INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

ADMINISTRADOR DE POLÍTICAS DE SEGURIDAD ORGANIZACIONALES

TESIS QUE PARA OPTAR EL GRADO DE
MAESTRO EN CIENCIAS COMPUTACIONALES
PRESENTA

KAREN AZURIM GARCÍA GAMBOA

Asesor:	DR. RAÚL MONROY BORJA	
Co-Asesor:	DR. JOSÉ DE JESÚS VAZQUEZ GÓMEZ	
Jurado:	DR. LUIS ANGEL TREJO RODRÍGUEZ	Presidente
	DR. JOSÉ DE JESÚS VAZQUEZ GÓMEZ	Secretario
	DR. RAÚL MONROY BORJA	Vocal

MAESTRÍA EN CIENCIAS COMPUTACIONALES

DICIEMBRE, 2005

CONTENIDO

1 INTRODUCCIÓN

La información es el principal recurso de una organización, preservar su integridad se ha convertido en una tarea fundamental necesaria para lograr los objetivos de la empresa que normalmente van enfocados a su crecimiento tanto económico como social.

La seguridad y en especial, las políticas de seguridad están tomando mayor importancia en la empresa ya que representan un mecanismo de protección de sus recursos principalmente de la información.

Las políticas de seguridad en general se definen como normas o lineamientos (de carácter obligatorio) necesarios en la protección de recursos. No existen estándares que deban seguirse para la descripción de una política, sin embargo es importante que éstas sean fáciles de entender y aplicables. Hablando específicamente de las políticas de seguridad de información, éstas apoyan la protección, control y dirección de los recursos de información de la organización (TI).

Escribir políticas de seguridad no es una tarea sencilla debido a que debe garantizarse el correcto funcionamiento de un sitio que está expuesto a una serie de amenazas, eso trae como consecuencias que muchas empresas aún no cuenten con un documento de políticas de seguridad.

Por otro lado, solo algunas de las organizaciones que han diseñado sus políticas de seguridad, llevan a cabo una adecuada administración de éstas, mientras que el resto de ellas no aseguran su validez y beneficio dentro de la empresa. De ahí surge la necesidad de evaluar cada una de las políticas escritas evitando contradicciones entre ellas mismas, ya que este hecho representaría vulnerabilidades latentes expuestas a diversas amenazas.

En la actualidad no existen herramientas que ayuden o faciliten la captura de políticas de seguridad organizacionales. Sin embargo, esta investigación incluye un capítulo en el que se estudian algunos lenguajes utilizados para especificar políticas de seguridad en aplicaciones, y sobre los que se extrajeron algunas ideas para diseñar un prototipo capaz de administrar políticas de seguridad.

El trabajo principal de esta tesis consiste en el diseño de una herramienta administradora de políticas de seguridad, las principales características de esta herramienta incluyen: 1) una interfaz gráfica utilizada para capturar políticas de seguridad mediante el uso de ventanas que surgen conforme se va construyendo una política, 2) un traductor de políticas que formaliza éstas en lógica de primer orden con sintaxis de *Otter* (demostrador automático de teoremas de primer orden), y 3) abre una consola gráfica de *Otter* para manipular su uso ya que éste trabaja mediante comandos del sistema operativo.

El prototipo fue desarrollado en lenguaje java para no limitar la herramienta a una plataforma en especial. Las secciones 4, 5 y 6 explican el trabajo realizado en la construcción del prototipo administrador de políticas de seguridad.

El prototipo administrador de seguridad que se presenta en este documento pretende únicamente capturar y validar políticas de seguridad, no concretar la aplicación de las mismas en una organización (implementación de las políticas capturadas).

2 MARCO TEÓRICO

2.1 INTRODUCCIÓN

La falta de políticas y procedimientos en seguridad es uno de los problemas más graves que enfrentan las empresas hoy en día en lo que a la protección de sus activos de información se refiere, frente a peligros externos e internos. Las políticas de seguridad son esencialmente normas (de carácter obligatorio) que indican cómo manejar los asuntos de seguridad y forman la base de un plan estratégico para la implantación efectiva de medidas de protección tales como: identificación y control de acceso, respaldo de datos, planes de contingencia, detección de intrusos, entre otros.

