PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Facultad de Ingeniería

Escuela de Ingeniería Informática

Desarrollo de Herramientas de Configuración de Políticas de Seguridad Para Selinux

Tesis Para Optar al Título de Ingeniero en Ejecución en Informática

Por :

María Luisa Arancibia – Cristian Orellana Vega

Profesor Guía : Iván Mercado B.

Enero, 2004

RESUMEN

Security Enhanced Linux (SELinux) es una extensión al kernel estándar de Linux que ha sido diseñado para aplicar estrictos controles de acceso. SELinux permite confinar a los procesos a la mínima cantidad de privilegios que requieran.

SELinux es una implementación práctica de siendo aplicado en un sistema operativo del mundo real. Con el desarrollo de SELinux, es posible crear sistemas con nivel de seguridad muy fuerte, sistemas que incluso pueden resistir ataques a través de vulnerabilidades en programas ejecutándose con altos niveles de privilegios.

Configurar las políticas de seguridad de SELinux es complicado, propenso a errores y no hay herramientas que faciliten esa tarea. Este proyecto entrega herramientas capaces de facilitar esa tarea, evitando así el manejo directo de los extensos archivos de configuración, obteniendo una comprensión mas clara de la política configurada.

ABSTRACT

Security Enhanced Linux (SELinux) is an extension to the standard Linux kernel that has been designed to enforce strict access controls. SELinux lets you confine processes to the minimum amount of privilege they require.

SELinux is a practical implementation of Mandatory Access Control being applied to a real-world operating system. By deploying SELinux, it is possible to create systems with a very strong level of security, systems that can even resist being attacked through vulnerabilities in programs running at the highest levels of system privilege.

Configuring security policies of SELinux is complicated, prone to errors and there is not tools that facilitate that work. This project deliver tools capable of facilitate that work, avoiding direct management of extensive configuration files, acquiring more clear vision of configurated policy.

1. INTRODUCCIÓN

Desde los primeros sistemas informáticos diseñados, hasta los existentes en la actualidad, ha ido aumentando la cantidad y también los tipos de datos que controlan y manipulan los cada vez más potentes computadores.

Con anterioridad a la aparición de la informática, los datos se mantenían seguros por medios físicos como archivadores con cerradura de combinación, pero con la aparición del computador es necesario el uso de otras herramientas que impidan el acceso a intrusos que desean obtener información de sistemas computacionales ajenos, o aprovecharse de las deficiencias en alguna aplicación para tomar el control de esos sistemas o simplemente dañarlos.

El incluir controles para la seguridad en el sistema operativo se ha convertido en la tendencia actual y de futuro. Con diversos proyectos para implementar seguridad en el sistema operativo, tan cerca del kernel como sea posible, seleccionar uno puede ser complicado, debido a las limitaciones a las que puede someter.

Dado que la mayoría de estos proyectos están en sus primeras etapas, las protección que pueden ofrecer aún dista mucho de lo que se quisiera. Esto se debe a las diferencias de enfoque que toman cada uno de estos proyectos, por los niveles de seguridad que pueden ofrecer, la flexibilidad en la definición de lo que se conocerá como “seguridad” y la forma en que se aplica.

Para responder a las necesidades de seguridad de los sistemas, se ha desarrollado una arquitectura que permite la implementación de políticas de seguridad al nivel del kernel del sistema operativo. Esta arquitectura es denominada Flux Advanced Security Kernel, Flask, y es el resultado de los esfuerzos conjuntos de la National Security Agency, NSA, con

asistencia de la Universidad de Utah y la Secaré Computing Corp., SCC. En la arquitectura Flask, la lógica de la política de seguridad esta encapsulada dentro de un componente separado dentro del núcleo del sistema operativo, junto con una interfaz general para obtener decisiones acerca de ésta.

La NSA está implementando la arquitectura Flask en Linux, proyecto denominado Security-Enhanced Linux, SELinux, que destaca por su nivel de integración con el sistema operativo y la característica de ofrecer flexibilidad en las políticas de seguridad. Siendo este un proyecto de investigación con poca prioridad y bajos recursos, sus desarrollo va algo lento, pero la parte mas fuerte de la implementación ya esta desarrollada. Ello por ser una "adaptación" de un proyecto anterior similar, pero implementado en microkernel. Este proceso de integración aún no se ha completado, y varias cosas cambian durante los distintos pasos en el avance de la integración con Linux, debido a las diferencias y limitaciones entre el kernel de Linux y el microkernel utilizado originalmente en Flask.

A pesar de todos los cambios que pueda sufrir esta iniciativa de la NSA, esta no contempla, en el mediano plazo, el desarrollo de herramientas de configuración de ninguna clase, debido a que mantienen sus esfuerzos centrados en la implementación de la arquitectura Flask en Linux. Tampoco existe documentación "para novatos", por lo que es necesario estudiar la documentación técnica que han desarrollado, para lograr comprender el funcionamiento de SELinux, las capacidades de configuración, lo que ha sido implementado de la arquitectura Flask y lo que no, los conceptos definidos en la arquitectura Flask, así como de los elementos que serán reemplazados por otros a medida que avance la integración del modelo Flask en Linux. La documentación inicialmente fue escasa, pero al momento de iniciarse este proyecto, SELinux comenzó a tener un cierto aumento en sus actividades, y durante la evolución de este proyecto aparecieron nuevos documentos técnicos por parte de la NSA, nuevas versiones del kernel SELinux y nuevos esfuerzos de integración en otras distribuciones Linux (NSA inició la integración de Flask

en Linux utilizando Red Hat Linux y, dado que el proyecto SELinux está en fase de desarrollo, no pueden dividir los esfuerzos portando SELinux a otras distribuciones).

2. OBJETIVOS

2.1 Objetivo General

Utilizando la implementación en SELinux de la arquitectura Flask, el proyecto consiste en investigar el funcionamiento de esa arquitectura. Como resultado de este proceso de investigación, será posible desarrollar herramientas que permitan construir las políticas de seguridad, permitiendo manejar los múltiples archivos de configuración. Evitando el manejo directo de los extensos archivos, logrando así una comprensión más clara de la política configurada y una propensión menor a los errores que ello conlleva.

2.2 Objetivos Específicos

Elaborar un resumen con las herramientas disponibles para SELinux, y así determinar el área donde no hay desarrollo, o continuar con el complemento de las ya desarrolladas.

Elaborar un resumen de los archivos de configuración usados por SELinux, su importancia y significado, para así determinar que tipo de políticas manipularán las herramientas a construir.

Desarrollar aplicaciones con interfaz gráfica, proporcionando las facilidades típicas de las herramientas de configuración Linux basadas en cuadros de diálogo, para la visualización de las propiedades de la política de seguridad y su actualización interactiva por el usuario (una aplicación independiente para cada función):

- administración de usuarios

- administración de roles, tipos, reglas.

3. SEGURIDAD EN SISTEMAS OPERATIVOS

La responsabilidad de implantar y mantener las medidas de seguridad en un sistema de información recae en el sistema operativo. A medida que los sistemas operativos evolucionan, los requisitos de seguridad son cada vez mayores y más difíciles de implantar. El que los sistemas sean fáciles de usar para el usuario, junto al desarrollo de redes y sistemas distribuidos hacen que los sistemas sean más vulnerables a ataques. Por otro lado, cada vez más información crítica se almacena y transmite de forma electrónica. Los mecanismos de seguridad deben garantizar que los sistemas funcionen sin interrupción y que la información se mantenga sin alteración.

Diferentes ambientes computacionales, y las aplicaciones que corren en ellos, tienen diferentes requerimientos de seguridad. Debido a que cualquier noción de seguridad está capturada en la expresión de una política de seguridad, existe una necesidad de varias políticas diferentes e incluso varios tipos de políticas. Para ser aceptable en general, cualquier solución de seguridad de computador debe ser lo suficientemente flexible para soportar este amplio rango de políticas de seguridad. Incluso, en los ambientes distribuidos de hoy, esta flexibilidad de políticas debe ser soportada por los mecanismos de seguridad del sistema operativo.

Los sistemas operativos deben ser flexibles en su soporte de políticas de seguridad, entregando mecanismos suficientes para soportar la amplia variedad de políticas de seguridad del mundo real. Esta flexibilidad requiere controlar la propagación de los derechos de acceso, reforzando los derechos de acceso de granularidad fina y soportando la revocación de derechos de acceso previamente concedidos.

3.1 Flask

Flask es una arquitectura de seguridad de sistema operativo que provee soporte flexible para políticas de seguridad. La arquitectura fue prototipada en el sistema operativo de investigación Fluke. Varias de las interfaces y componentes fueron portadas desde el prototipo Fluke hacia el OSKit.

En 1992 y 1993, investigadores en la NSA y SCC trabajaron en el diseño e implementación de DTMach, una consecuencia natural del proyecto TMach y el proyecto LOCK. DTMach integró una generalización de Type Enforcement, TE, un mecanismo de control de acceso flexible en el microkernel Mach. El proyecto DTMach fue continuado en el proyecto DTOS. Luego de completarse este, surge un nuevo esfuerzo conjunto, ahora de la NSA, SCC y el proyecto Flux de la Universidad de UTA, para transferir la arquitectura de seguridad en el sistema operativo de investigación Fluke. Durante esta integración, la arquitectura fue mejorada para proveer un mejor soporte a las políticas de seguridad dinámicas. Esta nueva arquitectura es la que fue denominada Flask.

3.2 SELinux:

Implementación de la arquitectura Flask en Linux. Desarrollado por la NSA, para compartir los resultados de las investigaciones de Flask en una comunidad más grande de desarrollados.

Así, al reconocer el rol critico de los mecanismos de seguridad en el sistema operativo para soportar la seguridad en niveles mas altos, los investigadores de la Information Assurance Research Group de la NSA, han estado investigado una arquitectura que puede proveer la

funcionalidad de seguridad necesaria de una forma que pueda ser apropiada a las necesidades de seguridad de un amplio rango de ambientes computacionales.

Esta versión de Linux tiene una fuerte y flexible arquitectura de control de acceso obligatorio en los principales subsistemas del Kernel. El sistema provee un mecanismo para reforzar la separación de información basada en la confidencialidad y requerimientos de integridad. Esto permite que las amenazas de saboteo y mecanismos de salto de seguridad de aplicaciones sean direccionadas y habilita el confinamiento del daño que pueda ser causado por aplicaciones maliciosas o defectuosas.

3.2.1 Conceptos y Definiciones de la Arquitectura Flask

La arquitectura de seguridad del sistema operativo Flask proporciona soporte flexible para políticas de Control de Acceso Mandatario, MAC, [SpencerUsenixSec1999]. La implementación SELinux de la arquitectura Flask esta descrita en [LoscoccoFreenix2001]. A continuación, se describen los conceptos definidos por la arquitectura Flask, cuya importancia y significado es necesario conocer a la hora de manipular una política de de seguridad SELinux.

3.2.2 Conceptos Flask

Cada sujeto (proceso) y objeto (archivo, socket, etc) en el sistema están asignados a una colección de atributos de seguridad, conocidos como *contexto de seguridad*. Un contexto de seguridad contiene todos los atributos de seguridad relevantes para la política de seguridad asociados a un sujeto u objeto en particular. El contenido y el formato de un contexto de seguridad depende del modelo de seguridad, así un contexto de seguridad solo

puede ser interpretado por el *servidor de seguridad*. Para encapsular mejor los contextos de seguridad y para proporcionar mayor eficiencia, el código de aplicación de política de SELinux típicamente maneja identificadores de seguridad, SIDs, en vez de contextos de seguridad. Un SID es un entero que está mapeado, por el servidor de seguridad, a un contexto de seguridad, en tiempo de ejecución. Los SIDs son identificadores locales y no persistentes, y deben ser traducidos a contextos de seguridad para el etiquetado de objetos persistentes como archivos o trabajo en red etiquetado. Un pequeño conjunto de valores SID son reservados para la inicialización del sistema u objetos predefinidos; a estos valores SID se les denomina SIDs iniciales.

Cuando es necesario tomar una decisión de seguridad, el código de aplicación de políticas entrega un par de SIDs (típicamente el SID del sujeto y el objeto, pero algunas veces un par de SIDs de sujeto o un par de SIDs de objeto), como por ejemplo un proceso, un archivo regular, un directorio, un socket TCP, etc. El servidor de seguridad busca el contexto de seguridad para los SIDs. Luego toma su decisión basándose en los atributos que poseen los contextos de seguridad y en la clase de seguridad del objeto. El servidor de seguridad proporciona dos categorías importantes de decisiones de seguridad para el código de aplicación de la política de seguridad: decisiones de etiquetado y decisiones de acceso.

