones 3.x, aunque aún está en desarrollo y se usa la serie 2.x. En este proyecto se emplea la versión 2.9 que es la que proporcionan los repositorios.

Por último, antes de finalizar esta introducción a AppArmor comentar que este capítulo está dedicado a profundizar en sus características mientras que la sección 4.2 de este documento muestra una prueba de su funcionamiento.

AppArmor proporciona el confinamiento a través de perfiles cargados en el kernel mediante `apparmor_parser`. Más información en [9].

AppArmor puede operar en dos modos: enforcement y complain:

- enforcement: Los perfiles cargados en este modo se ceñirán a la política definida en el perfil. La aplicación no podrá realizar ninguna acción no permitida y todos los intentos de violación de política se registrarán a través de syslogd.
- complain o aprendizaje: Los perfiles cargados en modo aprendizaje no reforzarán la política. En su lugar, solo reportarán los intentos de violación de política. Este modo es el utilizado para desarrollar perfiles.

  Para gestionar de este modo perfiles concretos se emplean `aa-complain` y `aa-enforce`. Estos programas toman el nombre de un programa como argumento. `aa-complain` sitúa un perfil en modo aprendizaje, mientras que `aa-enforce` sitúa un perfil en modo confinado. Para más información acerca del uso de estos comandos véase: [10] y [11].

AppArmor soporta el sistema de ficheros securityfs del kernel. Habitualmente, el script de arranque montará securityfs si no ha sido montado ya.

Por otra parte, AppArmor también restringe qué operaciones privilegiadas puede ejecutar un proceso confinado, incluso si el propietario del proceso es `root`. Un proceso confinado no puede hacer ninguna de las siguientes llamadas a sistema: `create_module, delete_module, init_module, ioperm, iopl, ptrace, reboot, sethostname, swapoff, swapon y sysctl`. A menos que se le de esa capacidad.

Perfiles.

Los perfiles son ficheros en texto plano como toda configuración en GNU/Linux. Están almacenados en `/etc/apparmor.d/` en ficheros cuyo nombre, por convención, reemplazan el caracter "/" en las rutas de los ejecutables por '.'. De esta forma los perfiles son más fáciles de gestionar. Por ejemplo `/etc/apparmor.d/bin.ping` es el perfil para el comando `/bin/ping`.

Para ponerlos en modo aprendizaje:

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

Para ponerlos en modo confinado:

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

Por otra parte, los perfiles se modifican sin que ello afecte a la configuración aplicada en el kernel.

`apparmor_parser` se utiliza para cargar un perfil en el kernel.

Para cargar un perfil que no ha sido cargado anteriormente se utiliza el siguiente comando:

```
$ cat /etc/apparmor.d/perfil.nombre | sudo apparmor_parser -a
```

El comando anterior puede usarse para recargar un perfil que esté cargado ya, usando la opción –r en lugar de la opción -a.

Por otra parte, para recargar todos los perfiles se puede emplear:

```
$ sudo /etc/init.d/apparmor reload
```

AppArmor aplica los perfiles a los procesos en el momento de su ejecución. Por tanto, un perfil cargado para un programa, no se aplicará hasta su próxima ejecución.

Empleando el sistema de ficheros securityfs, los siguientes comandos muestran la lista de perfiles actualmente cargados:

```
$ sudo mount -tsecurityfs securityfs /sys/kernel/security
$ cat /sys/kernel/security/apparmor/profiles
```

En caso de no emplear securityfs la configuración puede cambiar.

Algunos paquetes instalan sus propios perfiles y otros pueden ser encontrados en el paquete `apparmor-profiles`. El siguiente comando instala este y otros paquetes necesarios:

```
$ sudo apg-get install apparmor apparmor-profiles apparmor-utils auditd
```

Para comprobar que está funcionando tras la instalación:

```
$ cat /var/log/audit/audit.log | grep apparmor_parser
```

El anterior comando debería mostrar algo como:

```
type=AVC msg=audit(1316949034.097:108): apparmor="STATUS"
operation="profile_load" name="/bin/ping" pid=5207 comm="apparmor_parser"
```

Estos paquetes contienen utilidades de línea de comandos que se utilizan para cambiar el modo de ejecución de AppArmor, mostrar el estado de un perfil, crear nuevos perfiles, etc.

Para ver el estado de los perfiles:

```
$ sudo apparmor_status
```

Para listar los ejecutables que están actualmente confinados por un perfil AppArmor:

```
$ ps auxZ | grep -v '^unconfined'
```

Aquí nos limitaremos a hacer una introducción ya que los perfiles tienen su propio lenguaje y una sintaxis muy amplia que es imposible de describir en un trabajo como este. Por ello, la referencia [12] sirve para ahondar en ese tema.

AppArmor compila los perfiles para generar una política en binario que se carga en el kernel. Generalmente, una aplicación específica tiene un perfil diseñado para confinarla. Cada perfil contiene reglas para permitir el acceso a los recursos necesarios. El acceso se deniega cuando no hay una regla que lo permita en el perfil.

El formato de un perfil es el siguiente:

```
/usr/bin/Firefox flags=(complain) {
#Contenido del perfil
}
profile usuario {
#Contenido del perfil
}
```

El perfil comienza con el nombre del perfil, después vienen algunos flags opcionales, seguido del contenido del perfil entre corchetes. El perfil comienza con la palabra reservada profile si el nombre del perfil no comienza con '/'.

Generar un perfil.

Diseñar un plan de pruebas y pensar sobre cómo la aplicación debe funcionar es el primer paso para generar un perfil. El plan de pruebas consiste en casos de prueba pequeños. Cada caso de prueba debe tener una descripción pequeña y debe listar los pasos a seguir.

Algunos casos estándar de prueba son:

- Iniciar el programa.
- Parar el programa.
- Recargar el programa.
- Comprobar todas las órdenes soportadas por el script de inicio.

`aa-genprof` genera un nuevo perfil. Desde una terminal:

```
$ sudo aa-genprof ejecutable
```

Por ejemplo:

```
$ sudo aa-genprof slapd
```

Cuando el programa da error, AppArmor envía a los archivos de registro un mensaje de intervención. El programa `aa-logprof` explora archivos de registro para mensajes de intervención de AppArmor, revisarlos y actualizar los perfiles. Desde un terminal:

```
$ sudo aa-logprof
```

Ejemplo de perfil:

Fichero `/etc/apparmor.d/bin.ping`:

```
#include <tunables/global>
/bin/ping flags=(complain) {
  #include <abstractions/base>
  #include <abstractions/consoles>
  #include <abstractions/nameservice>

  capability net_raw,
  capability setuid,
  network inet raw,

  /bin/ping mixr,
  /etc/modules.conf r,
}
```

`#include <tunables/global>`: incluye declaraciones de otros archivos. Esto permite que las declaraciones pertenecientes a varias aplicaciones se puedan colocar en un archivo común.

`/bin/ping flags=(complain)`: ubicación del programa perfilado, también establece el modo del perfil a aprendizaje.

`#include <abstractions/base>`

`#include <abstractions/consoles>`

`#include <abstractions/nameservice>`: estos ficheros son librerías de permisos. Todas las librerías disponibles se encuentran en [13].

`capability net_raw`: permite a la aplicación acceder a la capacidad CAP_NET_RAW Posix.1e.

`capability setuid`: permite hacer manipulaciones en los UIDs de los procesos.

`network inet raw:` permite el uso general de red IPv4.

`/bin/ping mixr`: permite a la aplicación leer y ejecutar el archivo.

`/etc/modules.conf r`: permiso de lectura para este fichero.

Todas las capacidades pueden consultarse en la referencia [14] y [15].

Comandos AppArmor.

`aa-status`: obtiene el estado actual de AppArmor y los perfiles.

`apparmor_parser`: carga un perfil en el kernel.

`aa-genprof`: herramienta de generación de perfiles.

`aa-logprof`: repara políticas vía log.

`aa-complain`: pone un perfil en modo aprendizaje.

`aa-enforce`: pone un perfil en modo confinado.

`aa-disable`: deshabilita un perfil.

`aa-unconfined`: encuentra aplicaciones sin confinar.

Personalizar los logs.

Los registros por defecto están almacenados en `/var/log/messages` mediante `rsyslogd`. De esta forma los datos se limitan por el kernel para no sobrecargar el número de mensajes. Para poder usar los datos de auditoría con AppArmor y generar perfiles automáticamente, es recomendable instalar la herramienta `auditd` y editar `auditd.conf`:

```
$ sed -i -re 's/max_log_file = [0-9]+/max_log_file = 200/' /etc/audit/auditd.conf

$ /etc/init.d/auditd restart
```

Cuando un proceso confinado trate de acceder a un fichero para el que no tiene permisos de acceso, el kernel reportará un mensaje a través de audit similar a:

```
audit(1386511672.612:238): apparmor="DENIED" operation="exec"
                parent=7589 profile="/tmp/sh" name="/bin/uname" pid=7605
               comm="sh" requested_mask="x" denied_mask="x" fsuid=0 ouid=0
```

El permiso solicitado por el proceso aparece en el campo operation. En el log aparece el nombre (limitado a 15 octetos), el id de proceso y el nombre del perfil.

Para procesos confinados bajo un perfil en modo aprendizaje el acceso no se deniega y aparece el log cambiando DENIED por ALLOWED.

Es conveniente dejar aquí esta recopilación de documentación porque existe una wiki relativamente completa en la referencia [13]. Aclaro "relativamente", algunas de las páginas de esa wiki están en borrador o son poco prácticas.

3.4 NSA SELinux Support

Introducción

La National Security Agency (NSA) desarrolló originalmente SELinux, pero Red Hat ha completado una gran parte del trabajo de desarrollo. SELinux proporciona algunas defensas que son muy útiles a la hora de proteger servicios.