Una declaración sobre políticas de seguridad describe solamente la forma general de manejar un problema específico, pero no debe ser demasiado detallada o extensa debido a que se convertiría en un procedimiento. Las siguientes secciones introducen al lector en conceptos generales de políticas de seguridad organizacional enfocadas a la información.

2.2 POLÍTICAS DE SEGURIDAD ORGANIZACIONAL

Una organización, ya sea comercial, educacional, gubernamental o cualquier otro tipo necesita proteger su información sin importar el medio en que se almacene o trasmita, así como sus demás recursos. Una política de seguridad es en general, el total de todos los procedimientos y

reglamentos organizacionales relacionados con la seguridad de los recursos valiosos entre ellos la información [1]. Una política de seguridad debe escribirse anticipándose a problemas futuros.

Antes de desarrollar un documento de políticas de seguridad es necesario realizar un análisis de riesgos, este análisis consta de una tabla en la que se plasman los recursos de la empresa y los riesgos a los que están expuestos dichos recursos. Un correcto análisis de riesgos es la pauta para construir un conjunto de políticas correctas que efectivamente cumplan con su objetivo que es el de proteger los activos de una empresa.

Además de proteger la seguridad de los recursos valiosos de una empresa (información), existen otras razones de peso para establecer una política de seguridad, algunas de ellas son: legales, regulatorias, contractuales, todas ellas encaminadas al bienestar de la empresa. Al escribir políticas de seguridad deben considerarse algunos conceptos importantes (se muestran en la figura 2.1.) y dos principales características que deben cumplirse para que se consideren como adecuadas:

- Deben ser fáciles de entender. Es importante que el documento de políticas esté escrito en un nivel de comprensión sencillo de tal forma que todo aquel a quien va dirigido este documento sea capaz de entender y llevar a la práctica los lineamientos ahí presentados.
- Deben ser aplicables. Esta característica es importante para asegurar que todo lo que se escribe reúne las necesidades de cada organización en específico. De ahí surge la necesidad de estandarizar un sistema que permita describir políticas de seguridad que puedan ser aplicables a múltiples necesidades.

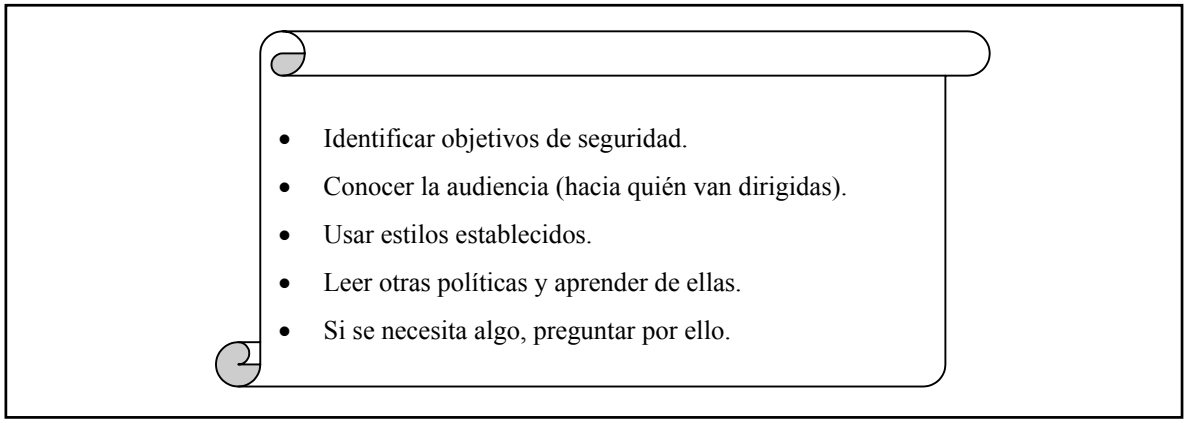


Fig. 2.1. Puntos importantes para desarrollar políticas de seguridad.