Las decisiones de etiquetado, también llamadas decisiones de cambio (transición), especifican los atributos de seguridad predeterminados a usar para un nuevo sujeto u objeto. Se solicita una decisión de cambio de proceso cuando se ejecuta un programa, basándose en el SID actual del proceso y el SID del programa. Se solicita una decisión de cambio de objeto cuando se crea un nuevo objeto, basándose en el SID del proceso que lo crea y el SID de un objeto relacionado, por ejemplo, el directorio padre en la creación de archivos. Estas decisiones de etiquetado predefinidas pueden ser anuladas manualmente por aplicaciones conscientes de la seguridad utilizando las nuevas llamadas de sistema SELinux. En otro caso, el uso de la nueva etiqueta debe ser aprobada por una decisión de acceso o la operación fallará.

El código de aplicación de la política es responsable de unir las etiquetas a los sujetos y objetos en el sistema. Para objeto transitorios, como procesos o sockets, se puede asociar un SID con el objeto correspondiente del kernel. Sin embargo, los archivos y

directorios son persistentes y requieren soporte adicional para proporcionar etiquetado persistente. Un mapa con el etiquetado persistente es almacenado en cada sistema de archivos, y este especifica el contexto de seguridad para cada archivo y directorio en ese sistema de archivos.

El mapa de etiquetas persistentes esta particionado en dos archivos de mapeo, uno que une inodes a enteros de identificadores de seguridad persistentes, PSIDs, y otro que une los PSIDs a los contextos de seguridad. El mapeo es inicializado por una utilidad llamada **setfiles** a partir de un archivo de configuración de *contextos de archivo* que especifica contextos de seguridad para archivos, basándose en expresiones regulares de nombres de ruta. Subsecuentemente, el mapa es mantenido dinámicamente por el código de aplicación de la política, para reflejar las operaciones de creación, eliminación y re-etiquetado.

El archivo de configuración de contextos de archivo es separado lógicamente de la configuración de la política. Este archivo sólo es utilizado por la utilidad **setfiles** y solo es necesario cuando se inicializa o reinicia el mapa de etiquetas persistentes. El mapa de etiquetas persistentes es utilizado en tiempo de ejecución por el código de aplicación de la política. La configuración de la política es utilizada por el servidor de seguridad y es necesaria para obtener decisiones de seguridad. La configuración de la política especifica decisiones de seguridad basándose completamente en lo atributos de seguridad, en tanto que el archivo de configuración de contextos de archivo especifica contextos de seguridad para archivos basándose en los nombres de ruta.

Las decisiones de acceso especifican cuando se debe conceder un permiso para un par dado de SIDs y clase. Cada clase de objeto tiene asociado un conjunto definido de permisos asociados para controlar las operaciones en objetos con esas clases. Estos conjuntos de permisos son representados por un mapa de bits llamado vector de acceso.

Cuando se solicita una decisión de acceso, el servidor de seguridad retorna un vector de accesos permitidos conteniendo las decisiones para todos los permisos definidos para la clase de objeto. Este vector de acceso es almacenado (cached) por un componente de la arquitectura Flask llamado Cache de Vector de Acceso, AVC, para usos posteriores por parte del código de aplicación de política. El componente AVC proporciona una interfase al servidor de seguridad para soportar la administración del caché durante cambios en la política.

Además de proporcionar unos vectores de accesos permitidos, el servidor de seguridad proporciona dos vectores de acceso relacionados con la auditoria. Una decisión *auditallow* indica que siempre debe ser auditada la concesión del permiso cuando este sea concedido. Por ejemplo, si un permiso esta asociado con una operación muy delicada, puede ser deseable auditar cada uso de esa operación. Una decisión *auditdeny* indica que el permiso debe ser auditado cuando se niegue el permiso.

3.2.3 Definiciones Flask

Un pequeño conjunto de archivos de configuración son compartidos entre el módulo del kernel SELinux y la configuración de política de ejemplo. Estos archivos definen las clases de seguridad Flask, los SIDs iniciales y los permisos del vector de acceso. Esta información no es específica a un modelo de seguridad en particular, y rara vez debiera ser cambiada. Los cambios a estos archivos requieren la recompilación del módulo y de la política y, típicamente, requiere actualización del módulo y de la política para que puedan usar los nuevos valores. Estos archivos no debieran ser modificados para configurar el modelo de seguridad implementado por el servidor de seguridad de ejemplo.

El significado de las clases de seguridad y permisos del vector de acceso en la implementación original de SELinux está descrita en [LoscoccoNSATR2001] y en [LoscoccoFreenix2001]. Los cambios hechos para la implementación de SELinux basada en LSM está descrita en [SmalleyModuleTR2001].

Los originales de estos archivos están ubicados en el subdirectorio Flask del módulo SELinux (en security/selinux del árbol del kernel-LSM parchado). Estos archivos son instalados en /usr/local/selinux/Flask y son utilizados cuando es compilada la configuración de la política de ejemplo. Estos archivos son:

Tabla 1.

Nombre de archivos	**Descripción**
security_classes	Declara las clases de seguridad
initial_sids	Declara los SIDs iniciales
access_vectors	Define los permisos del vector de acceso para cada clase

3.2.4 Modelo de Seguridad

El servidor de seguridad de ejemplo implementa un modelo se seguridad que es una combinación del modelo Type Enforcement, TE, y el modelo Role-Based Access Control, RBAC. El modelo TE proporciona control de granularidad fina sobre procesos y objetos en el sistema, y el modelo RBAC proporciona un alto nivel de abstracción para simplificar la administración de usuarios.

3.2.4.1 Modelo TE

Un modelo TE tradicional une un atributo de seguridad denominado dominio a cada proceso, y une un atributo de seguridad denominado tipo a cada objeto. El modelo TE tradicional trata a todos los procesos en el mismo dominio y trata a todos los objetos que tienen el mismo tipo de la misma forma. Por lo tanto, dominios y tipos pueden ser vistos como clases equivalentes de seguridad. Un par de matrices de acceso especifica como los dominios pueden acceder a los tipos y como los dominios pueden interactuar con otros dominios. Cada usuario está autorizado para operar en ciertos dominios.

Un modelo TE soporta fuertes controles sobre ejecución de programas y transiciones de dominio. Un programa, como cualquier otro objeto, esta asignado a un tipo, y la matriz de acceso TE especifica que tipos pueden ser ejecutados por cada dominio. Además, la matriz de acceso especifica que tipos pueden ser ejecutados para entrar inicialmente a un dominio. Por lo tanto, un dominio pude ser asociado con un programa de punto de entrada en particular y opcionalmente con un programa auxiliar en particular y/o librerías compartidas. Esta característica de TE es útil en asociaciones de permisos con un conjunto particular de código basado en sus funciones, confiabilidad y en la protección en contra de la ejecución de código malicioso.

El modelo TE SELinux difiere al tradicional modelo TE en que usa un solo atributo de tipo en el contexto de seguridad para procesos y objetos. Un dominio es simplemente un tipo que puede ser asociado con un proceso. Un solo tipo puede ser usado como dominio de un proceso y como el tipo de un objeto relacionado con un proceso. Una simple matriz de acceso especifica como tipos pueden acceder o interactuar con otros tipos en términos de los permisos definidos por la arquitectura Flask. Aunque la configuración TE de ejemplo frecuentemente usa el termino dominio cuando hace referencia al tipo de un proceso, el modelo TE SELinux no distingue internamente los dominios de los tipos.

El modelo TE SELinux también difiere del modelo TE tradicional en la forma en que usa la información de clase de seguridad proporcionada por la arquitectura Flask. Una transición TE SELinux o de decisión de acceso está basada en un par de tipos y en la clase de seguridad. Por lo tanto, la política puede tratar objetos que tienen el mismo tipo pero diferente clase de seguridad.

Una tercera diferencia entre el modelo TE SELinux y el modelo TE tradicional es que el modelo TE SELinux no tiene usuarios asociados directamente con dominios. En cambio, SELinux usa el modelo RBAC para proporcionar una capa adicional de abstracción entre usuarios y dominios.

Una regla de transición TE para un proceso especifica el nuevo dominio basado en el dominio actual del proceso y el tipo del programa. Una regla de transición para un objeto especifica el nuevo tipo basado en el dominio del proceso creador, el tipo del objeto relacionado, y la clase de seguridad de objeto. Si ninguna regla que corresponda es encontrada en la configuración TE, entonces el modelo TE SELinux proporciona una apropiada conducta por defecto para la clase de objeto. Para un proceso, el dominio del proceso es dejado igual a través de la ejecución del programa. Para un objeto, el tipo de objeto relacionado es usado para el nuevo objeto.

Una regla de vector de acceso especifica un vector de acceso basado en el par de tipo y clase de seguridad de objeto. Las reglas pueden ser especificadas para cada categoría de vector de acceso, incluyendo vectores *allowed*, *auditallow*, y *auditdeny*. Estas reglas de vector de acceso definen la matriz de acceso. Si ninguna regla que corresponda es encontrada en la configuración TE, entonces el modelo TE SELinux define por defecto una conducta apropiada para cada categoría de vector de acceso. Los permisos son denegados a

menos que exista una regla *allow* explícita. Los permisos no son auditados cuando se conceden, a menos que exista un regla *auditallow* explícita.

3.2.4.2 Modelo RBAC

Un modelo tradicional RBAC autoriza usuarios para actuar en ciertos roles, y le asigna un conjunto de permisos a cada rol. El modelo RBAC SELinux autoriza a cada usuario a un conjunto de roles, y autoriza a cada rol a un conjunto de dominios TE. Una relación de dominación de rol puede ser especificada opcionalmente en la configuración RBAC para definir una jerarquía entre roles. La asignación de permisos esta principalmente delegada a la configuración TE. Este metodología combina la fácil administración proporcionada por el modelo RBAC con la protección de granularidad fina que provee el modelo TE.

El modelo RBAC SELinux mantiene un atributo de rol en el contexto de seguridad para cada proceso. Para los objetos, el atributo de rol típicamente esta definida como el rol genérico object_r y no se utiliza. La configuración RBAC especifica transiciones autorizadas entre roles basándose en el par de roles. Sin embargo, también es deseable limitar las transiciones de rol a ciertos programas para asegurarse de que un código malicioso no pueda causar tales transiciones. Por lo tanto, las transiciones de rol típicamente están limitadas a ciertos dominios TE en la configuración de la política.

3.2.4.3 Modelo de Identidad de Usuario

Los atributos de identidad de usuarios son inconvenientes para el uso de SELinux. Los uids Linux son cambiados frecuentemente solo para expresar un cambio en los permisos o en los privilegios, lo opuesto a cambiar el usuario actual, presentando problemas para la auditoría de usuarios. Los uids Linux pueden ser cambiados en cualquier momento a través de un conjunto de llamadas **set*uid**, sin proporcionar control sobre el estado de herencia o la inicialización del proceso en la nueva identidad. Los uids Linux pueden ser cambiados arbitrariamente por los procesos superusario.

Más que imponer nuevas restricciones en los atributos de identidad de usuarios Linux, SELinux mantiene un atributo de identidad de usuario en el contexto de seguridad, que es independiente de los atributos de identidad de usuario de Linux. Al usar un atributo de identidad de usuario separado, el control de acceso obligatorio SELinux permanece ortogonal al control de acceso existente en Linux. SELinux puede aplicar controles rigurosos sobre los cambios a su atributo de identidad de usuario sin afectar la compatibilidad con las semánticas uid de Linux.

La configuración de política limita la habilidad para cambiar atributos de identidad de usuarios SELinux a ciertos dominios TE. Estos dominios están asociados con ciertos programas, tal como *login*, *crond* y *sshd*, que han sido modificados para llamar las nuevas funciones de librería para establecer la identidad de usuario SELinux apropiadamente. Por lo tanto, las sesiones de login de usuario y trabajos crond inicialmente están asociados con la identidad de usuario SELinux apropiada, pero cambios subsecuentes en el uid Linux pueden no ser reflejados en la identidad de usuario SELinux. En algunos casos, esto es deseable para proporcionar auditoria de usuario o para prevenir violaciones de seguridad.

Dado que la identidad de usuario SELinux es independiente del uid Linux, es posible mantener espacios de usuario separados para SELinux y Linux, con un mapeo apropiado realizado por los programas que asignan la identidad de usuario SELinux. Por

ejemplo, en vez de mantener una entrada separada para cada usuario Linux en la política SELinux, puede ser deseable mapear la mayoría de los usuarios Linux a un solo usuario SELinux sin privilegios. Este método es adecuado cuando no es necesario separar estos usuarios a través de la política SELinux.