La primera defensa es implementar más de un bloqueo de los ficheros. SELinux implementa etiquetas en el sistema de ficheros para proveer de una mayor granularidad en los permisos de los ficheros. Estas etiquetas representan un contexto de seguridad que contiene cuatro campos. El primer campo contiene el usuario SELinux, el segundo contiene el rol, el tercero el tipo y el cuarto el nivel MCS. El nivel MCS (seguridad multicategoría: Multi-Category Security) es un parámetro que interviene en el establecimiento de una política de protección de la confidencialidad, que regula el acceso a archivos basándose en su sensibilidad. Por ejemplo, la ejecución del comando que se muestra tras este párrafo muestra el usuario, rol y tipo de cada uno de los ficheros y directorios. Para SELinux, todos los ficheros son considerados objetos, por ello estos directorios tienen una categorización de rol de objeto. El tipo está basado en el contenido de los directorios. El directorio `cgi-bin` contiene scripts ejecutables, por ello ese es el tipo asignado en la etiqueta. El directorio `html` incluye contenido asociado al servicio `apache`, por ello se le asigna ese tipo. Los ficheros en un directorio tendrán normalmente la misma etiqueta que el directorio, aunque eso no tiene porqué ser siempre el caso.

```
$ ls -lZ
drwxr-xr-x root system_u:object_r:httpd_sys_script_exec_t:s0 cgi-bin
drwxr-xr-x root system_u:object_r:httpd_sys_content_t:s0 html
```

SELinux tiene dos modos: permisivo y obligatorio. Cuando está activo el modo permisivo las acciones se registran, pero SELinux no hace nada por prevenir la acción o acceso. Cuando está en modo obligatorio, los módulos del kernel de SELinux evitan el acceso sin autorización. Todos los permisos se generan mediante una política que es configurada por el administrador. Los usuarios no pueden cambiar los permisos. SELinux almacena la política como un fichero binario, compilado de un conjunto de ficheros de configuración situados en `/etc/selinux`. SELinux puede proteger más cosas aparte de ficheros. Los puertos también tienen un contexto asociado. Si un proceso no tiene el contexto adecuado no podrá unirse a un puerto. El siguiente comando proporciona una lista de algunos contextos y puertos asociados.

```
$ sudo semanage port -l
```

La configuración básica se almacena en el directorio `/etc/sysconfig/` en el fichero `selinux`. Dicho fichero está enlazado a `/etc/selinux/config`. Las dos directivas en este fichero son:

- SELINUX – esta directiva tiene el valor `enforcing, permissive, o dissabled`. En modo permisivo, SELinux registra todas las violaciones de política en los ficheros de registro `messages` y `audit/audit.log` en el directorio `/var/log`.
- SELINUXTYPE – puede valer `targeted` o `strict`. Bajo la política `targeted`, todos los sujetos y objetos funcionan en el tipo unconfined_t excepto algunos demonios específicos. Los demonios en tipo `targeted` incluyen los siguientes servicios: DHCP, Apache, BIND, DNS, NCSD, NTP, Portmap, SNMP, Squid Web Proxy y el servicio de logging. Los objetos con tipo unconfined_t usan el sistema clásico de permisos DAC.

  La otra opción es la política estricta. En la política estricta cada sujeto y objeto existe en un dominio específico de seguridad y todas las interacciones posibles se definen en las reglas de la política.

Las opciones de configuración están detalladas en el directorio `/selinux/`. Cuando el administrados cambia la directiva SELINUX a disabled o desde disabled es necesario reiniciar para que SELinux etiquete adecuadamente todos los ficheros. Si SELinux está siendo habilitado, el reinicio puede tardar varios minutos más.

Los permisos de acceso de SELinux se representan por mapas de bits conocidos como vectores de acceso. Access vector cache (AVC) es el nombre de la forma en que SELinux almacena los permisos de acceso.

SELinux es en concepto como un cortafuegos. Todas las acciones las deniega al principio. Los contextos de SELinux sirven para que los usuarios, aplicaciones y servicios puedan trabajar normalmente. Estos contextos se asocian con los ficheros de configuración y librerías que necesitan los usuarios autorizados para hacer funcionar las aplicaciones y servicios.

SELinux puede emplearse para asignar acceso y derechos de transición a cada usuario, aplicación, proceso o fichero. Los derechos de transición es la capacidad de obtener los privilegios de otro contexto de seguridad en determinadas circunstancias. Las interacciones entre estas entidades se rigen por las políticas de SELinux. Por ejemplo, si el usuario del servicio `apache` de alguna manera consigue acceso `root,` ese usuario no podrá modificar ningún otro servicio.

Otros aspectos

SELinux distingue dos tipos de objetos en el sistema: sujetos (usuarios y procesos) y objetos (texto, binarios, sockets…). SELinux Permite un control preciso del acceso de los sujetos a los objetos a nivel bajo, impidiendo la interacción entre sujetos y objetos por defecto. Además, permite la creación de dominios. Un dominio puede ser un conjunto de aplicaciones como `httpd, htpasswd y htdigest` que tienen la posibilidad de ser tratadas conjuntamente como un grupo `http_r`.

Modos de funcionamiento

SELinux funciona en 3 modos:

`Enforcing o obligatorio` – SELinux aplica la política (`targeted`) y deniega el acceso si no cumple la política.

`permissive o permisivo` – SELinux aplica la política, pero solo se registran las violaciones.

`disabled o deshabilitado` – La política no se aplica.

Configuración básica de SELinux

La forma más simple de encontrar el estado actual de SELinux en el sistema local es con el comando `sestatus`. Este comando muestra la configuración en `/etc/sysconfig/selinux`.

Otra herramienta básica es `ls -Z` que muestra el estado de los ficheros en el directorio actual.

```
pi@raspberrypi:/var $ ls -Z
system_u:object_r:backup_store_t:SystemLow backups
         system_u:object_r:var_t:SystemLow cache
     system_u:object_r:var_lib_t:SystemLow lib
```

Figura 9 Ejemplo de ejecución de ls -Z

En este ejemplo, la salida muestra las características de los directorios en `/var/`. Los directorios están asociados con usuarios del sistema (`system_u`) y objetos del sistema (`object_r`) en un tipo de directorio asociado a un nivel de baja sensibilidad.

Para cambiar el contexto de SELinux por ejemplo, en caso de cambiar el usuario y tipo es útil el siguiente comando:

```
$ chcon -u system_u -t public_content /directorio
```

Tras su ejecución el sistema asignará el usuario y tipo especificado al directorio.

Configuración desde la linea de comandos

En los sistemas Debian, la mayoría de los comandos importantes de SELinux son parte de los paquetes `policycoreutils`, `setools` y `libselinux-utils`. Estos paquetes incuyen unos ficheros de configuración: `sestatus.conf` y `selinux/restorecond.conf` en el directorio `/etc/`. Estos ficheros incluyen listas de ficheros críticos y de servicios que vigilar. Los comandos de los anteriores paquetes se describen brevemente a continuación (aunque se usan en otros apartados):

- `audit2allow`

Revisa los logs y devuelve opciones que permitirían acciones que violan las políticas actuales de SELinux. La ejecución del siguiente comando muestra un ejemplo de cómo funciona:

```
$ audit2allow < /var/log/audit/audit.log
```

- `audit2why`

Traduce los mensajes de auditoría de SELinux en razones más fáciles de entender para una violación de SELinux. La ejecución del siguiente comando muestra un ejemplo de cómo funciona:

```
$ audit2why < /var/log/audit/audit.log
```

- `avcstat`

Muestra las estadísticas AVC (Access Vector Cache) relacionadas con la actividad de SELinux.

```
root@raspberrypi:/home/pi# avcstat
   lookups       hits     misses     allocs   reclaims      frees
    334702     329887       4815       5060       4256       4535
```

Figura 10 Ejemplo de uso de avcstat

- `chcat`

Especifica categorías para la seguridad de SELinux. Su fichero de configuración está en `/etc/selinux/targeted/setrans.conf`. Para ver las categorías actuales se ejecuta `chcat -L`.

```
root@raspberrypi:/home/pi# chcat -L
s0                             SystemLow
s0-s0:c0.c1023                 SystemLow-SystemHigh
s0:c0.c1023                    SystemHigh
```

Figura 11 Ejemplo de uso de chcat

- `fixfiles`

Re-etiqueta ficheros específicos. No es recomendable aplicar a un sistema completo salvo si se reinicia para evitar inestabilidad. Por ejemplo:

```
$ touch /.autorelabel
$ reboot
```

Durante el arranque, `init.rc` comprueba la existencia de `/.autorelabel`. Si este fichero existe, SELinux realiza un re-etiquetado de todo el sistema de ficheros (usando el comando `/sbin/fixfiles -f -F relabel`).

- `genhomedircon`

Genera contextos para los directorios de los usuarios. El comando `genhomedircon` usa la configuración de file_contexts para crear configuraciones `file_contexts.homedir` en el directorio `/etc/selinux/targeted/contexts/files`.

- `getenforce`

Devuelve el estado actual de SELinux: `enforcing, permissive o disabled`.

- `getsebool`

Devuelve el valor de un booleano de SELinux. `getsebool -a` devuelve la lista completa de booleanos. Los booleanos permiten cambiar partes de la política de SELinux en tiempo de ejecución, sin ningún conocimiento sobre la escritura de políticas de SELinux. Los valores se almacenan en el directorio `/selinux/boobleans/`.

- `load_policy`

Carga en el kernel la política actualmente configurada. Además, activa los cambios configurados.

- `matchpathcon`

Identifica el contexto de seguridad para un directorio especificado.

```
root@raspberrypi:/home/pi# matchpathcon /home/pi/
/home/pi        unconfined_u:object_r:user_home_dir_t:SystemLow
```

Figura 12 Contexto de seguridad del directorio

- `open_init_pty`

Abre y ejecuta un programa en un pseudo-terminal.