Las políticas de seguridad varían de acuerdo a la organización. No existe una política que se considere como la "mejor" porque cada organización es única; cada una tiene sus propios objetivos, propósitos e intereses y estas políticas deben desarrollarse tomando en cuenta estos elementos.

A pesar de que las políticas varían dependiendo la empresa para la que fueron desarrolladas y no es necesario clasificarlas, varios autores dedicados al estudio de seguridad mencionan y recomiendan escribir las políticas de seguridad dividiéndolas en tres tipos que son: generales, de tema específico y de aplicación específica [2]. Muchos otros autores unen las políticas de tema específico con aplicación específica y las incluyen en la clasificación de políticas específicas.

- Las políticas generales, se usan para crear la visión general de seguridad de la información de una organización.

- Las políticas de tema específico, se dirigen a temas de seguridad específicos que preocupan a la organización.

- Las políticas de aplicación específica, se enfocan en decisiones tomadas por la dirección principal para proteger aplicaciones o sistemas particulares.

2.2.1 POLÍTICAS DE SEGURIDAD DE INFORMACIÓN

La política de seguridad de información apoya la protección, control y dirección de los recursos de información de la organización (TI). Estas políticas son requeridas para cubrir toda la información dentro de la organización qué podría incluir datos e información que son:

- Almacenados en bases de datos.
- Impresos o escritos a mano en papel, pizarrón etc.
- Almacenados en medios de comunicación trasladables como CD-ROMs, Zip Disc TM, discos duros y otros medios de comunicación similares.
- Almacenados en medios de comunicación fijos como discos duros y sistemas subalternos del disco.

- Presentados en diapositivas, proyectores, usando medios de comunicación visuales y de audio.
- Transmitidos por cualquier medio de comunicación.

Los objetivos de las políticas de seguridad de información son [3]:

- Proteger la información de la empresa y la información de cualquier cliente en su custodia salvaguardando su confidencialidad, integridad y disponibilidad.
- Establecer resguardos que protejan la información de robo, abuso, mal uso o cualquier forma de daño.
- Establecer responsabilidades de los involucrados en la protección de la información.
- Proveer normas y estándares al personal involucrado con la seguridad de información para que tengan la habilidad de minimizar la ocurrencia y severidad de incidentes.
- Asegurar que la organización es capaz de continuar sus actividades comerciales en caso de incidentes significativos en la seguridad de información.

2.2.2 IMPORTANCIA DE IMPLANTAR POLÍTICAS DE SEGURIDAD DE INFORMACIÓN

Cuando una compañía no cuenta con un documento de políticas de seguridad de información, ésta podría encontrarse en graves peligros de romper la seguridad establecida, perder ventaja competitiva y perder credibilidad ante aquellos a los que ofrece un servicio. Implementando políticas de seguridad, la organización toma control de su funcionamiento y reduce la probabilidad de que las vulnerabilidades presentes en la información sean aprovechadas por una amenaza externa o interna.

El punto clave es que toda organización necesita un conjunto de políticas que protejan la información. Estudios realizados en los últimos años encontraron que los principales crímenes sobre la información ocurren en organizaciones que no cuentan con un documento escrito de políticas de seguridad. El programa de políticas crea una actitud positiva hacia la información y muestra que información es un recurso valioso propiedad de la empresa y la cual debe protegerse de accesos no autorizados, modificación y destrucción ya sea deliberada o accidentalmente.

2.3 CONCLUSIONES

Concluyendo, las políticas de seguridad adecuadas representan un elemento fundamental dentro de toda organización para establecer la seguridad de sus recursos valiosos y contribuyen en el ejercicio laboral y/o empresarial. La falta de políticas en seguridad es uno de los problemas más graves que confrontan las empresas hoy día en lo que se refiere a la protección de sus activos de información frente a peligros externos e internos.