4. PROPUESTA DE DESARROLLO

En SELinux, la configuración de la política de seguridad está almacenada en archivos de texto plano (como todo archivo de configuración en Linux) que, dependiendo del tamaño de la organización que los utilice, pueden alcanzar tamaños que imposibilitan realizar modificaciones sin conducir a error.

La propuesta desarrollada consiste en dos herramientas de configuración que permiten configurar la política de seguridad que será aplicada en SELinux: una para la administración de los usuarios SELinux y otra para la creación de nuevas definiciones TE y RBAC.

Se llegó a esta propuesta, luego de examinar los sitios relacionados con SELinux (que son pocos) en donde hay algún tipo de herramienta desarrollada, en el Anexo A se describen algunas de estas. Entre estos sitios, se encuentra uno que ha ido adquiriendo importancia para quienes están interesados en la evolución de SELinux: “SELinux Distribution Integration News” [SELDIN]. Aquí se puede apreciar que el área en la que hay mayor interés es la de herramientas para la configuración de la política de seguridad.

Al momento de iniciarse este proyecto, solo había un par de herramientas disponibles: una para la configuración de usuarios, y otro que sólo permite “analizar”” la política (en realidad, entregar algún tipo de resumen sobre las declaraciones que contiene). Ambas pertenecen a Tresis Technology [TRESIS]. Se analizó la herramienta para la configuración de usuarios (SEUSER), y se decidió realizar una con el mismo objetivo, pero más funcional

(SEUSER es complicado de utilizar, y la idea es *facilitar* la administración de usuarios). Además, estas herramientas requerían de la instalación de algunos componentes que no vienen incluidos en Red Hat Linux, por lo que complica aun más el poder utilizarlas. La otra herramienta (APOL) sólo analiza, y durante la investigación se hizo evidente la necesidad de una herramienta que permita crear los archivos de configuración a partir de los cuales se genera la política de seguridad (no existía ninguna que lo hiciera, o que ayudara un poco en esa labor).

Así, las herramientas desarrolladas en este proyecto, están pensadas de forma que sean "llegar y usar" (sólo utilizan componentes incluidos en RedHat Linux), y naturalmente tienen una orientación netamente administrativa (debe conocerse el funcionamiento de SELinux) y son capaces de modificar los archivos de configuración empleados por SELinux para construir la política de seguridad, facilitando así la labor del administrador de sistemas, quien no tendrá necesidad de ensuciarse las manos con los archivos de texto que expresan estas configuraciones. La administración de usuarios, roles, dominios, tipos, reglas y estricciones podrán ser realizadas con una fiabilidad mayor, debido a la eliminación del "error humano" en la edición de los archivos de configuración.

Dada la complejidad de algunas acciones que son necesarias para poder realizar algún tipo de configuración, las herramientas funcionan en el ambiente gráfico incluido en Red Hat Linux (KDE, Gnome), simplificando las labores de configuración, dada las características de facilidad de uso que ofrece un ambiente gráfico, en especial si recién se comienza a comprender las capacidades de SELinux.

A continuación se describen las principales características de las herramientas desarrolladas, y que han sido nombradas como: SELUM (acrónimo de **S**ecurity **E**nhanced **L**inux **U**ser **M**anager) y SELPC (acrónimo de **S**ecurity **E**nhanced **L**inux **P**olicy **C**onfigurator).

4.1 SELUM

El objetivo de esta aplicación es permitir la administración de los usuarios SELinux, sin necesidad de intervenir manualmente los archivos de configuración de la política de seguridad de SELinux, lo que implica:

- Agregar usuarios del sistema Linux a SELinux. Nótese que el usuario **debe** estar definido en Linux. Esta herramienta no crea usuarios Linux, solo autoriza a ciertos usuarios Linux a iniciar sesión en SELinux.

- Eliminar usuarios SELinux

- Modificar los roles y contextos de seguridad predeterminados para cada usuario.

La herramienta posee sólo una interfaz, por lo que permite visualizar toda la información sobre las propiedades del usuario de una sola mirada. Para más detalles sobre la descripción de la interfaz y su funcionamiento, consultar el "manual de usuario" de SELUM, en el Anexo C.

En la siguiente tabla se resumen los archivos que SELUM genera. Para más detalles sobre la ubicación del archivo o sintaxis de las sentencias utilizadas en el archivo consultar el Anexo B.

Tabla 2.

Archivo	**Descripción**
user	Los cambios aplicados al listado de usuarios SELinux se ven reflejados en este archivo (los roles a los que puede tener acceso un usuario).

cron_context	Los cambios aplicados al contexto para los trabajos de crond de cada usuario son reflejados aquí (el rol y dominio bajo el cual se ejecutarán estos trabajos).
default_context	Los cambios aplicados al contexto predeterminado de inicio de sesión de cada usuario son reflejados aquí (el rol y dominio bajo el cual iniciará sesión).
policy.conf	Este archivo se ve modificado sólo si se recompila la política de seguridad de usuarios, actualizando así la política de seguridad general SELinux (dando acceso a los usuarios especificados).

4.2 SELPC

El objetivo de esta aplicación es permitir la configuración de la política de seguridad de SELinux, para aplicaciones no contempladas en ella y/o definición de nuevos roles, sin necesidad de intervenir manualmente los archivos de configuración de SELinux, siendo necesaria la capacidad de crear :

- Nuevos roles

- Nuevos tipos

- Nuevas reglas de seguridad, para relacionar estos nuevos tipos/roles.

- Nuevos archivos, en donde se almacenarán estas definiciones, y que deberán ser tomadas en cuenta por SELinux sin intervención manual del usuario, o sea, evitar la necesidad de editar archivos en distintos lugares (siguiendo el modelo de seguridad de ejemplo).

SELPC no elimina la necesidad de compilar y ajustar la política de seguridad manualmente, debido a que la modificación de la política de seguridad es un proceso delicado, y en el que se debe tener experiencia en el uso que se le puede dar a las capacidades de seguridad que ofrece SELinux, además de conocer muy bien el funcionamiento de la aplicación que se quiere "confinar" por medio de la nueva política de seguridad que se desea construir. SELPC solo elimina el inconveniente de crear varios archivos en distintas ubicaciones (los que son descritos más adelante) y de escribir las variadas y extensas instrucciones necesarias para lograr que un tipo o rol sea utilizable dentro de la política de seguridad.

SELPC consta de una sola interfaz, en donde las labores de creación se dividen en dos:

1) instrucciones RBAC: creación de nuevos roles y/o reglas de interacción entre roles existentes y el nuevo rol creado.

2) instrucciones TE: creación de nuevos tipos, reglas de seguridad, macros y contextos de archivo.

En la siguiente tabla se resumen la ubicación y finalidad de los archivos que SELPC genera. La sintaxis de las sentencias que crea SELPC están descritas en detalle en [SmalleyConfiguring2002] y de forma resumida en [StatementsTresis2002]. Los archivos de entrada que procesa SELPC para poder interpretar la política de seguridad existente son descritos en el Anexo B.

Tabla 3.

Archivo	**Descripción**
*.te	Este tipo de archivos contienen las declaraciones de los nuevos tipos, reglas de seguridad (transición de tipo, cambio de tipo, vector de acceso y aserción de vector de acceso), macros y los comentarios, relacionados a través de SELPC. Siguiendo el modelo de seguridad de ejemplo incluido con SELinux, estos archivos son generados en /etc/security/selinux/src/policy/domains/program.
*.fc	Este tipo de archivo contiene las declaraciones de contexto de archivo que se le asigna a ciertos archivos o directorios por medio de SELPC, los que son utilizados por la utilidad de SELinux **setfiles**. Siguiendo el modelo de seguridad de ejemplo incluido con SELinux, estos archivos son generados en /etc/security/selinux/src/policy/file_contexts/program.
rbac	Este archivo es una centralización de la declaración de las relaciones entre los roles. Este archivo puede ser modificado por SELPC, aunque se puede generar una copia independiente, para así examinarla con mas detalle y luego agregarla al definitivo (o reemplazarlo). En caso de escribir directamente a este archivo, se sigue el modelo de seguridad de ejemplo incluido en SELinux,

	modificando el archivo ubicado en /etc/security/selinux/src/policy

4.3 Diseño

Esta sección tiene la intención de exponer lo que fue necesario hacer y utilizar para llevar a buen termino el proyecto, además de lo que es necesario estudiar para lograr comprender a cabalidad la terminología SELinux.

4.3.1 Herramientas utilizadas

El proyecto fue desarrollado utilizando Red Hat Linux en su versión 7.1, con el kernel selinux versión 2.4.19. Existen varios documentos que describen diversos aspectos de SELinux. Los temas principales que es necesario estudiar son:

Funcionamiento interno del kernel, ver [SpencerUsenixSec1999] y [LoscoccoOLS2001].
Implementación actual de SELinux, ver [SmalleyModuleTR2001].
Política de seguridad de ejemplo incluida con SELinux, ver [SmalleyNAITR2001].
Configuración de la política de seguridad incluida con SELinux, ver [SmalleyConfiguring2002].
Ubicación, finalidad y sintaxis de los archivos de configuración, ver [StructureTresis2002] y [StatementsTresis2002]

Dada la naturaleza modular y de código abierto que caracterizan a Linux, se utilizaron varias herramientas, del tipo GNU y GPL, para la construcción de SELUM y SELPC.

Para la construcción de las interfaces se utilizó **Glade** con **GTK+**, que vienen incluidos en Red Hat Linux.

Glade permite diseñar la interfaz gráfica de una aplicación de forma visual. Esto conlleva un enorme ahorro de tiempo y trabajo, pues evita tener que codificarlo manualmente. Además separa el código del GUI del código de la aplicación. Para esto, Glade guarda el diseño de la interfaz gráfica en un archivo XML.

GTK+ es una biblioteca escrita en C, que aglutina a un conjunto de widgets orientados hacia la programación de aplicaciones gráficas en X Window. Está altamente orientada a objetos y se acopla con los lenguajes más populares, como C++, Objective C, Perl, TOM, Guile, Python, etc. GTK+ además usa GLib, que es una biblioteca en C muy útil, incluye ayudas para portar los programas a diferentes arquitecturas y contenedores como listas enlazadas o tablas de claves

Para la edición del código fuente que manipula la interacción de las señales de la interfaz gráfica, se utilizó el IDE **Anjuta** versión 0.4.

Para la programación de las funciones de interpretación de la política se utilizó **C**, que también viene incluido con Red Hat Linux. En realidad se programó utilizando el editor **Turbo C++** versión 1.01 de Borland (para DOS), dado que posee un excelente editor y depurador, pero utilizando C estándar para asegurar la compatibilidad con Linux.

4.3.2 Organización del código fuente

El código fuente está organizado de forma que las funciones que manejan la interfaz gráfica son independientes de las funciones que procesan la política. Así se pudo reutilizar el código desarrollado para SELUM en SELPC.

4.3.2.1 Código fuente para el procesamiento de la política

Este código esta dividido en varios archivos, con el fin de agrupar las funciones de un mismo tipo (funciones íntimamente relacionadas) y también de facilitar el proceso de depuración, debido a la complejidad de las aplicaciones. Los archivos en los que se han dividido las diversas funciones se listan a continuación:

Tabla 4.

Archivo	Descripción
funcs.c	Funciones generales utilizadas a lo largo de todo el programa.
functype.c	Funciones para el procesamiento de las declaraciones de tipos SELinux.
funcuser.c	Funciones para el procesamiento de las declaraciones de usuarios SELinux.

funcrols.c	Funciones para el procesamiento de las declaraciones de rol SELinux.
funcattr.c	Funciones para el procesamiento de las declaraciones de atributos SELinux. Las funciones que contiene este archivo solo son utilizadas por SELPC.
funcclas.c	Funciones para el procesamiento de las declaraciones de clases de seguridad SELinux. Las funciones que contiene este archivo solo son utilizadas por SELPC.
funccrea.c	Reúne todo el código para la generación de los nuevos archivos. Las funciones que contiene este archivo solo son utilizadas por SELPC.
selum.h	Definición de las constantes y estructuras de datos utilizadas en las funciones declaradas en los archivos anteriormente descritos. La versión de SELPC es distinta a la de SELUM, por el hecho de que procesa una mayor cantidad de instrucciones de la política de seguridad.
protos.h	Declaración de los prototipos de TODAS las funciones. Es una centralización de los prototipos, dado que, a pesar de estar en distintos archivos, las funciones para procesar la política son muy dependientes entre si.

selum.c	Punto de partida. Solo tiene instrucciones include. Este es el archivo que es necesario incluir en las aplicaciones que quieran utilizar las funciones para el procesamiento de la política.