- `restorecon`

Restaura el contexto original de SELinux, con la opción –r funciona en modo recursivo. En el siguiente ejemplo se observa un fichero y su contexto. Se cambia su contexto con el comando `chcon` y seguidamente se restaura el contexto original con el comando `restorecon`.

```
root@raspberrypi:~# ls -Z
unconfined_u:object_r:user_home_t:SystemLow fichero.txt
root@raspberrypi:~#
root@raspberrypi:~# chcon --type=http_sys_content fichero.txt
root@raspberrypi:~#
root@raspberrypi:~# ls -Z
unconfined_u:object_r:http_sys_content:SystemLow fichero.txt
root@raspberrypi:~#
root@raspberrypi:~# restorecon fichero.txt
root@raspberrypi:~#
root@raspberrypi:~# ls -Z
unconfined_u:object_r:user_home_t:SystemLow fichero.txt
```

Figura 13 Ejemplo de uso de chcon y restorecon

- `restorecond`

Especifica el demonio asociado con el contexto original en un fichero. Cuando se ejecuta sin más parámetros se centra en los ficheros en `/etc/selinux/restorecond.conf`. En el siguiente ejemplo se ha cambiado el contexto de un fichero que se encuentra en `/etc/selinux/restorecond.conf` posteriormente se ha ejecutado `restorecond` y éste ha recuperado el contexto original del fichero.

```
root@raspberrypi:~# ls -Z
unconfined_u:object_r:user_home_t:SystemLow fichero.txt
root@raspberrypi:~#
root@raspberrypi:~# chcon --type=http_sys_content fichero.txt
root@raspberrypi:~#
root@raspberrypi:~# ls -Z
unconfined_u:object_r:http_sys_content:SystemLow fichero.txt
root@raspberrypi:~#
root@raspberrypi:~# restorecond
root@raspberrypi:~#
root@raspberrypi:~# ls -Z
unconfined_u:object_r:user_home_t:SystemLow fichero.txt
```

Figura 14 Ejemplo de uso de restorecond

- `run_init`

Ejecuta un script de inicio en el contexto SELinux. Se basa en el fichero `/etc/selinux/targeted/contexts/initrc_context`.

- `secon`

Revisa los contextos de SELinux de un fichero o programa. En el siguiente ejemplo se comprueba el contexto de un proceso.

```
root@raspberrypi:~# secon --pid 882
user: unconfined_u
role: unconfined_r
type: unconfined_t
sensitivity: SystemLow
clearance: SystemHigh
mls-range: SystemLow-SystemHigh
```

Figura 15 Ejemplo de uso de secon

- `sechecker`

Comprueba las políticas de SELinux de perfiles y módulos. Usando el parámetro -l devuelve la lista actual. El siguiente ejemplo muestra un fragmento de la salida al listar los perfiles y los módulos disponibles.

```
root@raspberrypi:~# sechecker -l

Profiles:
            analysis    common analysis checks
         development    common development checks
                 all    all available checks
          all-no-mls    all available checks not requiring MLS
Modules:
    attribs_wo_rules    attributes not used in any rule
    attribs_wo_types    attributes with no types
     domain_and_file    types treated as a domain and file type
    domains_wo_roles    domains with no roles
    find_assoc_types    utility module
        find_domains    utility module
```

Figura 16 Ejemplo de uso de sechecker

- `sediff`

Analiza la diferencia entre dos ficheros de texto de políticas de SELinux. Si se le pasa como primer y segundo parámetro dos ficheros de políticas diferentes muestra las diferencias entre ambos.

- `seinfo`

Devuelve estadísticas de las políticas de SELinux. Acepta como parámetro un fichero de políticas con extensión .conf o un fichero binario de políticas.

```
root@raspberrypi:~# seinfo /etc/selinux/default/policy/policy.29

Statistics for policy file: /etc/selinux/default/policy/policy.29
Policy Version & Type: v.29 (binary, mls)

   Classes:            83    Permissions:       251
   Sensitivities:       1    Categories:       1024
   Types:            3939    Attributes:        289
   Users:               6    Roles:              14
   Booleans:          228    Cond. Expr.:       256
   Allow:           90949    Neverallow:          0
   Auditallow:         25    Dontaudit:       15441
   Type_trans:      10528    Type_change:        72
   Type_member:        16    Role allow:         30
   Role_trans:        428    Range_trans:      2773
   Constraints:        83    Validatetrans:       0
   Initial SIDs:       27    Fs_use:             24
   Genfscon:           84    Portcon:           450
   Netifcon:            0    Nodecon:             0
   Permissives:         0    Polcap:              2
```

Figura 17 Ejemplo de uso de seinfo

- `sesearch`

Consulta políticas. Para más información consultar `sesearch --help`. El siguiente ejemplo muestra cómo encontrar para qué dominios existe una entrada del tipo `ssh_exec_t`.

```
root@raspberrypi:~# sesearch -t ssh_exec_t -c file -p entrypoint -Ad
Found 5 semantic av rules:
   allow nx_server_ssh_t ssh_exec_t : file { ioctl read getattr lock execute ent
rypoint open } ;
   allow virsh_ssh_t ssh_exec_t : file { ioctl read getattr lock execute entrypo
int open } ;
   allow condor_startd_ssh_t ssh_exec_t : file { ioctl read getattr lock execute
 entrypoint open } ;
   allow xm_ssh_t ssh_exec_t : file { ioctl read getattr lock execute entrypoint
 open } ;
   allow ssh_t ssh_exec_t : file { ioctl read getattr lock execute entrypoint op
en } ;
```

Figura 18 Ejemplo de uso de sesearch

- `semanage`

Gestiona políticas de SELinux para login, usuarios, puertos, interfaces y contextos. El siguiente ejemplo muestra los tipos y protocolos asignados a cada puerto.

```
root@raspberrypi:~# semanage port -l
SELinux Port Type              Proto    Port Number

afs3_callback_port_t           tcp      7001
afs3_callback_port_t           udp      7001
afs_bos_port_t                 udp      7007
afs_fs_port_t                  tcp      2040
afs_fs_port_t                  udp      7000, 7005
```

Figura 19 Ejemplo de uso de semanage

- `semodule`

Carga y gestiona módulos de políticas. SELinux almacena las listas de módulos en el directorio `/etc/selinux/targeted/modules/previous/modules`. Se tiene un ejemplo de uso en el apartado 4.3.2.

- `semodule_deps`

Muestra las dependencias necesarias para los diferentes paquetes de módulos.

- `semodule_expand`

Se usa internamente por la librería de gestión de SELinux para expandir un paquete de módulos a un fichero binario de políticas.

- `semodule_link`

Enlaza módulos, se emplea principalmente por desarrolladores.

- `semodule_package`

Crea un paquete de políticas SELinux.

- `sestatus`

Devuelve el estado actual de SELinux.

- `setenforce`

Modifica el estado actual de SELinux a `enforcing o permissive`.

- `setsebool`

Da un valor a un booleano de las opciones de SELinux. Se tiene un ejemplo de uso en el apartado 4.3.2.

- `togglesebool`

Cambia el valor de un booleano de SELinux. Los booleanos se encuentran en el directorio `/selinux/booleans`.

Módulos post-instalación

Las reglas básicas de SELinux tienen un abanico amplio de aplicaciones soportadas.

Los módulos de las políticas están por defecto en la ruta `/usr/share/selinux/default`. Es interesante entrar aquí y ver si hay algún módulo que interese habilitar. Por ejemplo, el módulo de `apache` es bastante probable que se emplee en la prueba de SELinux. Para habilitar el módulo solo es necesario el siguiente comando:

```
$ semodule -e apache
```

Es recomendable deshabilitar todos los módulos que no se utilizan. `semodule -l` muestra todos los módulos que están cargados en SELinux. El siguiente comando sirve para deshabilitar un módulo innecesario:

```
$ semodule -d modulo
```

4 PRUEBAS REALIZADAS

Programar sin una arquitectura o diseño en mente es como explorar una gruta sólo con una linterna: no sabes dónde estás, dónde has estado ni hacia dónde vas.

- Danny Thorpe -

Esta sección está dedicada a probar algunos de los módulos de seguridad del kernel sobre Raspbian con el kernel compilado con los módulos correspondientes.

4.1 TOMOYO Linux

4.1.1 Preparación del kernel.

Para poder emplear TOMOYO Linux son necesarios los siguientes módulos y opciones de seguridad en la compilación del kernel:

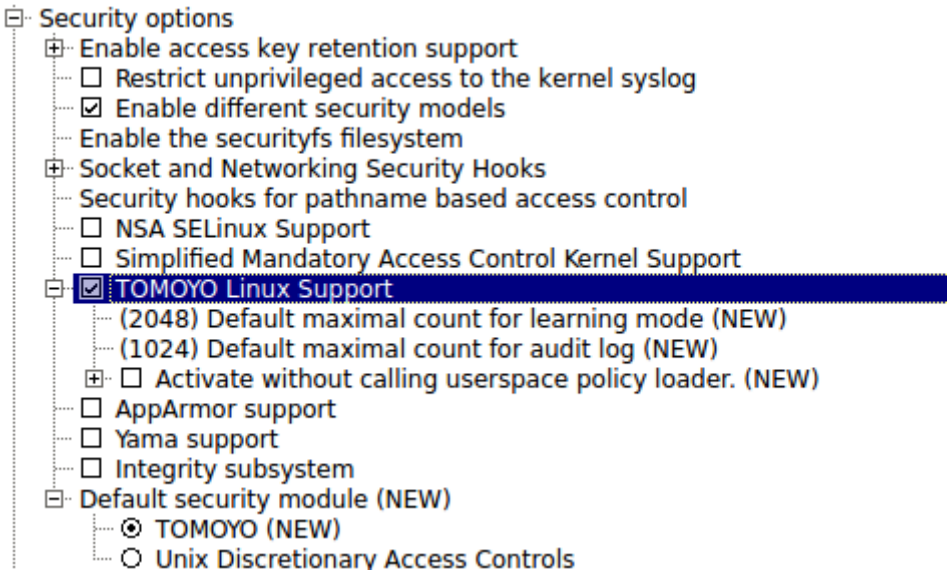

Figura 20 Configuración del kernel con soporte para TOMOYO

4.1.2 Instalación

Para hacer uso del módulo de seguridad TOMOYO Linux es necesario un conjunto de herramientas que se

instalan con el comando:

```
$ sudo apt-get install tomoyo-tools
```

Para arrancar las políticas de tomoyo:

```
$ sudo /usr/lib/tomoyo/init_policy
```

4.1.3 Ejemplo de Securización de Apache.

Los pasos a seguir son los siguientes:

En primer lugar, se abre el editor de políticas con el siguiente comando:

```
$ sudo tomoyo-editpolicy
```

Ahora es necesario cambiar a la pestaña `exception policy editor`. Para ello se pulsa '**w**' y luego '**e**'. La tecla para añadir una nueva directiva es '**a**'. El objetivo es securizar `apache` independientemente de cómo haya sido ejecutado, por ello se inicializa el dominio con la siguiente línea en `exception policy editor`:

```
initialize_domain /usr/sbin/apache2 from any
```

La siguiente imagen muestra justo debajo de la línea subrayada en verde cómo queda el nuevo dominio. También revela que el dominio está en el perfil 0, es decir, ningún tipo de restricción.