Una política de seguridad es adecuada si contribuye en el establecimiento de metas de seguridad que incluyen: integridad, disponibilidad, confidencialidad, autenticación y no repudiación, sin obstaculizar la misión de la organización para la que son desarrolladas.

Si bien se mencionó que las políticas varían considerablemente según el tipo de organización de que se trate, en general incluyen declaraciones globales sobre metas, objetivos, comportamiento y responsabilidades de los empleados en relación a las violaciones de seguridad.

Escribir políticas de seguridad consiste en desarrollar planes generales que garanticen el correcto funcionamiento de una organización que opera en un ambiente posiblemente hostil. No es tarea sencilla porque es fácil cometer errores, las políticas pueden ser inconsistentes y ambiguas además de que consumen muchos recursos humanos y materiales que la mayoría de las veces se refleja en presupuestos muy elevados.

3 HERRAMIENTAS PARA POLÍTICAS DE SEGURIDAD

3.1 INTRODUCCIÓN

Escribir políticas de seguridad consiste en desarrollar planes generales que garanticen el correcto funcionamiento de un sitio que opera en un ambiente expuesto a una serie de amenazas esperando la oportunidad de aprovechar una vulnerabilidad para llevar a cabo su cometido de causar daños a la información de la empresa.

Pocas empresas administran adecuadamente sus políticas de seguridad, por lo que éstas a menudo contienen una gran cantidad de inconsistencias e imprecisiones[1], además de que pocas veces se aplican y muchas otras se ignoran. Esto provoca la necesidad de contar con herramientas automatizadas que, no sólo disminuyan los errores de administración, sino que además fomenten la administración misma.

Para la realización de la presente tesis se investigó la existencia de herramientas que resuelva el problema de administración de políticas de seguridad, que además valide posibles inconsistencias entre ellas. Desafortunadamente, no se encontraron tales herramientas. Sin embargo existen lenguajes que especifican políticas de seguridad (algunos de ellos se consideran tema de investigación) para mantener seguridad en equipos de computo, pero estos lenguajes además de requerir conocimientos básicos de seguridad y programación, no resuelven el problema de administrar políticas de seguridad organizacionales.

[1] El Dr. Jesús Vázquez observó el interés de algunos ingenieros por contar con un prototipo administrador de políticas de seguridad y en colaboración con el Dr. Raúl Monroy (asesor) propusieron la creación de esta tesis.

En este capítulo hablamos de estos lenguajes que, a pesar de no ser la herramienta adecuada para desarrollar y administrar políticas de seguridad es una investigación que puede aportar ideas para mejorar el administrador de políticas de seguridad.

3.2 LENGUAJES QUE ESPECIFICAN POLÍTICAS DE SEGURIDAD

Los lenguajes para especificar políticas de seguridad son una formalización de peticiones realizadas por usuarios, de forma tal que puedan ser leídas e interpretadas por computadoras. Cada lenguaje especifica una política definiendo entidades (programas, usuarios, comunicaciones, etcétera) y atributos de acuerdo a las acciones y permisos que puedan ser otorgados tomando en cuenta los criterios que cumplen.

La mayoría de los lenguajes que especifican políticas trabajan con formato de texto y este formato permite más errores que recaen en la necesidad de verificar posteriormente las especificaciones de las políticas. A continuación se desglosa brevemente la descripción de estos lenguajes.

3.2.1 KEYNOTE

Keynote [5] es un lenguaje que nos permite especificar políticas de seguridad para aplicaciones computacionales que trabajan en redes de comunicación. Un ejemplo ilustrativo para describir su uso principal es en una aplicación de correo electrónico. Keynote es el encargado de determinar si un usuario determinado está autorizado para leer un mensaje basándose en su llave pública o privada y en las políticas de seguridad especificadas por el administrador.

Keynote es un lenguaje basado en texto, que estudia formalmente las condiciones obtenidas de una aplicación para determinar cómo se relacionan con las políticas dadas. Keynote es un motor que responde a una pregunta enviada por una aplicación, la respuesta se basa en atributos de acciones específicas.