En general, las funciones manejan listas enlazadas simples. Esto, por la necesidad de maximizar el espacio de memoria para los datos. En la versión del kernel SELinux que se utilizó a lo largo de todo el proyecto, se declaraban 827 tipos, cada uno de los cuales posee ciertos atributos de seguridad. También están los usuarios y sus atributos, mas los roles y otras entidades que es necesario tener en memoria, especialmente para SELPC. Aquí se describe el proceso de lectura de información, de modo de comprender como utilizar las funciones para el procesamiento de la política de seguridad:

Este proceso involucra el llamar a cada una de las funciones de lectura de información de los archivos y almacenar el puntero que retornan. El nombre y ubicación de estos archivos están definidos como constantes en selum.h, para facilitar la adaptación a posteriores versiones de SELinux.

PASO 1: Llevar la información sobre los atributos a memoria, por medio de la función LeeSELAttribs.

PASO 2: Llevar la información sobre las clases a memoria, por medio de la función LeeSELclases.

PASO 3: Llevar la información sobre los tipos a memoria, por medio de la función LeeSELtypes.

PASO 4: Llevar la información sobre los roles a memoria, por medio de la función LeeSELroles.

PASO 5: Llevar la información sobre los usuarios a memoria, por medio de la función LeeSELusers.

Los punteros a las listas obtenidos son almacenados en una estructura declarada con el único fin de tener todos los punteros en una sola "variable" (plSELdatos), para disminuir la cantidad de parámetros que necesiten las funciones. Esto, porque algunas funciones (como LeeSELtypes) utilizan otras listas (en este caso, la de atributos), lo que implica que los pasos deben ser realizados necesariamente en el orden aquí señalado (no es posible leer los tipos declarados sin conocer los atributos válidos que pueden poseer).

Durante el funcionamiento de la aplicación que manipule estos datos, se agregarán, eliminarán y modificarán elementos de estas listas.

Ahora, para generar los archivos existen dos funciones. Una es utilizada en SELUM y la otra en SELPC. Estas funciones realizan procesos muy similares, pero en distintos archivos y estructuras. En SELUM se generan 3 archivos (el de la información de usuarios, cron y login), mientras que en SELPC se crean de 1 a 3 archivos (declaración de tipos, roles y contextos de archivo). Lo único que realizan estas funciones es recorrer las listas pertinentes y volcarlas a los archivos pertinentes.

4.3.2.2 Código fuente para el procesamiento de las interfaces

Glade al generar automáticamente código fuente de la aplicación, permite que la programación de las interfaces se facilite enormemente y que la integración de las

funciones que manejan la política se realice fácilmente, dado que son 100% independientes de la aplicación que las utiliza (ya sea en una aplicación grafica o de consola).

El modelo de objetos de GTK+ es un marco de trabajo orientado a objetos para un lenguaje no orientado a objetos como lo es el lenguaje de programación C. Incluye una singular herencia de objetos, métodos virtuales, señales, modificación de objetos en tiempo de ejecución y comprobación de tipos en tiempo de ejecución. Los objetos en GTK+ son simplemente estructuras en C y la herencia se logra incluyendo la estructura padre como primer elemento en la estructura hija.

GTK+ es una biblioteca basada en componentes y en contenedores: la ubicación de un componente siempre es dentro de un contenedor, que podrá alojar a este o a más componentes. El uso de esta biblioteca en un principio resulta un tanto confusa, ya que su API es muy extenso. Se ha usado un subconjunto de funciones básicas para construir el sistema, y las algunas más específicas que manejan al objeto en concreto (listas, combo, botones, etc.). El trabajo con la biblioteca GTK+ ha sido complicado en general, dada la nula experiencia previa. Algunos componentes son especialmente complicados de manejar, como por ejemplo el “widget”“ lista, y se necesita un tiempo considerable para adaptarse a su uso.

Para construir la interfaz de SELUM y SELPC se utilizan widget que permiten mostrar los tipos, roles y usuarios definidos por la política de seguridad. Otros widget, como contenedores son utilizados y declarados en la aplicación pero GLADE se encarga de generar el código fuente donde se declaran y define sus atributos.

Para la programación de las señales, se deben declarar y ubicar los punteros que manejan los widget que las emiten mediante el uso de la función gtk_lookup_widget en cada una de

las funciones para el procesamiento de señales, dado que al entrar a una de estas funciones solo se conoce el puntero del objeto que emite la señal. Por lo tanto para facilitar el manejo de los widget estos se declaran globales para que siempre estén disponibles para su uso, y así evitar el tener que declararlos en más de una oportunidad. Este proceso se realiza en el main.c, y son declarados como variables externas en callback.c para la programación de las señales.

A continuación se describen los widget utilizados tanto en SELUM, como en SELPC.

Tabla 5.

Widget	**Descripción**
GtkCList	Lista multi-columna, la que permite un mejor manejo de los elementos que se desea listar. En el caso de SELUM y SELPC se utilizaron listas con una columna.
GtkButton	Este widget crea una señales cuando se presiona.
GtkCombo	Widget compuesto por un campo de entrada de texto y de una lista desplegable. Utilizado para mostrar los tipos, que se deben seleccionar para configurar sentencias RBAC, TE y de usuario.
GtkCheckButton	Se utilizaron botones checkbox para definir instrucciones opcionales en la política, tanto de RBAC,TE y de usuarios.

GtkNotebook	Gracias a este componente no fue necesario utilizar más interfaces que pudieran realizar la tarea de administración de usuarios y configuración de política utilizar
GtkLabel	Debido a la complejidad de utilizar una barra de estado, se utilizo etiqueta para mostrar mensajes de alerta o de información al usuario de SELUM y SELPC, así se facilita la programación.

Del código generado por Glade, solo dos archivos son modificados manualmente, para programar las funciones que son invocadas al producirse eventos y señales en la interfaz, para SELUM se modificaron el archivo main.c y callbacks.c. Para SELPC además de modificar estos archivos, se crearon dos más que permiten un mejor manejo de la interfaz, ya que muchas de estas funciones son requeridas más de una vez, estos archivos son funcrbac.c y functe.c, descritos a continuación.

Tabla 6.

Archivo	**Descripción**
main.c	Glade genera este archivo la primera vez que se construye el código a partir de la interfaz, luego no lo vuelve a tocar más. Posee algunas declaraciones de “widgets” e inicia el ciclo de procesamiento de señales de la interfaz.
callbacks.c	Glade genera este archivo la primera vez que se construye el código a partir de la interfaz, luego solo agrega las nuevas declaraciones de

	funciones para el procesamiento de señales. Este solo contiene funciones para el procesamiento de señales, y aquí es donde se define como interactúan los diversos elementos de la interfaz, además de utilizar las funciones de procesamiento de política.
funcrbac.c	Funciones genéricas para el manejo de las listas que manipulan las declaraciones de los roles. Además se incluyen funciones que permiten la limpieza sobre los widget para configurar RBAC.
functe.c	Funciones genéricas para el manejo de las listas que manipulan las declaraciones de los tipos, reglas, macros y combos que manipulan tipos. Además se incluyen funciones que permiten la limpieza sobre los widget para configurar TE.

4.4 Interfaces de SELUM y SELPC

SELUM y SELPC cuentan con una sola interfaz cada uno, lo que permite un acceso más inmediato a la información que manipulan. Para una completa descripción de cada interfaz, asi como la forma de utilizarlas, remitirse al Anexo C.

Figura 1.- Interfaz de SELUM

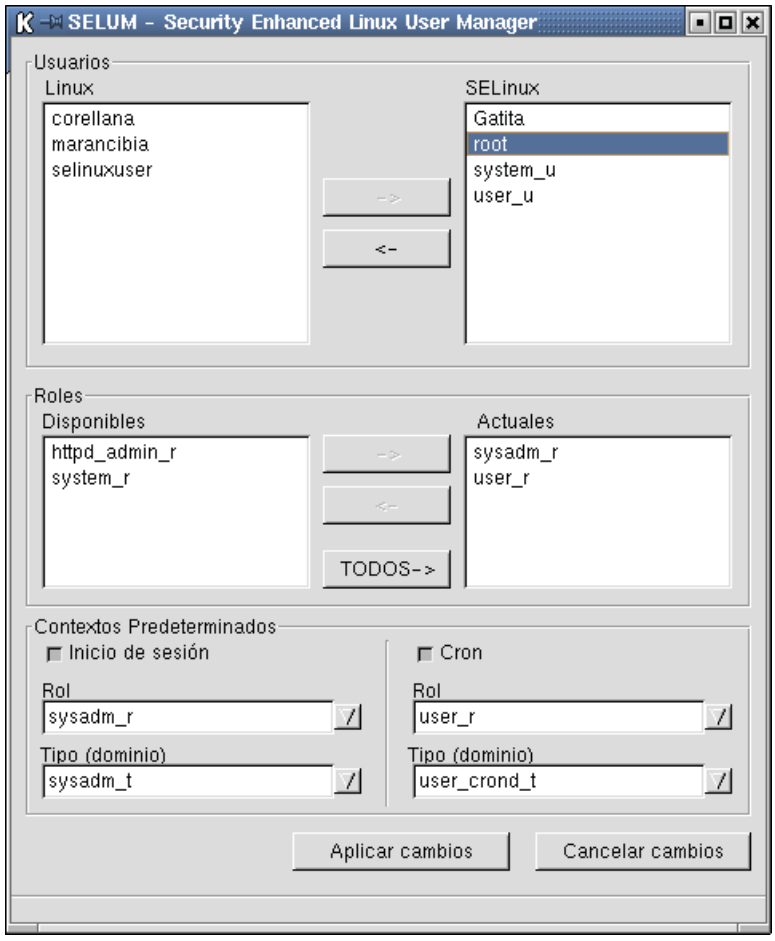


Figura 2.- Interfaz de SELPC para la definición de reglas TE

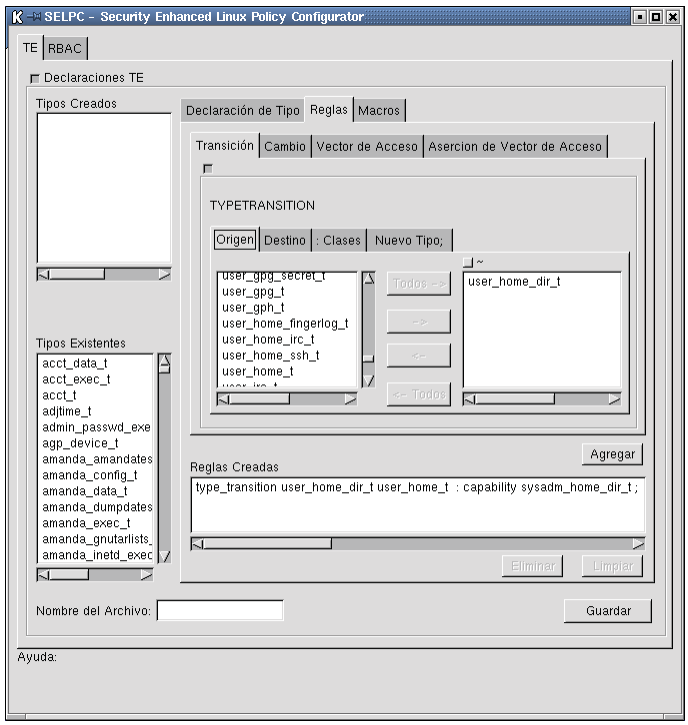
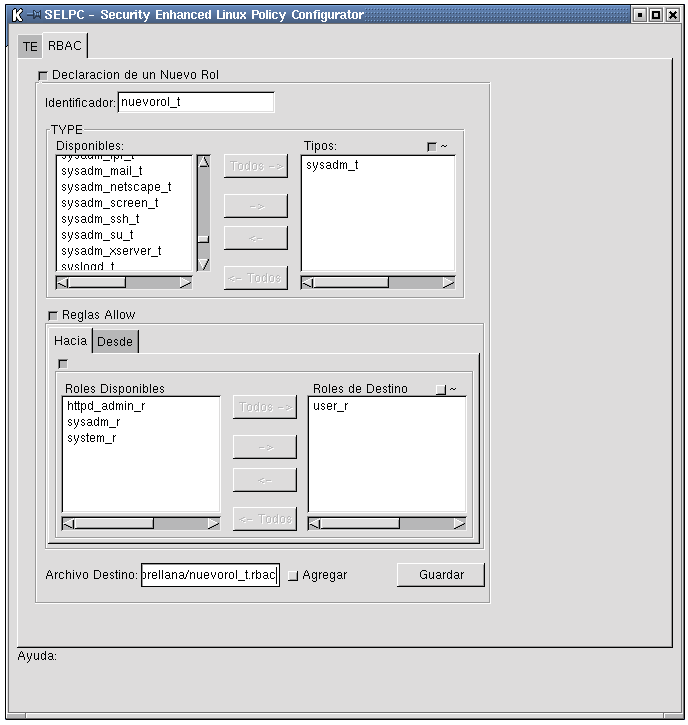


Figura 3.- Interfaz de SELPC para la definición de roles



5. CONCLUSIONES

El proyecto SELinux al estar en constante evolución, resultó ser una investigación muy interesante, de actualidad y que realmente es necesaria, porque aplicar la seguridad desde el núcleo mismo del sistema operativo, nos permite asegurar la integridad y disponibilidad de la información que esta protegiendo el sistema operativo. Este proyecto llama mucho la atención por sus "líder", la NSA. Nos sorprendió también por sus altos y bajos en el avance que demostraba… casi nada cuando iniciamos el trabajo, que luego incrementó a niveles que nos permitieron obtener gran cantidad de información que de otro modo hubiera sido necesario "urgar" en el código fuente. Este ritmo se ha mantenido hasta el fin de nuestro proyecto, y da señales de aumentar mucho mas, dado que la comunidad de desarrolladores Linux interesados por el proyecto son más y más.