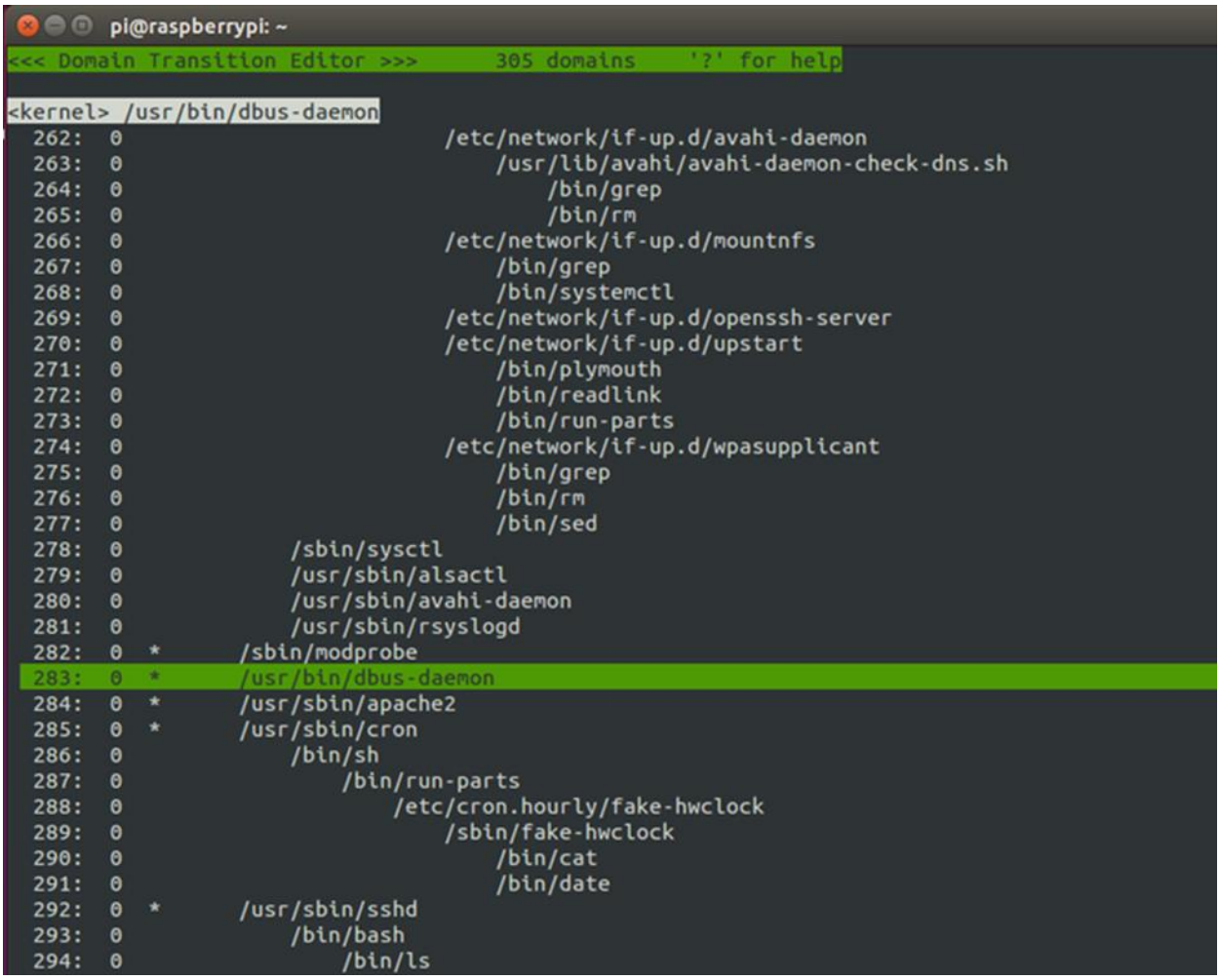

Figura 21 Domain Transition Editor

En `domain transition editor` se pulsa '**s**', se asigna el perfil 1 y se pulsa Intro. Tras estas acciones `apache` está en modo aprendizaje para reunir los permisos necesarios y comenzar la securización de este servicio.

```
$ /etc/init.d/apache2 restart
```

En este punto se deben realizar todas las acciones que permite el script de arranque de `apache`, accesos web etc para recolectar todos los permisos. El último paso es guardar los permisos tras realizar todas las pruebas necesarias.

```
$ /usr/sbin/tomoyo-savepolicy
```

El script `RcargarPolitica.sh` del repositorio [14] sirve para cargar la política en el kernel. El contenido del fichero se muestra a continuación:

```
#!/bin/bash
#RcargarPolitica.sh - script para cargar la política de tomoyo en el kernel.
```

```
/usr/sbin/tomoyo-loadpolicy -df < /etc/tomoyo/domain_policy.conf
/usr/sbin/tomoyo-loadpolicy -ef < /etc/tomoyo/exception_policy.conf
/usr/sbin/tomoyo-loadpolicy -p < /etc/tomoyo/profile.conf
/usr/sbin/tomoyo-loadpolicy -m < /etc/tomoyo/manager.conf
```

La siguiente acción a llevar a cabo es poner `apache` en modo obligatorio de la misma forma que antes se ha asignado el perfil 1 ahora se asigna el 3. Se puede ver en el `process state viewer` como todos los procesos de `apache` están en modo obligatorio. Es decir, perfil 3.

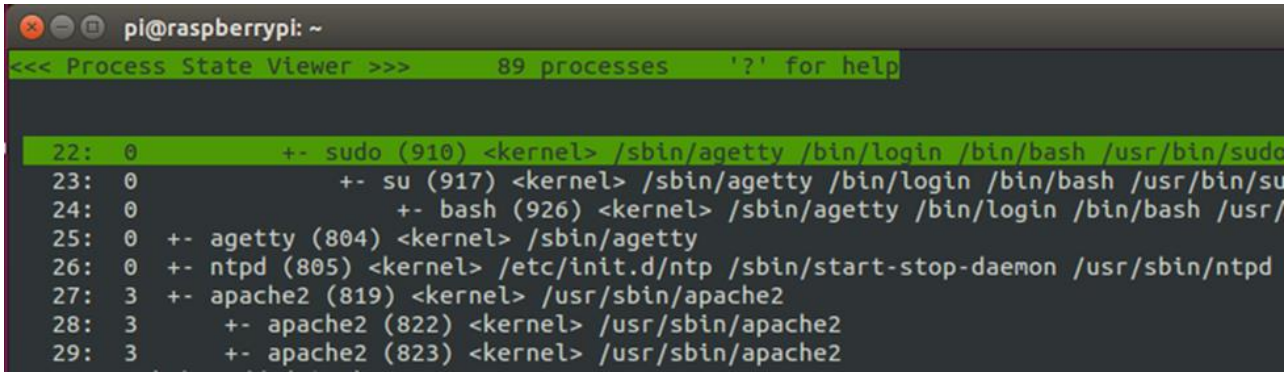

Figura 22 Procesos de Apache en modo obligatorio

Este ejemplo simula un error en la recolección de permisos mediante un acceso web. Durante la recolección de permisos nadie ha accedido a la web y una vez habilitado el modo obligatorio es imposible acceder.

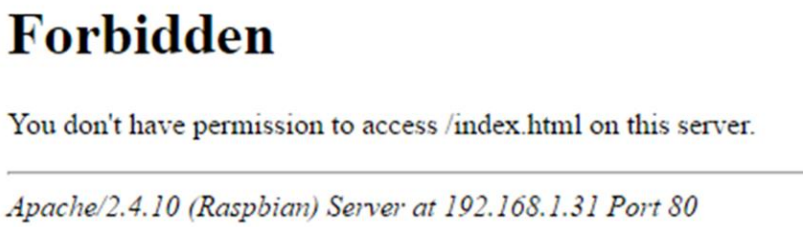

Figura 23 Ejemplo de error de permisos del perfil de tomoyo

La situación es la siguiente, la política tiene un error que debe subsanarse. Lo ideal sería activar el modo aprendizaje y añadir los permisos necesarios a la política. Sin embargo, en un sistema en producción no puede desactivarse el modo obligatorio. Para estas situaciones se emplea `tomoyo-queryd`, que sirve para arreglar estos errores:

```
$ sudo tomoyo-queryd
```

Este demonio gestiona los accesos y violaciones de políticas. En el momento que alguien realiza el acceso no permitido aparece un mensaje. En este caso una regla no deseada produce la violación de la política, por lo que

la acción a llevar a cabo es añadir a la política. El comando muestra la nueva regla de acceso añadida a la política.