Un ejemplo de la estructura del lenguaje se muestra en el cuadro 3.1. El ejemplo muestra el pseudocódigo para la validación de solicitudes enviadas en una aplicación en red. El sistema Keynote valida si la solicitud se permite o no y envía la respuesta correspondiente al solicitante.

```
/* En cada punto de la aplicación cuando se determina que alguien solicita una
acción se hace lo siguiente: */

    requester           = requesting principal's identifier;
    action_description  = data structure describing action;
    policy              = data structure describing local policy, typically
                          read from a local file;
    credentials         = data structure with any relevant credentials,
                          typically sent along with the request by the
                          requesting principal;
    PCV                 = Call_KeyNote (requester, action_description,
                          policy, credentials);
    if (PCV == "allowed")
            do the requested action
    else
            tell principal that action isn't allowed
    endif
```

Cuadro 3.1. Pseudocódigo básico para usar Keynote en una aplicación.

La principal aplicación del lenguaje Keynote se presenta en sistemas o aplicaciones donde es necesario determinar si un conjunto de condiciones satisfacen o no un conjunto de políticas dadas. El lenguaje no discierne el significado de las condiciones y políticas que tiene para una aplicación específica, éste solo estudia formalmente las condiciones dadas para determinar cómo se relacionan con las políticas establecidas.

Keynote es un lenguaje de especificación general considerado como excelente por investigadores en materia, pero su generalidad presenta problemas al definir relaciones del ambiente y sus atributos, así como los posibles valores de retorno y sus interpretaciones debido a que diferentes interpretaciones hacen difícil la combinación de las políticas de seguridad. Este problema podría solucionarse mediante la combinación de un lenguaje general como Keynote con relaciones ajustadas de un ambiente de seguridad específico, ejemplo firmas de e-mail o reglas de firewall. Esto sería comparable a usar un lenguaje de descripción general como XML y ajustarlo a las aplicaciones específicas.

En conclusión, el lenguaje Keynote representa una solución efectiva para aplicaciones que trabajan en red. Para la descripción de políticas de seguridad organizacionales se podría trabajar sobre la herramienta adaptando las investigaciones que se han realizado hasta la fecha y orientándolas a la especificación de políticas de seguridad organizacionales. Sin embargo no solucionaríamos el problema de especificar políticas de forma sencilla, ya que se necesitan conocimientos mínimos de programación además de que el formato de texto permite que haya más errores en la descripción.

3.2.2 SPSL

SPSL [5] es un lenguaje diseñado específicamente para expresiones de políticas de seguridad. Se concentra en expresar políticas para proteger la comunicación entre equipos de cómputo. Puede usarse para definir reglas de firewall o parámetros aplicados a conexión VPN. Sin embargo fue creado pensando en el uso de conexiones de seguridad IPSEC, así como el manejo de sistemas IKE Key.

Este lenguaje usa objetos para representar políticas de seguridad, cada objeto tiene ciertos atributos que pueden ser obligatorios u opcionales y éstos contienen los datos de cada ejemplar de objeto por clases. SPSL trata con políticas basadas en nodos o en dominio, dichos nodos o dominios se asocian al objeto. Un nodo es una representación de una sola entidad de red que tiene por lo menos una interfaz y un nombre DNS. Un dominio es un conjunto de nodos en la red que requiere de por lo menos un servidor de políticas que maneje las políticas de dicho dominio.

El cuadro 3.2 muestra un ejemplo básico de un atributo de una política de seguridad.

```
policy:  dst 193.197.128.43 \
         src * \
         xport-proto 6 permit
```

Cuadro 3.2. Esta política permite todas las conexiones a 193.197.128.43 desde cualquier fuente usando TCP.

Actualmente, este lenguaje se usa principalmente para especificar reglas de firewalls o para definir conexiones IPSEC/IKE o con cualquier problema orientado a comunicación. SPSL es un

lenguaje diseñado para el problema de filtrar comunicaciones. Sin embargo, es posible usarlo para otros propósitos, extendiendo la sintaxis actualmente definida.