Lograr flexibilizar la definición de una política de seguridad, así como su aplicación en el sistema que la utilizará, es lo más destacable de SELinux. Su "punto débil" está en la forma de definir la política de seguridad, que es muy compleja, extensa y difícil de mantener y aprender. Es por esto que al lograr mejorar la usabilidad de SELinux, también ser logra reducir el tiempo de implementación de una política de seguridad, y su mantención. Esta "mejora" en la usabilidad, gracias a las herramientas desarrolladas, fue lo que más satisfecho nos ha dejado. Sobretodo, SELPC, herramienta que fue desarrollada desde cero (no hay otra igual ni similar). Esto no significa que SELUM nos desilusionara, nada de eso. SELUM realmente cumple con su objetivo, y asi nos olvidamos de tener que editar archivos de texto… la meta para las herramientas de este proyecto.

Lamentablemente, la falta de estándares para el desarrollo de interfaces gráficas en Linux, y la complejidad de algunas de las que actualmente se están "imponiendo", obliga a invertir una buena cantidad de tiempo, solo para decidirse por una, selección que también nos ha

dejado satisfechos, por demostrar la mayor simplicidad de uso y uniformidad con los ambientes gráficos de Linux.

REFERENCIAS

Los documentos que representan estas referencias aquí listadas, pueden ser descargados desde el sitio web de la NSA para SELinux: http://www.nsa.gov/selinux. Se exceptúan dos documentos, que son resúmenes realizados por Tresis Corp: http://www.tresis.com/selinux.

[LoscoccoFreenix2001]. Integrating Flexible Support for Security Policies into the Linux Operating System. (Integrando soporte flexible de políticas de seguridad en el sistema operativo Linux). Peter Loscocco, NSA; pal@epoch.ncsc.mil, Stephen Smalley; NAI Labs, sds@tislabs.com; Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference; The USENIX Association, Junio 2001.

[LoscoccoNSATR2001]. Integrating flexible support for security policies into the Linux Operating System. (Integrando soporte flexible de políticas de seguridad en el sistema operativo Linux). Peter Loscocco, NSA, pal@epoch.ncsc.mil; Stephen Smalley, NAI Labs, sds@tislabs.com. Versión “técnica”, Febrero del 2001

[SmalleyConfiguring2002]. Configuring the SELinux Policy. (Configurando la política de SELinux). Stephen Smalley, NAI Labs, ssmalley@nai.com. Versión liberada el 2 de Mayo del 2002

[SmalleyModuleTR2001]. Implementing SELinux as a Linux Security Module. (Implementando SELinux como un Módulo de Seguridad Linux). Stephen Smalley, NAI Labs, ssmalley@nai.com; Chris Vance, NAI Labs, cvance@nai.com; Wayne Salamon, NAI Labs, wsalamon@nai.com. Versión del 2 de Mayo del 2002

[SmalleyNAITR2001]. A Security Policy Configuration for the Security Enhanced Linux. (Una configuración de política de seguridad para el Linux de Seguridad Mejorada). Stephen

Smalley; Timothy Fraser. Versión del 18 de Diciembre del 2000, NAI Labs, slinux@tislabs.com

[SpencerUsenixSec1999]. The Flask Security Architecture: System Support for Diverse Security Policies. (La arquitectura de seguridad Flask: Soporte en el sistema para diversas políticas de seguridad). Ray Spencer, SCC (Security Computing Corporation); Stephen Smalley, NSA; Peter Loscocco, NSA; Mike Hibler, Universidad de Utah; David Andersen, Universidad de UTA; Jay Lepreau, Universidad de UTA; Proceedings of the Eighth USENIX Security Symposium. The USENIX association, August 1999, http://www.cs.utah.edu/flux/flask/

[StatementsTresis2002]. Overview of Security-Enhanced Linux Type Enforcement and Role-Based Access Control Statements. (Resumen de las instrucciones de control de aplicación de tipo y control de acceso basado en roles de Security-Enhanced Linux). Tresys Technology, Versión 0.2, del 8 de Abril de 2002, http://www.tresis.com/selinux

[StructureTresis2002]. Overview of Security-Enhanced Linux Policy File Structure. (Resumen de la estructura de archivos de política de Security-Enhanced Linux). Tresys Technology, Versión 0.2, del 8 de Abril de 2002, http://www.tresis.com/selinux

Las siguientes referencias complementan la investigación y abarcan diversos temas.

[APACHE]. Confining the Apache Web Server with Security-Enhanced Linux. (Confinando el servidor web Apache con Security-Enhanced Linux). Mitre; MichelleJ.Gosselin,mgoss@mitre.org; Jennifer Schommer, jschommer@mitre.org; http://www.mitre.org/about/ciis/selinux/index.html

[GLADE]. GLADE GTK+ User Interfaz Builder, http://glade.gnome.org/

[GTK]. Guía Avanzada de Desarrollo de aplicaciones Linux con GTK+ y GDK. Eric Harlow, Prentice Hall Iberia, Madrid, 1999

[LoscoccoPaper2000]. Integrating flexible support for security policies into the Linux Operating System. (Integrando soporte flexible de políticas de seguridad en el sistema operativo Linux). Peter Loscocco, NSA, loscocco@tycho.nsa.gov; Stephen Smalley, NAI Labs, ssmalley@nai.com, Versión “paper”, del 18 de Diciembre del 2000

[ORIBIUM]. SELinux Firewall. (Muro de Fuego SELinux). Oribium Labs, (c) 2001 by Reino Wallin

[SELDIN]. SELinux Distribution Integration News, http://selinux.sourceforge.net/

[TRESIS]. SELinux Distribution Integration News, http://www.tresis.com/selinux

ANEXOS

ANEXO A Herramientas Disponibles para SELinux

Al momento de iniciarse este proyecto existían muy pocas herramientas que permitían configurar SE Linux o algunos de sus aspectos mas importantes, aspectos que se pueden dividir en las siguientes categorías:

Políticas
GUI
Kernel
Trabajo en red
Documentación

1. Políticas

Tresys Technology

Tresys Technolgy ha desarrollado un par de herramientas basadas en GUI, denominadas *apol* y *seuser*. *Apol* es una herramienta de análisis de la política de seguridad, mientras que *seuser* permite administrar usuarios (agregar, modificar, etc).

Figura 4.- Interfaz de APOL

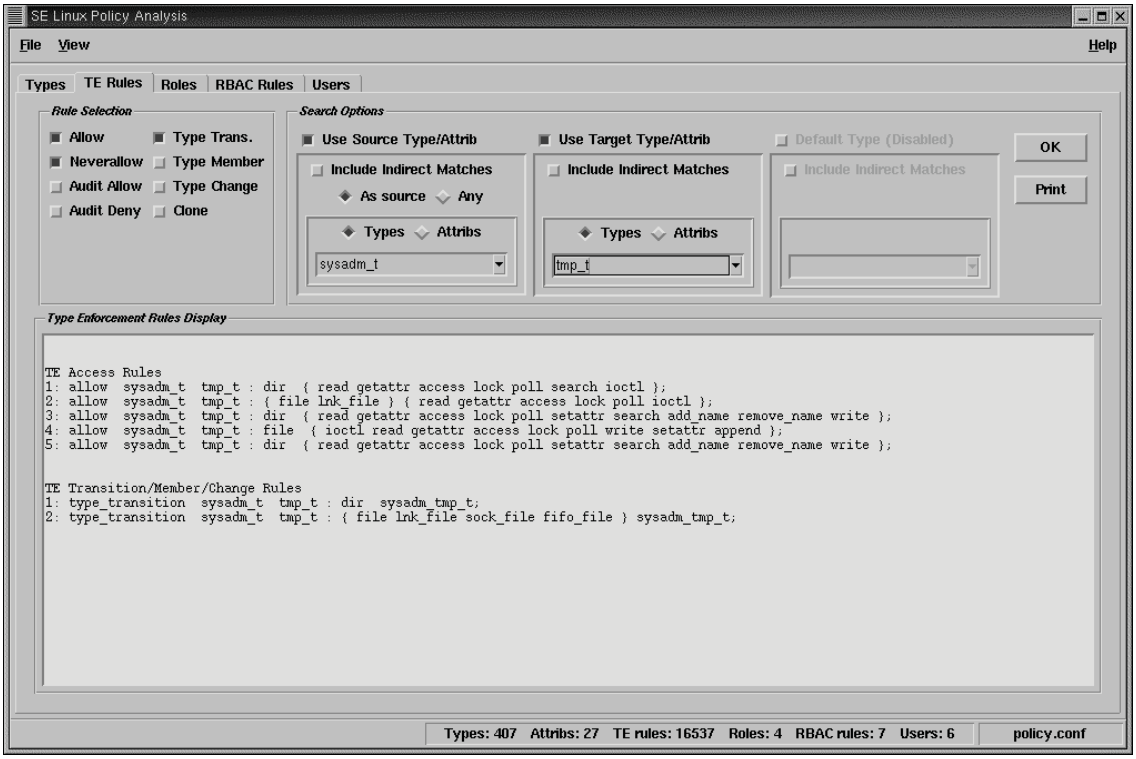
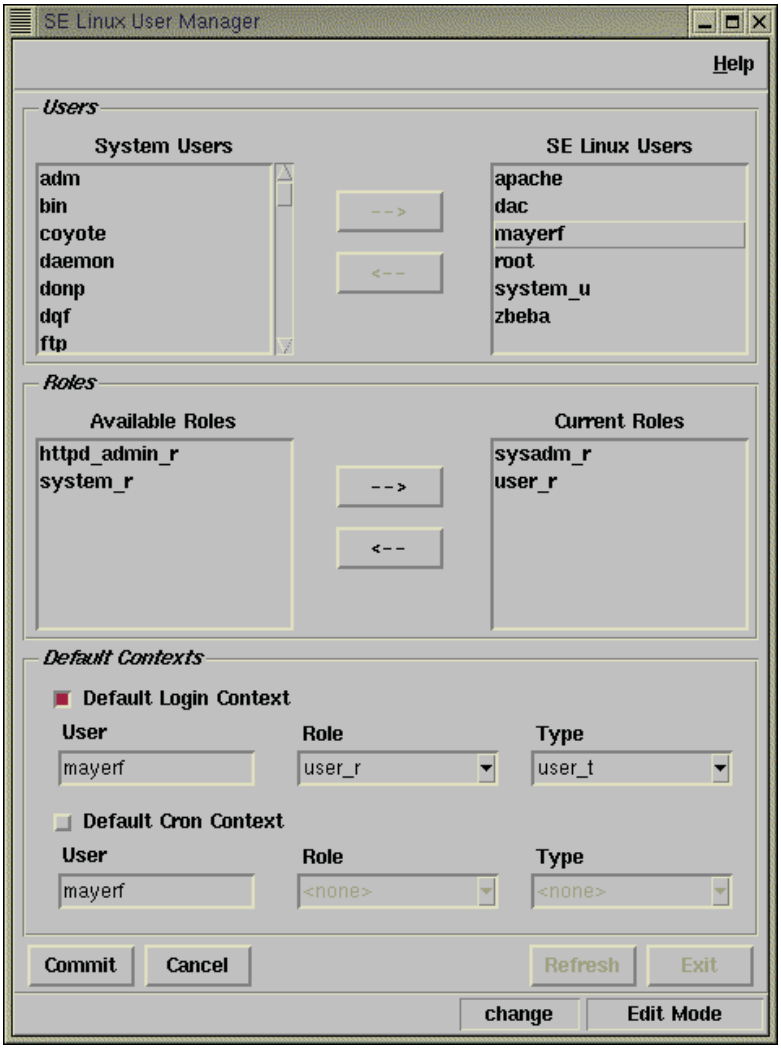


Figura 5.- Interfaz de SEUSER



Oribium Labs

Oribium labs ha desarrollado un conjunto de scripts para la implementación de una capa de aplicación firewall.