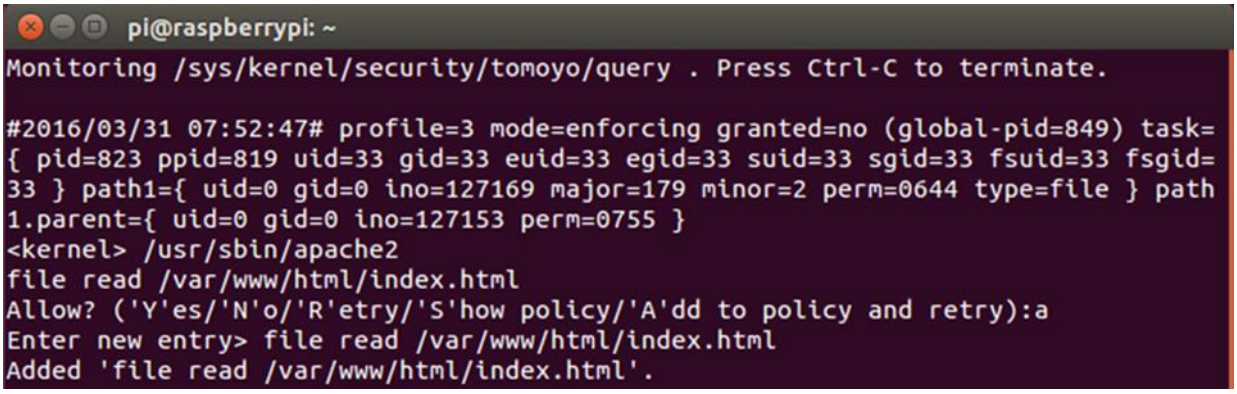

Figura 24 tomoyo-queryd para editar políticas

Una vez añadida la excepción a la política es posible el acceso. Como siempre, hay que guardar la política para que tomoyo no la borre al reiniciar.

```
$ /usr/sbin/tomoyo-savepolicy
```

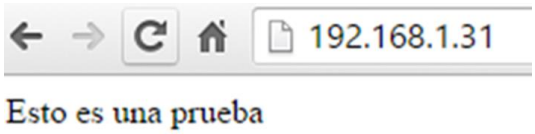

Figura 25 Ejemplo de permisos adecuados

Esto es solo un ejemplo de edición de la política. Para llevar a cabo una securización robusta hay que empezar por definir qué tiene que hacer exactamente el software y aplicar el proceso mostrado para perfilar la política.

4.2 AppArmor

Este apartado muestra un ejemplo de instalación y securización de `apache` con AppArmor. Este caso muestra un resumen de los pasos para generar un perfil que limita las acciones que puede realizar `apache`.

4.2.1 Preparación del kernel

Para obtener un kernel con soporte para AppArmor es necesario marcar las opciones mostradas en la siguiente figura en el menú de configuración del kernel.

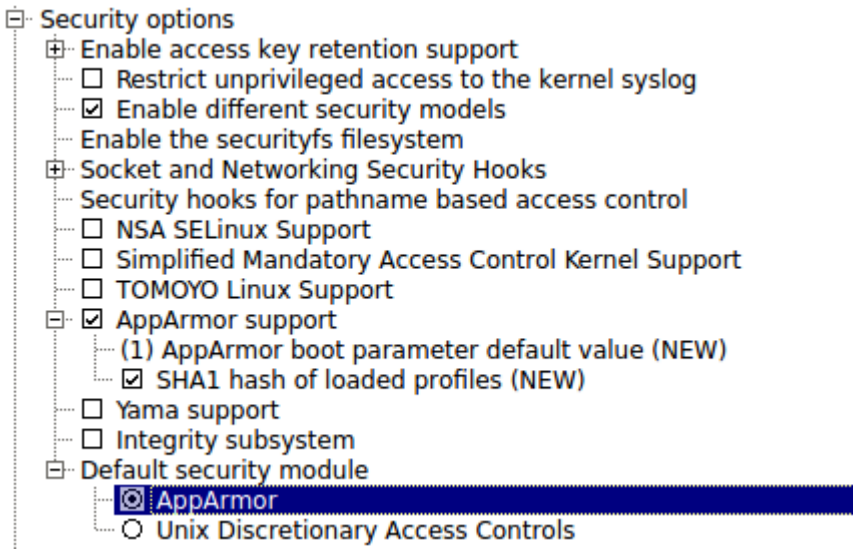

Figura 26 Configuración del kernel para AppArmor

4.2.2 Instalación

El primer paso es instalar los paquetes necesarios para AppArmor que emplearán los módulos que previamente han sido compilados en el kernel.

```
$ sudo apt-get install apparmor-profiles apparmor-utils auditd
```

Además, también es necesario tener instalado `apache`.

4.2.3 Generar un perfil.

Para generar un perfil para `apache` es necesario un plan de pruebas, en este caso las pruebas consistirán en reiniciarlo y realizar accesos a la web.

Se comienza conectando por ssh a la Raspberry y ejecutando:

```
$ sudo aa-genprof apache2
```

```
root@raspberrypi:/home/pi# aa-genprof apache2

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" option 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/sbin/apache2

[(S)can system log for AppArmor events] / (F)inish
```

Figura 27 Generación de un perfil para Apache

Esto pone el perfil para `apache` en modo aprendizaje. Es el momento de realizar las pruebas. Dichas pruebas habrán generado logs y ahora seleccionando la opción '`Scan`' apparmor busca los eventos en los logs del sistema.

Para cada evento AppArmor da la opción de permitir o denegar el acceso. De esta forma AppArmor asigna los permisos y capacidades necesarias para que la aplicación funcione.

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

Profile:    /usr/sbin/apache2
Capability: net_bind_service
Severity:   unknown

 [1 - #include <abstractions/nis>]
  2 - capability net_bind_service
[(A)llow] / (D)eny / (I)gnore / Audi(t) / Abo(r)t / (F)inish
```

Figura 28 Ejemplo de solicitud de permisos

Como muestra la figura anterior, cada evento genera permisos que van completando el perfil para `apache`. Tras cierto tiempo dedicado a esta tarea la aplicación funciona correctamente y de forma segura.

La siguiente figura representa cómo hay algunos permisos que pueden "globalizarse", es decir, si la aplicación necesita acceder a un directorio para acceder a un fichero, existe la opción de darle permisos para leer todo el directorio.

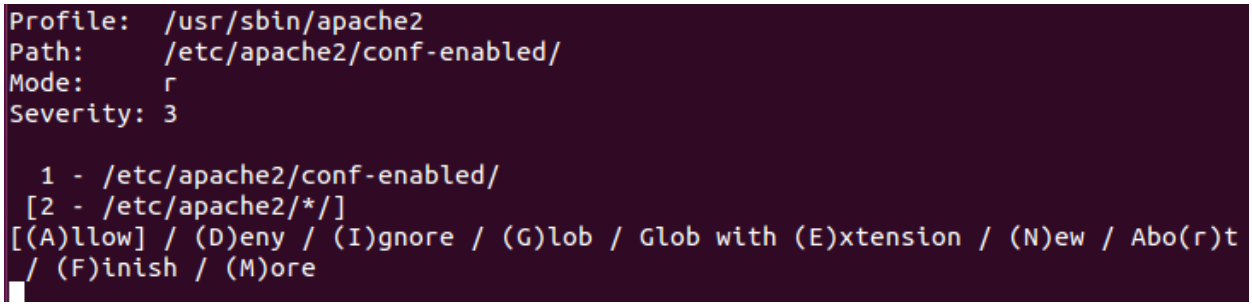

Figura 29 Ejemplo de globalización de permisos

Finalmente, AppArmor da la opción de guardar los cambios en el perfil.

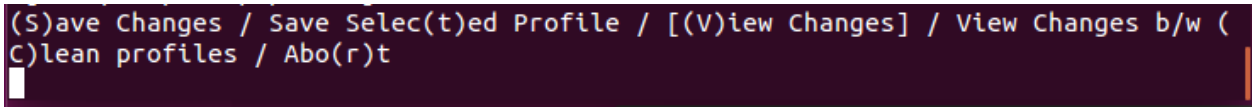

Figura 30 Guardar el perfil generado

4.2.4 Uso de la aplicación.

Una vez generado el perfil, el siguiente comando cambia el perfil a modo enforce:

```
$ sudo aa-enforce apache2
```

En caso de haber realizado un buen plan de pruebas en modo complain, todo funcionará correctamente en modo enforce. Tras realizar estas tareas el perfil generado para `apache` debe parecerse al de la siguiente figura.

```
root@raspberrypi:/etc/apparmor.d# cat usr.sbin.apache2
# Last Modified: Tue Apr  5 07:58:55 2016
#include <tunables/global>

/usr/sbin/apache2 {
  #include <abstractions/base>
  #include <abstractions/nis>


  capability net_bind_service,
  capability setgid,
  capability setuid,

  /etc/apache2/* r,
  /etc/apache2/*/ r,
  /etc/apache2/conf-available/* r,
  /etc/apache2/mods-available/* r,
  /etc/apache2/sites-available/* r,
  /etc/gai.conf r,
  /etc/group r,
  /etc/host.conf r,
  /etc/hosts r,
  /etc/ld.so.preload r,
  /etc/mime.types r,
  /etc/nsswitch.conf r,
  /etc/passwd r,
  /etc/resolv.conf r,
  /run/apache2/apache2.pid rw,
  /usr/lib{,32,64}/** mr,
  /usr/sbin/apache2 mr,
  /var/log/apache2/access.log a,
  /var/log/apache2/error.log w,
  /var/log/apache2/other_vhosts_access.log a,