SPSL y Kenynote son lenguajes buenos para definir medidas que son aplicadas en un punto final de la comunicación o posiblemente entre dos entidades con la misma política. En ambientes más grandes con múltiples actores y diferentes agendas de trabajo, algunas formas de negociación necesitan estar de acuerdo en una política consistente para diferentes dominios de comunicación y posiblemente diferentes organizaciones.

Ambos lenguajes podrían ser una herramienta adecuada en la descripción de políticas de seguridad organizacional, pero su uso requiere de conocimientos de programación limitando su utilización. Una posible solución a sus limitantes es realizar algunas modificaciones a la estructura del lenguaje así como una interfaz gráfica que facilite su uso a los usuarios. Sin embargo, esto implicaría un estudio exhaustivo de la estructura actual del lenguaje que repercutiría en los recursos involucrados. Por otro lado, tanto SPSL como Keynote para que sean realmente efectivos necesitan incluir un lenguaje de especificación formal, el cual deberá ser también formalmente verificable y completo. Esta situación complica más la tarea de especificar políticas de seguridad organizacional.

3.2.3 LASCO

LASCO [6] es un lenguaje creado para la seguridad en aplicaciones y trabaja mediante restricciones a objetos. Representa las políticas en forma de grafos dirigidos y en lógica matemática. Es un lenguaje visual permitiendo mayor facilidad de manejo y entendimiento de las políticas declaradas por parte de los usuarios. Permite declarar políticas en sistemas operativos y lenguajes de programación, en particular los orientados a objetos.

En este lenguaje, una política consta de dos partes, el dominio y los requerimientos. El dominio determina el alcance de la política y los requerimientos describen las restricciones de la política. Los nodos del grafo representan objetos en el sistema, los arcos representan los eventos del sistema y su semántica se basa en la lógica de primer orden.

Un ejemplo representativo del lenguaje es un sistema de administración de exámenes en línea, los cuadros 3.3 y 3.4 muestran el código de los objetos del sistema con su identificador y atributos correspondientes:

```
type="user"
name: a string
roles: a set from
        {"instructor", "student",
         "admin", "secadmin"}
courses_taught: a set of strings
...
```

Cuadro 3.3. Código de un objeto en un sistema educativo que representa un usuario del sistema.

```
type="exam_db"
course: a string        ...
exam_id: an integer
created: an integer
...
```

Cuadro 3.4. Código de un objeto en un sistema educativo que representa una base de datos de exámenes.

Los cuadros 3.5 y 3.6 muestran el código de los eventos del sistema, en el se incluyen diferentes acciones del sistema de examen, cada una con diferentes parámetros y atributos:

```
action="submit-answer"
time: an integer
answer: a string
```

Cuadro 3.5. Código de un evento del sistema educativo que representa una solicitud enviada.

```
action="post-soln"
time: an integer    ...
soln: a string
...
```

Cuadro 3.6. Código de un evento del sistema educativo que representa la solución del examen enviada a la base de datos.

El diagrama 3.1 representa el ejemplo de un sistema de exámenes en línea en el que cada estudiante enviará las respuestas de su examen al servidor en un tiempo determinado, después de ese tiempo es obligación del profesor a cargo enviar a la base de datos la solución del examen para que sea accedido por los estudiantes después de haber enviado sus respuestas. Nunca la solución al examen deberá estar en la base de datos antes que las respuestas de cada estudiante registrado.

De esta forma queda establecida la política de seguridad para que la aplicación que trabaja bajo red proteja la información de la base de datos (la solución al examen) y ésta no sea publicada antes de que todos los estudiantes registrados hayan enviado sus respuestas.

El uso de grafos facilita la representación de políticas de seguridad especificando controles de acceso sobre la información y de esta forma proteger su integridad, confidencialidad y disponibilidad. Por otro lado pueden especificarse tiempos de uso y los usuarios autorizados solamente tendrían acceso a la información por un periodo determinado.