2. GUI

GDM: Existe un parche que permite la utilización de un GDM (lo que no es funcionalmente posible con la instalación inicial de SELinux). Mark Westerman ha

puesto a disposición este parche, que se puede descargar desde http://selinux.sourceforge.net/devel/gui.php3.

3. Trabajo en red

SIDs en los paquetes de red

James Morris tiene un parche que permite colocar SID en las cabeceras de los paquetes.

IPSEC

Mark Westerman ha contribuido a la distribución de la NSA con los componentes para permitir FreeS/WAN IPSEC.

ANEXO B Descripción de los Archivos Utilizados

Los datos de entrada de las herramientas son los siguientes archivos de configuración de SELinux:

user: La ubicación predeterminada de este archivo es en el directorio "*/etc/security/selinux/src/policy*". Este archivo contiene la declaración de los usuarios admitidos en SELinux. Si un usuario Linux no esta declarado en este archivo, no podrá ingresar al sistema.

Sintaxis de las declaraciones contenidas en este archivo:

user <*usuario*> roles <*rol*> | roles { <*rol1*> <*rol2*> ... };

cron_context: La ubicación predeterminada de este archivo es en el directorio "*/etc/security*". Este archivo contiene la declaración del contexto de seguridad para trabajos cron de cada usuario. Las declaraciones contenidas en este archivo modifican la conducta del programa **crond**.

Sintaxis de las declaraciones contenidas en este archivo:

<usuario>:*<rol>*:*<dominio>*

default_context: La ubicación predeterminada de este archivo es dentro del directorio "*/etc/security*". Este archivo contiene la declaración del contexto de seguridad predeterminado de inicio de sesión para cada usuario. Este archivo es utilizado por los programas **login** y **sshd**.

Sintaxis de las declaraciones contenidas en este archivo:

<usuario>:*<rol>*:*<dominio>*

policy.conf: La ubicación predeterminada de este archivo es en el directorio "*/etc/security/selinux/src"*. Este archivo contiene la política de seguridad compilada en su último estado "*humanamente legible"*. De este archivo se extraen los tipos, dominios y roles válidos para ser utilizados en SELUM y SELPC.

Sintaxis de las declaraciones contenidas en este archivo, y que son utilizadas por SELUM y SELPC (existen otras, pero no se toman en cuenta):

type <*tipo*> [alias <*alias*> | alias {<*alias1*> <*alias2*> …}][, *atributo1, atributo2,* …];

role <*rol*> types <*tipo1*> [, <*tipo2*>, …];

attrib.te: La ubicación predeterminada de este archivo es en el directorio "*/etc/security/selinux/src/policy"*. Este archivo contiene la declaración de los atributos válidos para ser utilizados en declaraciones de tipo, siendo utilizado mayormente por SELPC.

Sintaxis de las declaraciones contenidas en este archivo:

attribute <*atributo*>;

security_clases: La ubicación predeterminada de este archivo es en el directorio *"usr/local/selinux/Flask/"*. Este archivo contiene las declaraciones de clases de seguridad válidas para ser utilizadas en las declaraciones de reglas.

Sintaxis de las declaraciones contenidas en este archivo:

class <*clase*>

Además de los archivos utilizados por SELinux en la definición de la política de seguridad, se utiliza el siguiente archivo Linux como dato de entrada:

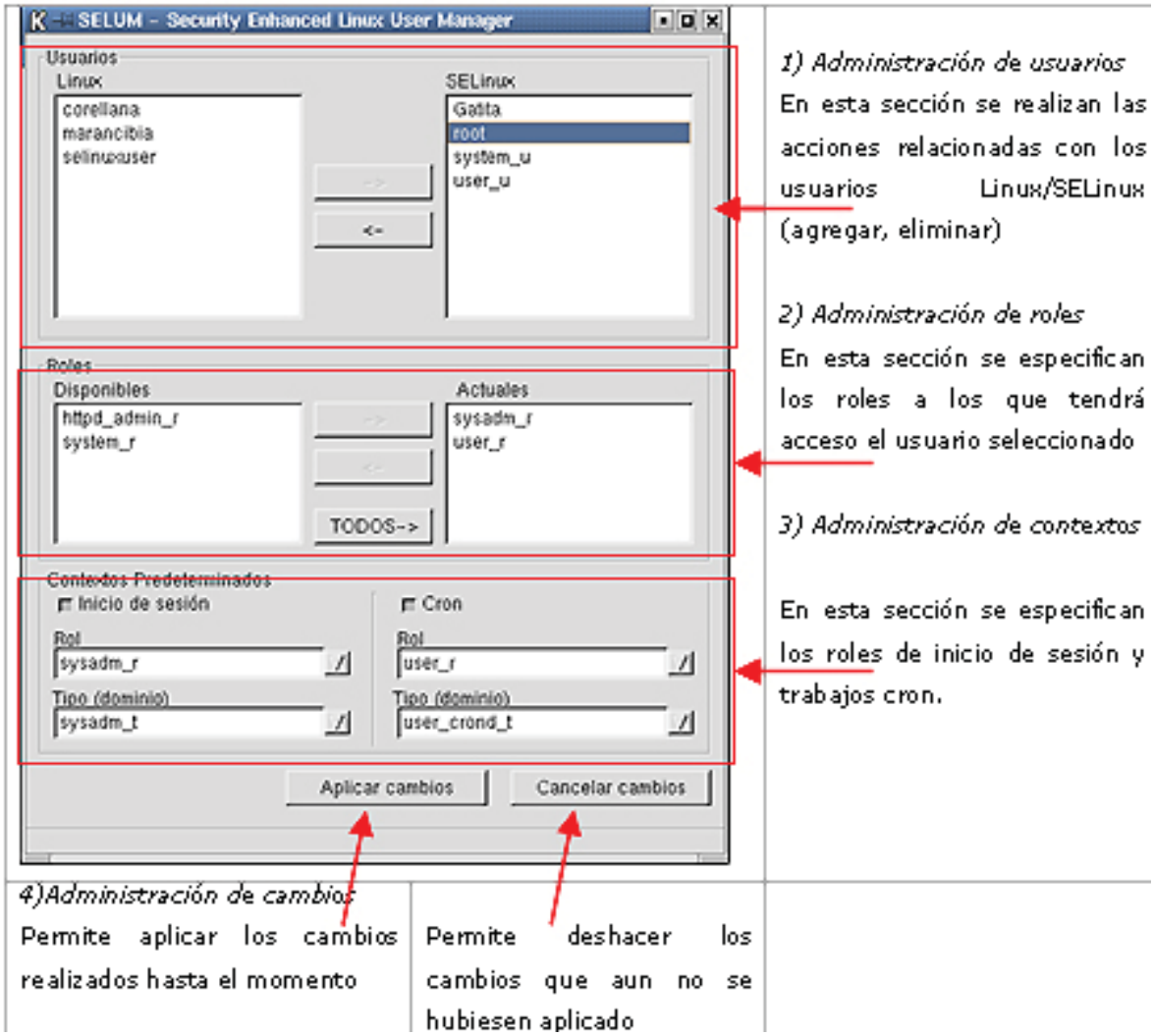
passwd: La ubicación predeterminada de este archivo es en el directorio *"/etc"*, del cual se extraen los nombres de los usuarios Linux.

ANEXO C Manual de Usuario

SELUM

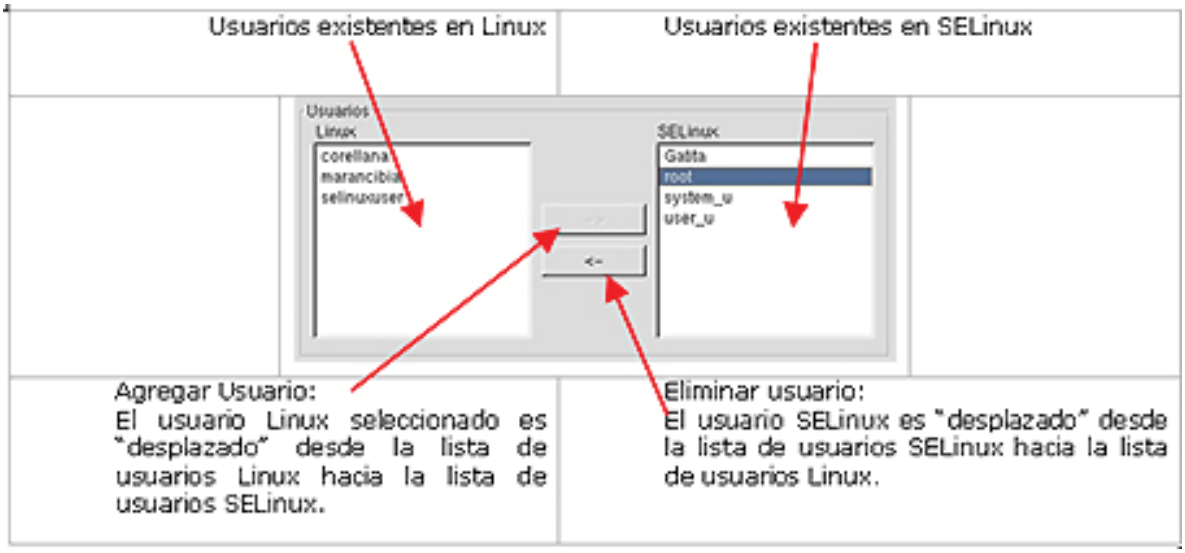
SELUM consta de una sola interfaz, dividida en 4 zonas, que se resumen en la siguiente figura:

Figura 6.- Interfaz de SELUM, resumida



SECCIÓN DE ADMINISTRACIÓN DE USUARIOS

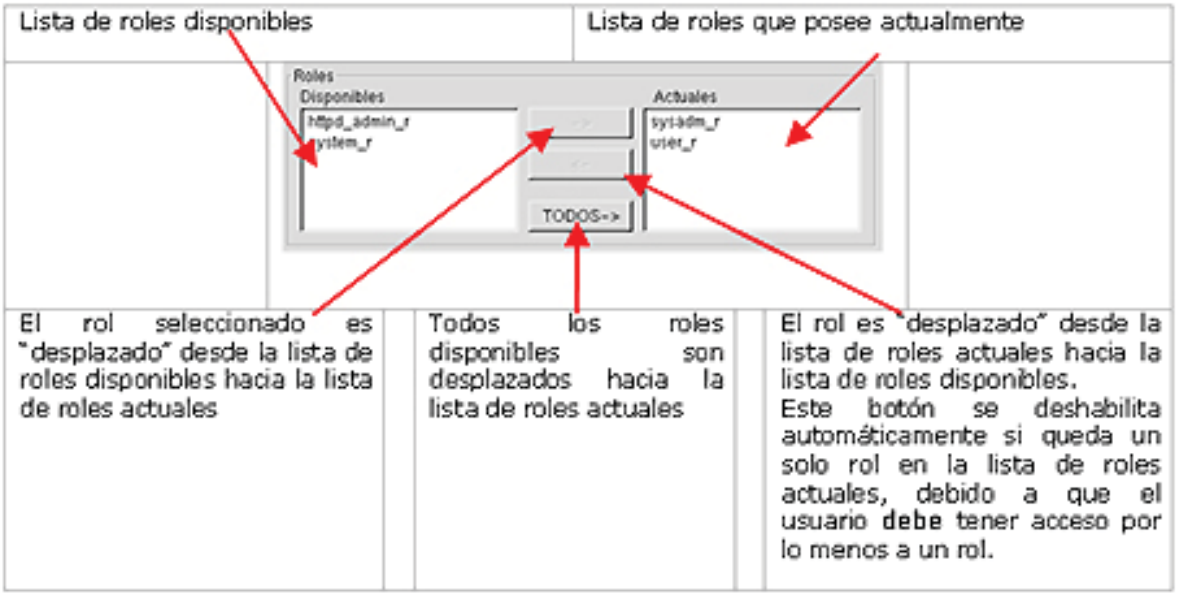
Figura 7.- Sección de adm. de usuarios en SELUM



Si se selecciona un usuario de Linux, se habilita el botón (agregar). Si se selecciona un usuario SELinux se habilita el botón eliminar, el frame "Roles" y el frame "Contextos Predeterminados", ambos mostrando la información correspondiente sobre como está configurado actualmente ese usuario, por lo que esa información se puede modificar (se pueden agregar o eliminar roles y definir los contextos predeterminados).