  ^DEFAULT_URI flags=(complain) {
  }

  ^HANDLING_UNTRUSTED_INPUT flags=(complain) {
  }
}
```

Figura 31 Perfil generado

4.3 SELinux

Este apartado está dedicado a probar SELinux en la Raspberry Pi.

4.3.1 Preparación del kernel.

En primer lugar, para compilar NSA SELinux Support en el kernel, es necesario marcar la opción `Auditing Support` en la pestaña `General Support` del menú de configuración del kernel.

Por otro lado, hay un conjunto de opciones que deben marcarse como está detallado en la siguiente figura:

.

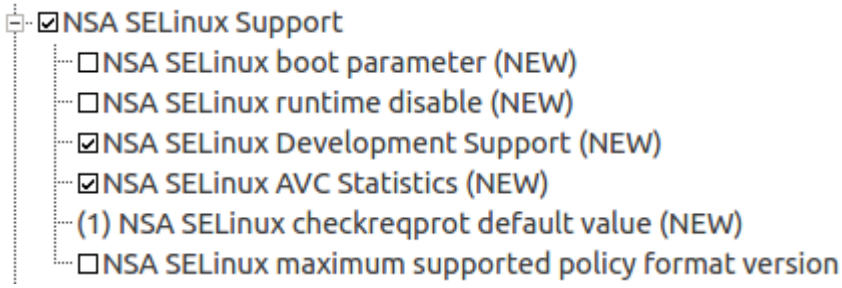

Figura 32 Configuración del kernel para su compilación con SELinux

Con esta configuración el sistema funciona con SELinux y es posible desarrollar políticas y monitorizar los registros y estadísticas. Existen otras configuraciones válidas, pero esta es la más adecuada al propósito y el dispositivo de este proyecto.

4.3.2 Instalación

Este módulo necesita una configuración de políticas y un sistema de ficheros etiquetado, para ello son imprescindibles políticas por defecto y herramientas. Las herramientas básicas para el uso de SELinux son las siguientes:

```
$ sudo apt-get install selinux-basics selinux-policy-default auditd
```

Estos paquetes son las políticas por defecto y el conjunto de herramientas básicas de SELinux. Además, una vez terminada la instalación hay que ejecutar el siguiente comando para terminar de configurar SELinux y el módulo PAM.

```
$ selinux-activate
```

La siguiente acción tras realizar la instalación y activación de SELinux es reiniciar. Durante el reinicio SELinux aplicará un autorelabel al sistema de ficheros.

```
$ sudo reboot
```

El reinicio puede durar entre 5 y 10 minutos porque todo el sistema de ficheros se etiquetará. Una vez iniciado el sistema, el siguiente comando verifica la instalación de SELinux:

```
$ check-selinux-installation
```

En este paso hay que editar el fichero `/boot/cmdline.txt` y añadirle `selinux=1 security=selinux`.

En este caso aparece un error:

```
FSCKFIX is not enabled – not serious, but could prevent system from booting…
```

La solución a este problema es editar el fichero `/etc/default/rcS` y dejar descomentada la siguiente línea:

```
FSCKFIX=yes
```

Esto activa la reparación automática de sectores defectuosos que SELinux detecte en el sistema de ficheros durante el arranque.

Es necesario un nuevo reinicio. Al arrancar SELinux estará funcionando. El comando `sestatus` permite comprobarlo.

```
root@raspberrypi:/home/pi# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             default
Current mode:                   permissive
Mode from config file:          permissive
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Max kernel policy version:      29
```

Figura 33 Ejemplo de ejecución de sestatus

El comando `audit2why -al` permite ver los logs de auditoría y qué procesos están incumpliendo las políticas actuales. Lo ideal es que no tener ningún mensaje de advertencia, aunque lo normal es encontrar muchos mensajes parecidos al de la siguiente figura.

```
type=AVC msg=audit(1467980036.792:226): avc:  denied  { execmod } for  pid=688
 comm="bash" path="/usr/lib/arm-linux-gnueabihf/libarmmem.so" dev="mmcblk0p2"
ino=9986 scontext=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 tconte
xt=system_u:object_r:lib_t:s0 tclass=file permissive=1
        Was caused by:
        The boolean allow_execmod was set incorrectly.
        Description:
        Allow allow to execmod

        Allow access by executing:
        # setsebool -P allow_execmod 1
```

Figura 34 Mensaje de advertencia de violación de política

El proceso para conseguir que el sistema funcione en modo obligatorio es en primer lugar seguir las recomendaciones generadas por `audit2why` y ejecutar los comandos que propone.

```
$ setsebool -P allow_execmod 1
$ setsebool -P allow_execmem 1
$ setsebool -P allow_execstack 1
$ setsebool -P httpd_execmem 1
```

En segundo lugar, para el resto de violaciones de políticas una opción es un módulo de políticas con el siguiente comando, donde "`mimodulo`" es el nombre que tendrá el módulo:

```
$ audit2allow -a -M mimodulo
```

La opción -M crea un fichero Type Enforcement (.te) con el nombre especificado en el directorio actual. Además, `audit2allow` compila el fichero Type Enforcement en un paquete de políticas (.pp).

```
root@raspberrypi:/home/pi# ls
mimodulo.pp  mimodulo.te
```

Figura 35 Módulo de políticas

Para instalar el módulo, ejecuta el siguiente comando:

```
$ semodule -i mi modulo.pp
```

Tras generar el módulo de políticas e instalarlo es el momento de editar el fichero de configuración de SELinux en `/etc/selinux/config` y poner SELinux en modo obligatorio.

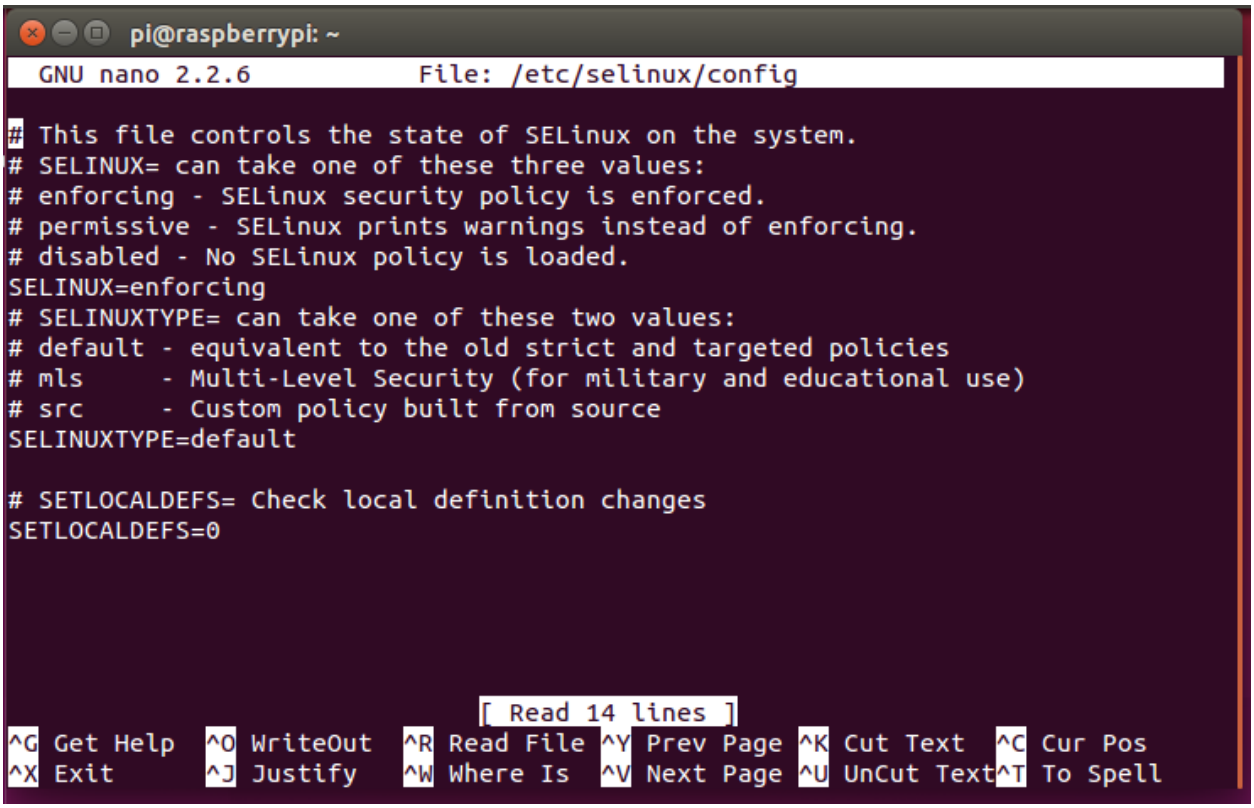

Figura 36 Fichero de configuración de SELinux

Para activar SELinux totalmente y habilitar el modo obligatorio es necesario reiniciar. Tras el reinicio la Raspberry Pi con `apache` está funcionando con SELinux en modo obligatorio.

Esta configuración inicial no es todo lo segura que debería ser la política de seguridad de un servidor, pero sirve como punto de partida para configuraciones más avanzadas que limiten el riesgo de amenazas de seguridad.

5 CONCLUSIONES Y TRABAJO FUTURO

El único sistema seguro es aquél que está apagado en el interior de un bloque de hormigón protegido en una habitación sellada rodeada por guardias armados

- Gene Spafford -

Este último capítulo está compuesto de dos apartados. El primero dedicado a hablar sobre las partes del proyecto y donde propone continuar el trabajo y el segundo apartado que establece las conclusiones del proyecto.

5.1 Trabajo futuro

Se proponen tres opciones para continuar este trabajo:

- A partir de las pruebas realizadas con tomoyo Linux, realizar una securización del sistema completo siguiendo los pasos descritos en este documento.
- Análogamente, realizar una securización completa del sistema mediante la generación de perfiles AppArmor.
- SELinux es un módulo de seguridad que restringe por defecto todas las acciones del sistema. Por tanto, cuando el software de la distribución tiene un comportamiento ligeramente diferente para el que las políticas por defecto se diseñaron, genera violaciones que son errores en el diseño de esas políticas. Esto ha llevado a encontrar gran cantidad de violaciones de políticas en modo permisivo y ha sido solucionado mediante la generación de un módulo de políticas que resuelve todos esos problemas usando el comando `audit2allow`. Pero esta solución puede mejorarse mediante la generación de un módulo para cada tipo de fallo. Por ejemplo, un módulo para violaciones relacionadas con `bash`, otro para `apache`, etc.

5.2 Conclusión

Este proyecto comienza a explorar las opciones de seguridad del kernel y hace una profundización en tres puntos Tomoyo Linux, AppArmor y SELinux. Realizando una recopilación de documentación básica que en muchos casos no es fácil de encontrar y trabajando en ella para que se adapte a lo que realmente un administrador encuentra al trabajar con estas herramientas. Esto es necesario debido a la desactualización y escasez de la documentación disponible. Por ello el trabajo ha sentado una base sobre la que puede continuarse en la técnica de securización.

También es importante destacar que gran parte del tiempo ha sido empleado en practicar mediante ensayo y error. Desde la compilación del kernel para la Raspberry Pi 2 que tiene sus peculiaridades, hasta las pruebas con los distintos módulos de seguridad del kernel.

Por otra parte, me gustaría destacar el gran aporte de conocimientos que me ha supuesto enfrentarme a las distintas filosofías y herramientas de securización del kernel tratadas en este proyecto. En el ámbito de la seguridad, tan importante hoy en día.

REFERENCIAS

[1] [En línea]. Available: http://elinux.org/RPi_SD_cards#SD_card_performance. [Último acceso: 21 07 2016].

[2] [En línea]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/. [Último acceso: 21 07 2016].

[3] [En línea]. Available: https://learn.adafruit.com/downloads/pdf/introducing-the-raspberry-pi-2-model-b.pdf. [Último acceso: 21 07 2016].

[4] «Raspberry Pi Foundation,» [En línea]. Available: https://www.raspberrypi.org/downloads/raspbian/. [Último acceso: 21 07 2016].

[5] «Repositorio raspberry pi linux,» [En línea]. Available: https://github.com/raspberrypi/linux . [Último acceso: 21 07 2016].

[6] [En línea]. Available: https://github.com/raspberrypi/tools. [Último acceso: 21 07 2016].

[7] J. Corbet, «TOMOYO Linux and pathname-based security,» [En línea]. Available: https://lwn.net/Articles/277833. [Último acceso: 01 04 2016].

[8] [En línea]. Available: http://tomoyo.osdn.jp/2.5/policy-specification/expression-rules.html.en#wildcard. [Último acceso: 21 07 2016].

[9] «manpages ubuntu,» [En línea]. Available: http://manpages.ubuntu.com/manpages/wily/en/man8/apparmor_parser.8.html. [Último acceso: 23 07 2016].

[10] «manpages ubuntu,» [En línea]. Available: http://manpages.ubuntu.com/manpages/wily/en/man8/aa-complain.8.html.

[11] «manpages ubuntu,» [En línea]. Available: http://manpages.ubuntu.com/manpages/wily/en/man8/aa-enforce.8.html.

[12] «manpages ubuntu,» [En línea]. Available: http://manpages.ubuntu.com/manpages/wily/en/man5/apparmor.d.5.html.

[13] «wiki apparmor,» [En línea]. Available: http://wiki.apparmor.net/index.php/Documentation.

[14] D. L. Ramírez, «RpiKernel (Repositorio de código del proyecto),» [En línea]. Available: https://github.com/dielenram/RpiKernel.

[15] «Sitio oficial WMWare,» [En línea]. Available: https://my.vmware.com/en/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/12_0. [Último acceso: 30 07 2016].

[16] «Sitio oficial Ubuntu,» [En línea]. Available: http://www.ubuntu.com/download/desktop. [Último acceso: 30 07 2016].

[17] «Sitio oficial Raspbian,» [En línea]. Available: https://www.raspberrypi.org/downloads/raspbian/. [Último acceso: 30 07 2016].

[18] «Sitio web de SDcard org,» [En línea]. Available: http://www.sdcard.org. [Último acceso: 30 07 2016].

[19] «sourceforge win32diskimager,» [En línea]. Available: http://sourceforge.net/projects/Win32diskimager/. [Último acceso: 30 07 2016].

[20] «TOMOYO Linux,» [En línea]. Available: http://tomoyo.osdn.jp. [Último acceso: 01 04 2016].

[21] P. González Pérez y C. Álvarez Martín, Hardenning de servidores GNU/Linux, 2013.

[22] M. Jang y R. Messier, Security Strategies in Linux Platforms and Applications, 2016.

[23] A. Ángel Ramos, J. P. García-Morán, F. Picouto, J. Grijalba, M. Mayan, Á. García, E. Inza y C. A. Barbero, Instala, administra, securiza y virtualiza entornos linux.

[24] «AppArmor Documentation,» [En línea]. Available: http://wiki.apparmor.net/index.php/Documentation. [Último acceso: 12 05 2016].

GLOSARIO

APIs:	Application Programming Interfaces
AVC:	Access Vector Cache
DAC:	Discretionary Access Control
EUID:	Effective UID
EVM:	Extended Verification Module
GCC:	GNU Compiler Collection
HMAC:	Código de autentificación de mensajes en clave-hash
IMA:	Integrity Measurement Architecture
LSM:	Linux Security Modules
MAC:	Mandatory Access Control
MCS:	seguridad multicategoría: Multi-Category Security
NSA:	National Security Agency
PAM:	Pluggable Authentication Modules, Módulos Externos de Autenticación
SELinux:	Security Enhaced Linux
SMACK:	Simplified Mandatory Access Control Kernel
TCG:	The Trusted Computing Group
TPM:	Trusted Platform Module
UID	User ID
UUID	Universally Unique Identifier o identificador único universal
Xattrs	Extended file Attributes

Anexo A: Manual de Configuración del Entorno

Este anexo detalla los pasos a seguir para hacer uso de la máquina virtual necesaria para compilar el kernel para la Raspberry 2 B. Esta es una posible implementación que funciona, pero no es la única.

Instalación de VMWare + Ubuntu en Windows 10.

-*Descarga de VMWare player:* [15]

Requiere reinicio tras la instalación.

-*Descarga e instalación de Ubuntu 64 bits:* [16]

Medios físicos en mi caso: 4 GB RAM, 32 GB Disco Duro y 3 procesadores.

Para poder usar la Micro SHDC tras la instalación hay que seguir los siguientes pasos:

- Es necesario obtener el número de disco que tiene asignado la tarjeta haciendo click derecho sobre "Este equipo" y "Administrar".

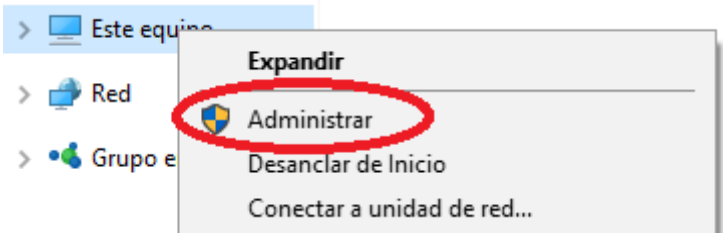

Figura 37 Obtención del número de disco

Dentro de la nueva ventana, Figura 38, pulsando en Administración de discos, aparece qué número de disco tiene la tarjeta. Este número se empleará en el siguiente paso.

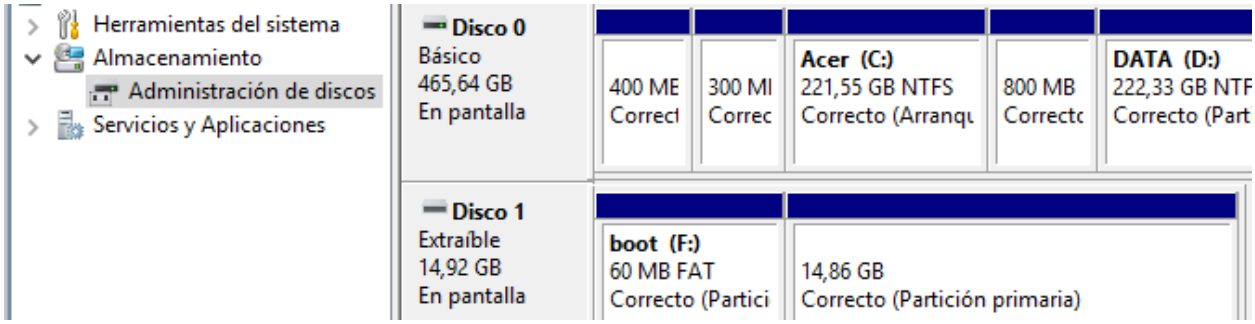

Figura 38 Obtención del número de disco 2

Primero se abre VMWare y luego se siguen los siguentes pasos:

- seleccionar Ubuntu
- seleccionar Player
- seleccionar Manage
- seleccionar Virtual Machine Settings
- seleccionar Add
- seleccionar Hard Disk
- seleccionar IDE
- seleccionar Use a physical disk (for advanced users), se elige el número de disco que se observó en el paso 1
- seleccionar “Use Entire Disk”.

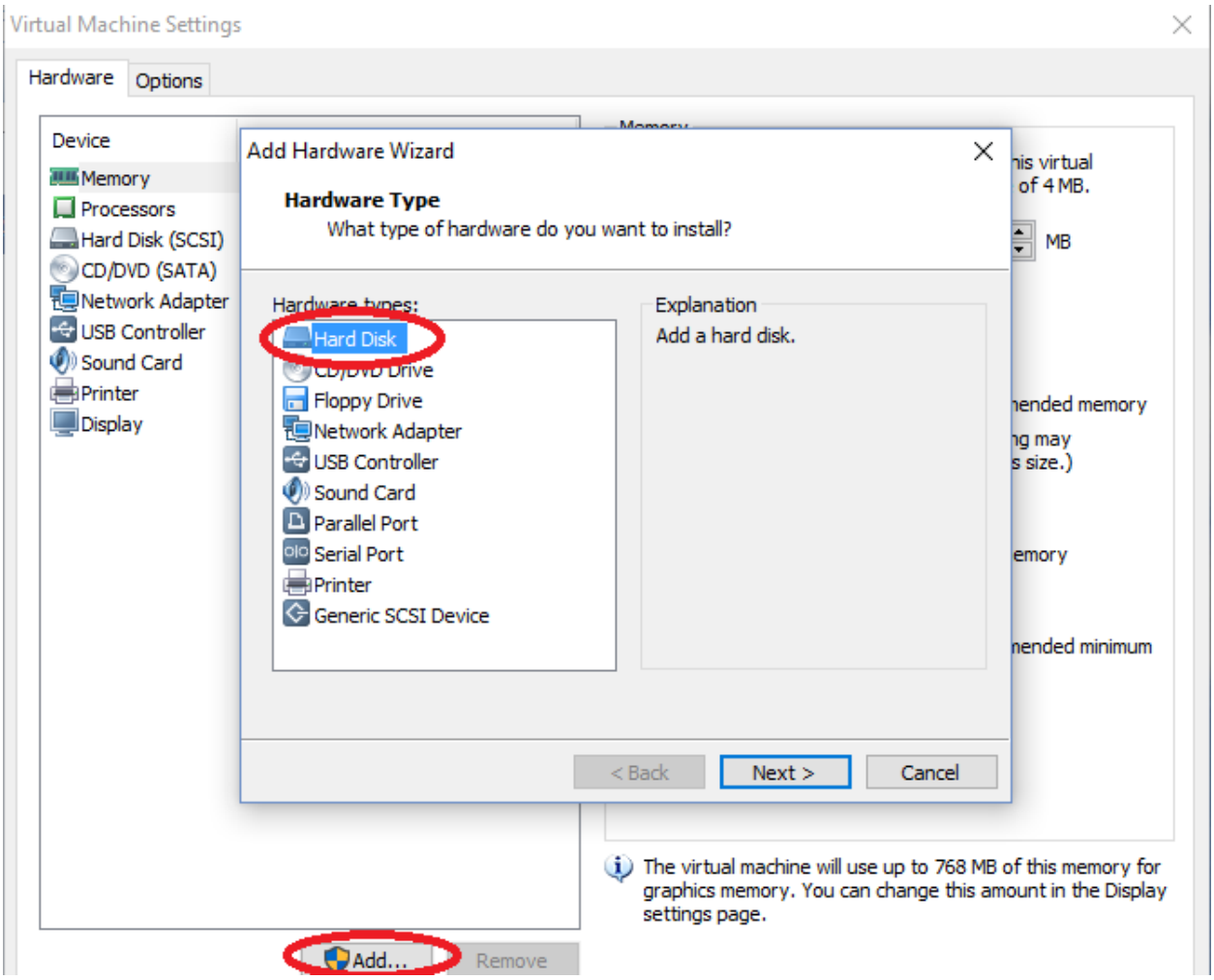

Figura 39 Configuración de VMWare para leer la tarjeta

En caso de arrancar la máquina virtual sin la SHDC dará error. Por ello, para trabajar con Ubuntu sin la tarjeta hay que eliminar el disco duro añadido de la máquina virtual.

Formatear la Micro SHDC y copiar una imagen de sistema.

La imagen se obtiene de [17].

Descargar RASPBIAN JESSIE LITE y comprobar el SHA-1.

En Windows

Se descarga el programa SD Card Formatter desde la web [18] en la pestaña Download y luego SD Card Formatter. Una vez descargado y ejecutado el programa, en la opción Drive se elige el dispositivo y se pulsa en Format.

También es necesario el programa WIN32diskimager para obtenerlo hay que acceder a la web [19] tras descargarlo e instalarlo, se selecciona la imagen y se pulsa write.

En Linux :

Es necesario saber qué dispositivo es la tarjeta. Para ello se ejecuta el siguiente comando:

```
$ sudo fdisk -l
```

Luego se inserta la tarjeta y se vuelve a ejecutar. El nuevo dispositivo que aparece es la tarjeta.

```
$ sudo fdisk -l
```

Para asegurar que las particiones están desmontadas utiliza:

```
$ sudo umount /dev/sdX
$ sudo fdisk /dev/sdX
```

Para borrar las particiones se pulsa '**d**'.

Para crear una nueva partición introduce '**n**', se elige primaria, se asigna al número 1 y tamaño máximo por defecto, que se traduce en pulsar dos veces Intro.

Para cambiar el tipo de partición se elige la opción '**t**', se elige la 1, y el tipo b (W95 FAT32).

Por último, se guardan los cambios con '**w**'.

Tras esto, se formatea la tarjeta con:

```
$ sudo mkfs.vfat /dev/sdX
```

Por ultimo, quema la imagen en la tarjeta:

```
$ sudo dd if=/dev/sdX of=/PathToImage bs=1M
```

Compilación cruzada del kernel e inserción en la tarjeta.

Primero los pasos que solo hay que seguir una vez:

- Instalar git:

```
$ sudo apt-get install git
```

- Añadir dependencias perdidas:

```
$ sudo apt-get install bc
```

- menuconfig requiere de ncurses:

```
$ sudo apt-get install libncurses5-dev
```

- En caso de preferir usar xconfig (modo gráfico) habría que instalar:

```
$ sudo apt-get install qt4-qmake libqt4-dev
```

- Instalación de la toolchain en el directorio ~/:

```
$ git clone https://github.com/raspberrypi/tools
```

Algunos make necesitan permisos de superusuario, es necesario añadir una ruta al PATH y `sudo` cambia las variables de entorno. Por tanto, ser usarán dos consolas una como `root` y otra como usuario no privilegiado en este ejemplo el usuario es "diego". La siguiente línea es la ruta que necesaria para simplificar los make

```
export PATH=/home/diego/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin/:$PATH
```

Para `root` se edita `.bashrc` en el directorio de `root`. Sólo hay que añadir el export al final. Para usuario no privilegiado se edita `~/.profile,` se añade el export al final y sólo falta reiniciar la máquina.

El siguiente paso es la obtención del código para configurar y compilar el kernel:

```
$ git clone --depth=1 https://github.com/raspberrypi/linux
```

Ahora las acciones que van a repetirse habitualmente, es recomendable emplear los script del siguiente apartado: (Se usa "//" al principio de las frases para distinguir las aclaraciones de los comandos).

```
$ cd linux
$ KERNEL=kernel7

// Para una configuración por defecto emplea el siguiente comando:
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcm2709_defconfig

// Para una configuración personalizada utiliza:
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
// o este otro para modo gráfico:
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- xconfig

// Cualquiera de los dos comandos anteriores genera un fichero .config para compilar.

// Ahora hay que construir el kernel, en esta máquina el compilador tiene asignado 3 procesadores con la opción -j pero eso depende de la máquina utilizada.
$ make -j3 ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage modules dtbs

// Estos comandos instalan directamente el kernel en la tarjeta
$ mkdir mnt/fat32
$ mkdir mnt/ext4
$ sudo mount /dev/sdX1 mnt/fat32
$ sudo mount /dev/sdX2 mnt/ext4
```

```
  // Nota muy importante: Los siguientes comandos necesitan permisos de
superusuario (sudo su) y no se puede utilizar ‘sudo comando’ porque cambia
las variables de entorno.

  # KERNEL=kernel7
  # make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
INSTALL_MOD_PATH=mnt/ext4 modules_install

  # cp mnt/fat32/$KERNEL.img mnt/fat32/$KERNEL-backup.img
  # scripts/mkknlimg arch/arm/boot/zImage mnt/fat32/$KERNEL.img
  # cp arch/arm/boot/dts/*.dtb mnt/fat32/
  # cp arch/arm/boot/dts/overlays/*.dtb* mnt/fat32/overlays/
  # cp arch/arm/boot/dts/overlays/README mnt/fat32/overlays/
  # umount mnt/fat32
  # umount mnt/ext4
```

Esta forma de instalar el nuevo kernel permite mantener una copia de seguridad de la version anterior.

Modo de empleo de algunos scripts útiles.

Los siguientes scripts reúnen los comandos explicados anteriormente de forma intuitiva y fácil de usar. Dependiendo de la implementación elegida, puede que necesiten modificación. Estos scripts están diseñados para las condiciones concretas de mi equipo de trabajo.

Partiendo de que ya se ha clonado el repositorio [14].

Dentro de la carpeta scripts están los siguientes scripts:

- `menuScript.sh` - Script de arranque de la interfaz de configuración del kernel.

  Incluye algunos comandos con comentarios que permiten generar una configuración por defecto o elegir configuración por `menuconfig` en vez de por `xconfig`.

- `compilar.sh` - Script de compilación del kernel. Adaptado a mi máquina virtual que tiene 3 procesadores con –j3.

- `Rmontarsd.sh` - Script para montar la micro sd card. En caso de añadirla a la máquina virtual empleando el procedimiento anteriormente descrito. Debe ejecutarse como `root`.

- `Rinstalarsd` - Script para instalar el kernel en la sd. Debe ejecutarse como `root`.

Por último, para facilitar la ejecución de los comandos debe añadirse al PATH el directorio /scripts del repositorio clonado.

Instalación y configuración del software necesario.

Una vez la Raspberry está preparada para arrancar por primera vez son necesarios algunos ajustes e instalaciones. Para empezar, el teclado viene para una distribución de otro país, para cambiarlo se ejecuta:

```
$ sudo dpkg-reconfigure keyboard-configuration
```

Tras una breve configuración, se establece la configuración elegida:

```
$ sudo setupcon
```

Otro aspecto necesario en la Raspberry es ampliar la partición para que se use toda la capacidad de la tarjeta. El software de la Raspberry proporciona la herramienta necesaria para ello:

```
$ sudo raspi-config
```

En el menú desplegado, la primera opción soluciona este problema. Tras ello solo queda reiniciar.

El servidor web `apache` se usará en las pruebas de algunos módulos. Por lo que será necesario instalarlo:

```
$ sudo apt-get install apache2
```

Por otro lado, también será necesario emplear el repositorio creado para este proyecto en la propia Raspberry Pi por lo que es necesario instalar el software git:

```
$ sudo apt-get install git
```

Un aspecto importante desde el punto de vista de la seguridad es partir de un sistema actualizado, para ello se ejecuta:

```
$ apt-get update
$ apt-get upgrade
```