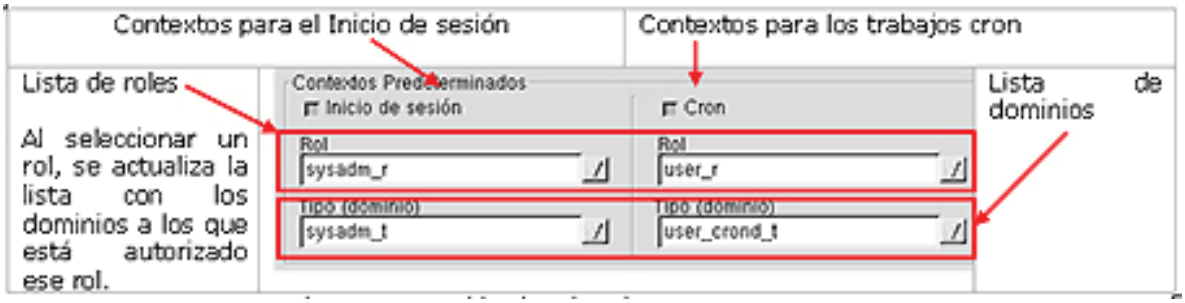
SECCION DE ADMINISTRACIÓN DE ROLES

Figura 8.- Sección de adm. de roles en SELUM



SECCIÓN DE ADMINISTRACIÓN DE CONTEXTOS

Figura 9.- Sección de adm. de contextos en SELUM



SECCIÓN DE ADMINISTRACIÓN DE CAMBIOS

8. Aplicar los cambios:

Si se presiona el botón “Aplicar Cambios”, los cambios realizados son guardados en los archivos correspondientes (**user, default_context** y **cron_context**), y se puede continuar con el procesamiento de los usuarios.

9. Cancelar los cambios:

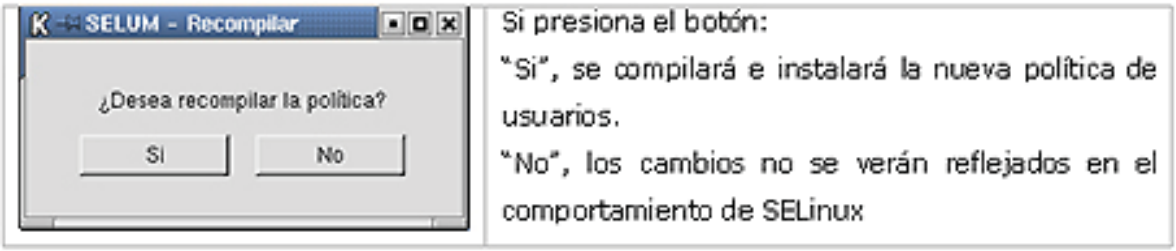
Si se presiona el botón “Cancelar Cambios” se cancelan todos los cambios que no han sido aplicados a los archivos de configuración.

10. Cerrar la aplicación:

Para cerrar la aplicación se debe presionar el botón de la barra de título. En ese momento, por medio de un cuadro de diálogo, se le pregunta al usuario si desea compilar la política.

11. Cuadro de Diálogo “Recompilar”:

Figura 10.- Ventana para la confirmación de recompilación de la política



Además, la aplicación cuenta con una barra de estado que permite visualizar si existen cambios pendientes, o los errores que vaya cometiendo.

SELPC

SELPC cuenta con una sola interfaz que permite la creación de tipos, roles, reglas y macros SELinux (Figura 2)

Esta interfaz cuenta con dos pestañas :

"TE" en donde se pueden crear declaraciones TE.
"RBAC" en donde se pueden crear declaraciones RBAC.

Figura 11.- Interfaz de SELPC – Pestaña TE.

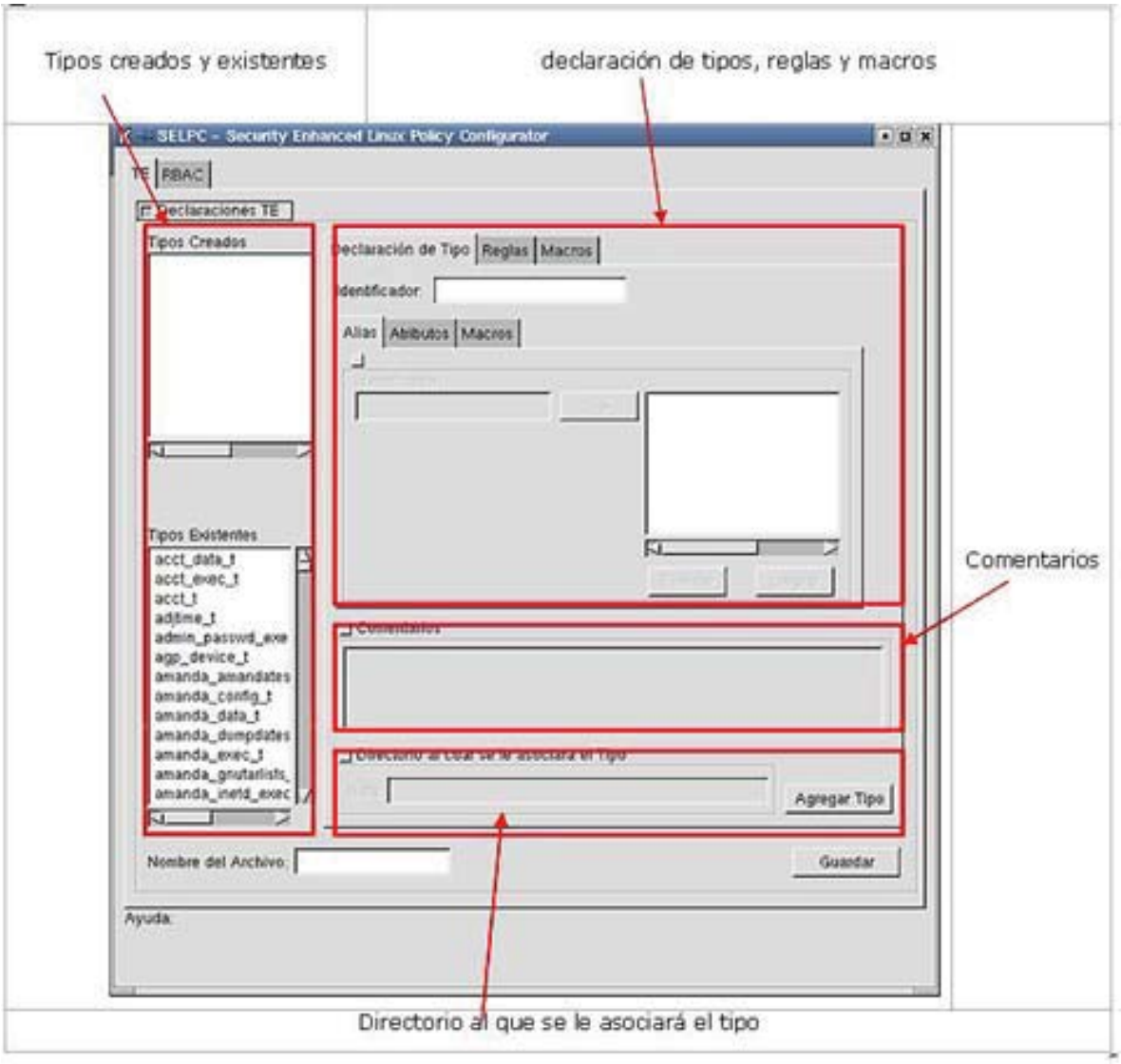
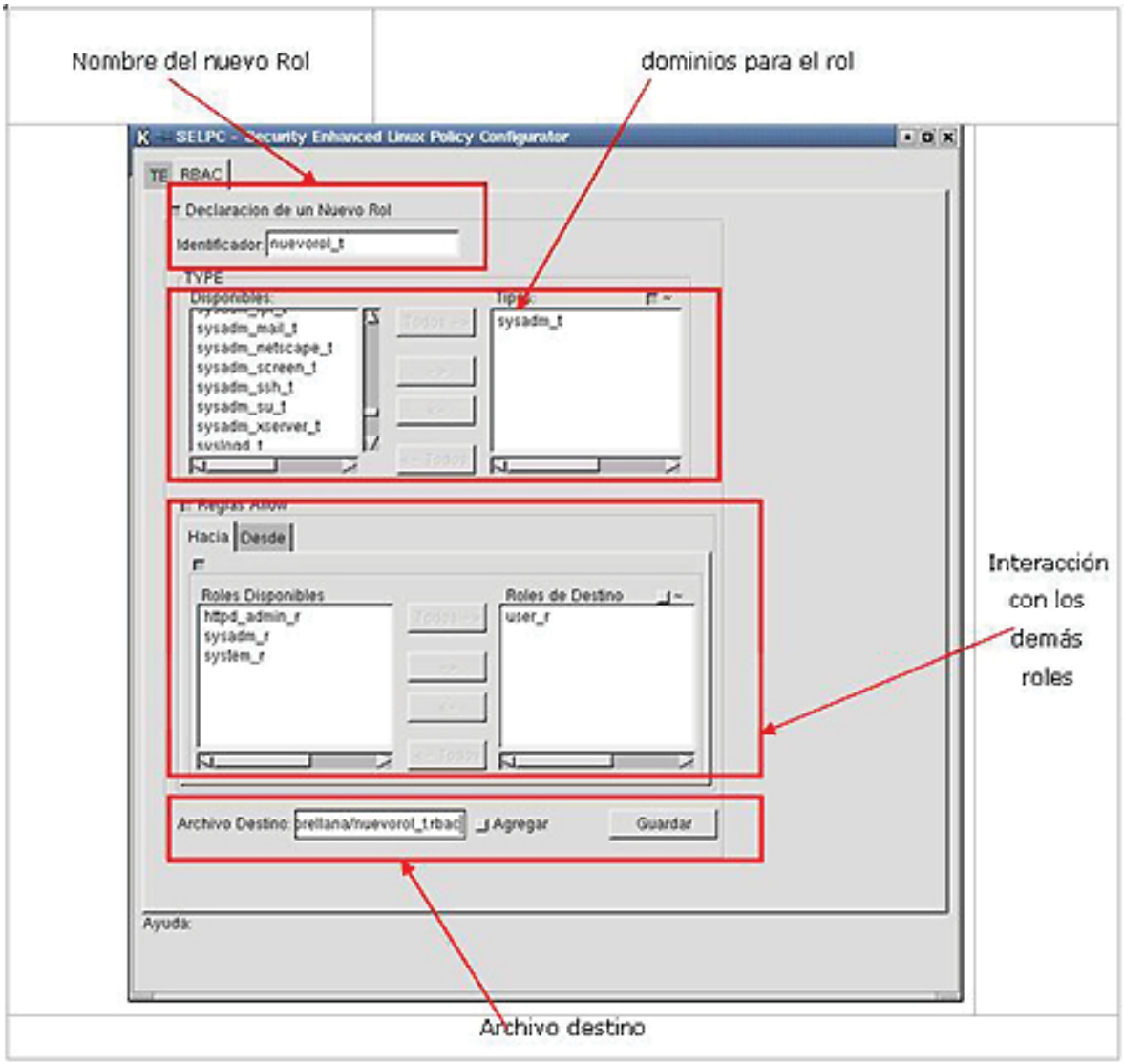


Figura 12.- Interfaz de SELPC – Pestaña RBAC.



1. Pestaña TE:

Se pueden observar dos áreas importantes dentro del frame “Declaración TE” (Figura 1): en la parte izquierda existen dos listas, que permiten visualizar los nuevos tipos que han sido creados y los tipos que están ya están definidos en la política (Figura 8). A

continuación existe campo de entrada de texto que permite definir el nombre del archivo donde se guardaran el nuevo tipos, reglas y macros definidas.

En el área derecha existe un notebook con tres pestañas donde se puede crear un nuevo tipo, y agregarle reglas y macros.

1.1 Pestaña Declaración de Tipo:

Esta pestaña se compone del campo “Identificador” (donde se define el nombre del nuevo tipo, por lo que es obligatorio llenarlo) y un notebook (donde se puede definir alias de tipo, los atributos y las macros, elementos que son opcionales).

Figura 13.- Pestaña “Alias”.

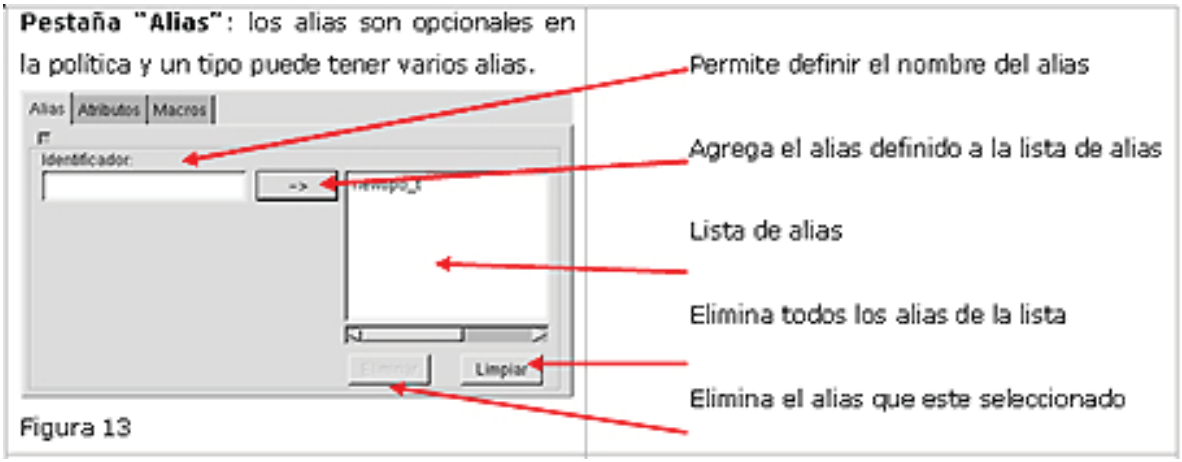


Figura 14.- Pestaña “Atributos”.

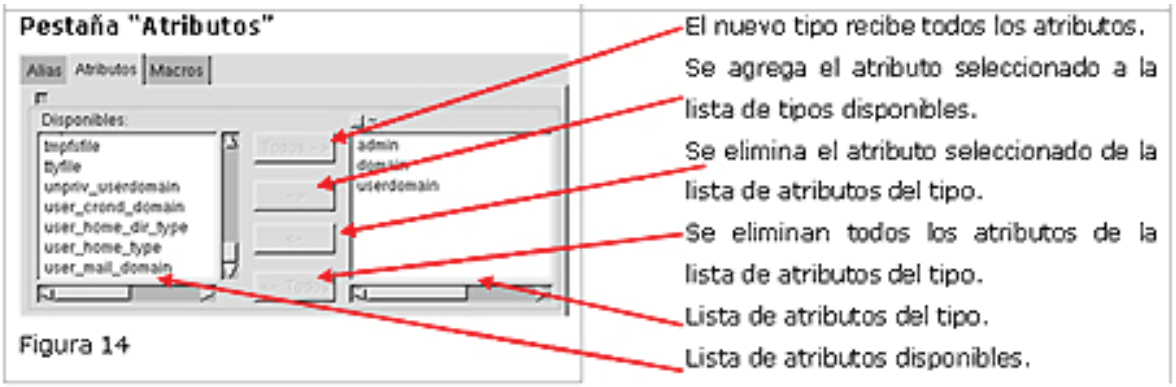


Figura 14

Figura 15.- Pestaña “Macros”.

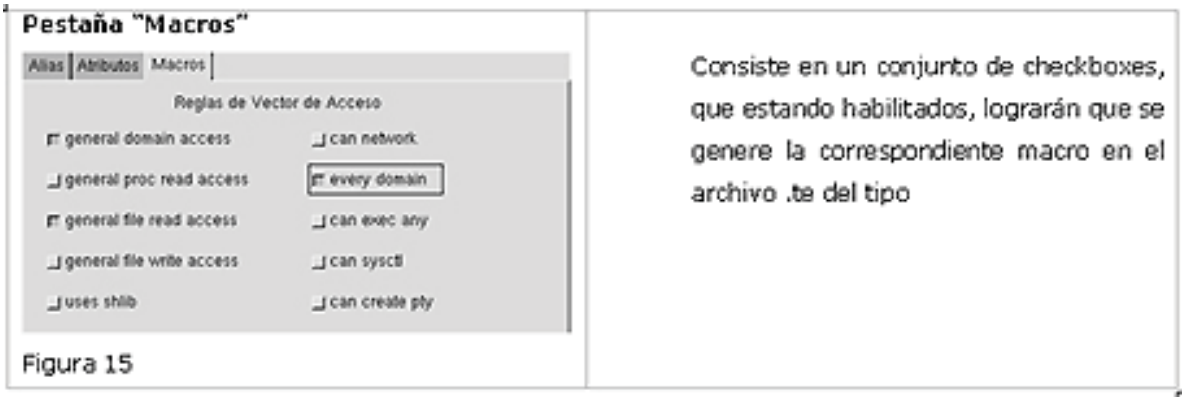


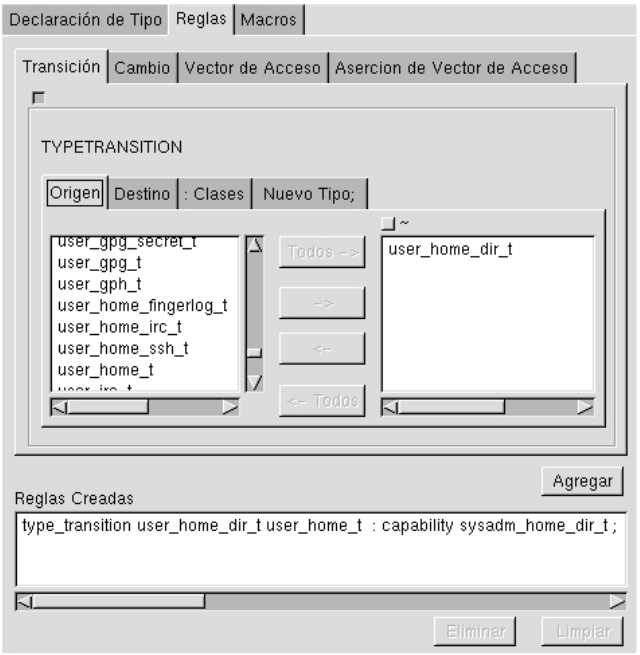
Figura 15

Opcionalmente se pueden ingresar comentarios que serán agregados en el archivo donde se guardaran las declaraciones del tipo.

Además, opcionalmente, se puede establecer la ruta o el archivo que debe ser etiquetado con el tipo que esta siendo declarado.

1.2 Pestaña Declaración de Reglas:

Figura 16.- Pestaña para declarar reglas



En la figura 12 se puede observar la pestaña donde se pueden definir las reglas. Esta compuesta por un notebook donde se pueden definir las reglas de transición de tipo, de cambio de tipo, de vector de acceso y aserción de vector de acceso. Cada una de estas pestañas funciona de forma similar.

El proceso típico para definir una nueva regla consiste de los siguientes pasos:

1. Elegir la pestaña que permite definir la regla deseada. Esta pestaña lleva otro notebook, donde cada pestaña representa una parte de la regla. Observando la imagen anterior, se aprecian: Origen, Destino, Clases, Nuevo Tipo.

2. Estas partes se comportan como la pestaña *atributos* en la *declaración del tipo*. Solo hay que desplazar uno o mas tipos desde la lista de la derecha(tipos disponibles) hacia la lista de la izquierda (tipos seleccionados).

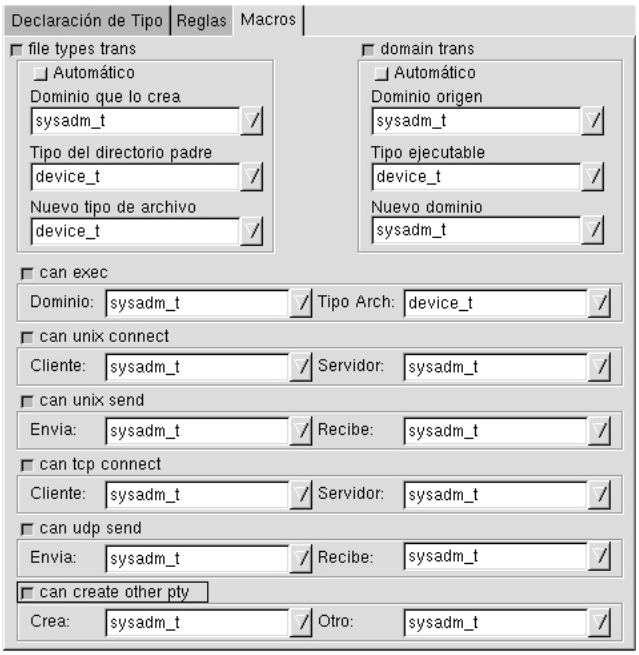
3. Una vez especificado lo deseado en cada pestaña, dar clic al botón **Agregar**.

4. La lista de reglas creadas recibe un nuevo elemento, que corresponde a la regla recién agregada, en el formato que maneja SELinux (lo que permite apreciar lo complejo que puede ser definir una regla manualmente, sin cometer errores).

5. Bajo esta lista, están los botones que permiten eliminar alguna de las reglas que hayamos definido (o todas).

1.3 Pestaña Macros:

Figura 17.- Pestaña para declarar macros en SELPC



Esta pestaña permite definir ciertas reglas básicas por medio de las macros disponibles con ese propósito en SELinux. Estas macros requieren 2 o 3 parámetros y, en general, habilitan ciertas transiciones o habilidades para crear objetos en ciertos dominios.

2. Pestaña RBAC:

Al habilitar el checkbox “Declaración de un Nuevo Rol” se puede crear un nuevo rol, para ello se debe especificar lo siguiente:

Identificador del rol.

Declarar los dominios para el nuevo rol.

Figura 18

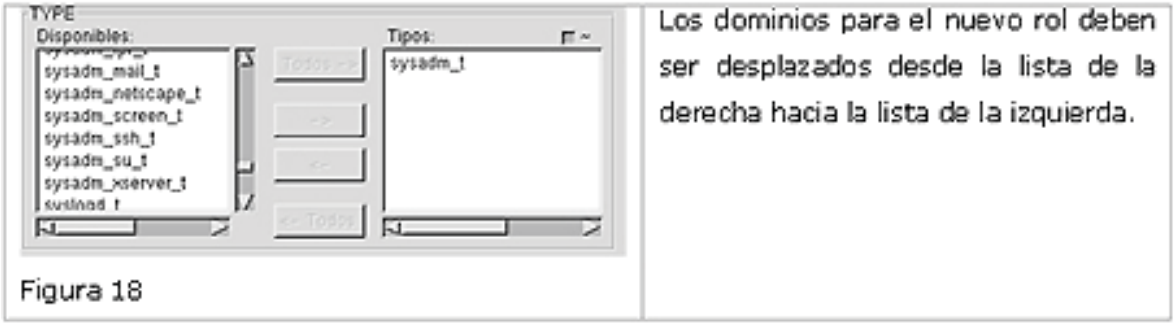


Figura 18

Declarar las relaciones con los demás roles.

Figura 19.

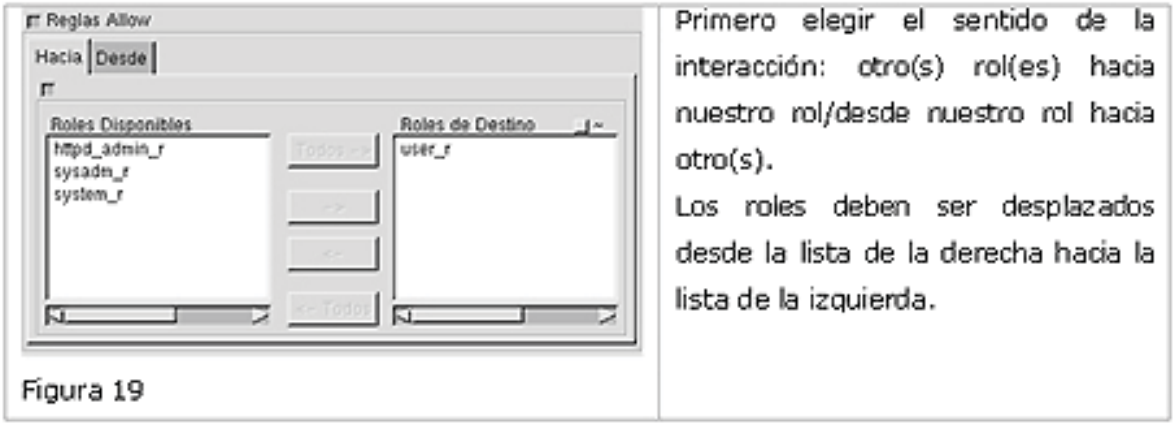


Figura 19

Especificar el archivo en donde se almacenaran las reglas.