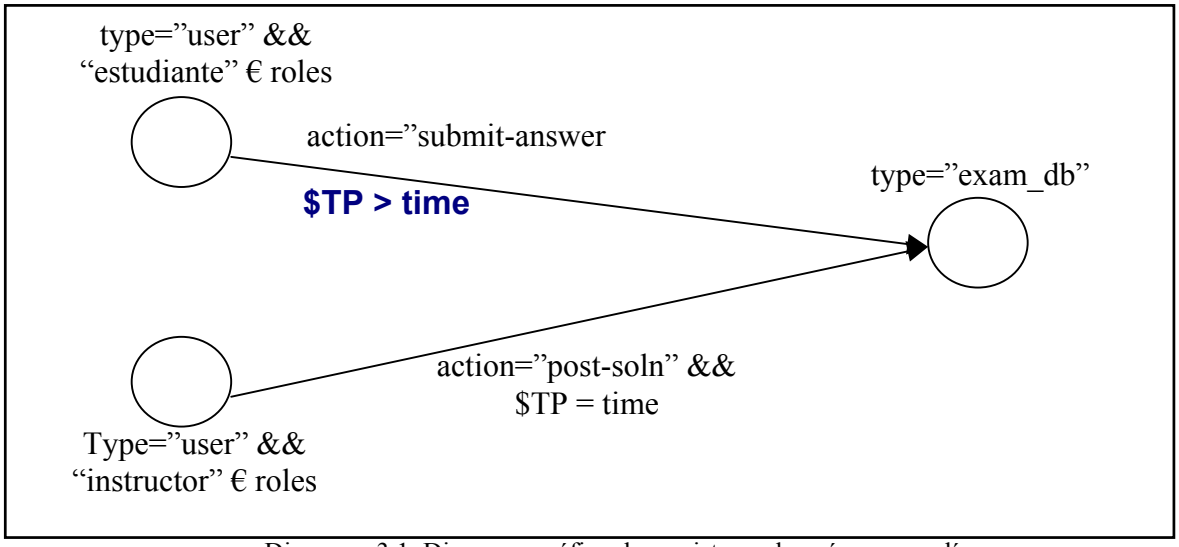


Diagrama 3.1. Diagrama gráfico de un sistema de exámenes en línea.

El uso de grafos facilita el entendimiento de una política de seguridad por lo que adaptamos la idea principal del leguaje LASCO para la especificación de políticas de seguridad organizacional específicas. Los elementos del lenguaje utilizados en el prototipo son los grafos que representan sujetos y objetos y los arcos que unen los grafos representan las acciones que pueden o no realizar los sujetos sobre los objetos.

3.2.4 MODELO *KST*

El Modelo propuesto por Ivan Krsul, Eugene Spafford y Tugkan Tuglular [7] representa las políticas como expresiones algorítmicas y matemáticas. El modelo provee un mecanismo para relacionar políticas desde niveles superiores hasta especificaciones detalladas. Identifica los elementos relevantes de la política y ofrece la facilidad de identificar información para detectar posibles violaciones en las políticas.

La política dentro del modelo se define con la palabra *policy*, y es un predicado que regresará el valor de verdadero o falso dependiendo si la operación violó la política. La función *system value*, ayuda a *policy* a calcular el valor del sistema, en el estado particular en que se encuentre. La función *object value* es necesaria dentro de *system value* para cada uno de los componentes de dicha función y nos devolverá el valor del objeto. Así como los lenguajes estudiados en secciones anteriores, este lenguaje se enfoca principalmente en aplicaciones computacionales que normalmente trabajan en red.

Como se muestra en el cuadro 3.7, una política puede ser definida como una función que toma como argumentos dos números, un valor de función y dos conjuntos de intereses (antes y después de la ejecución de una instrucción), y especifica sí el cambio en el valor causado por la instrucción es aceptado.

```
Policy : integer x integer x Value function x
         set of interest x set of interest -> boolean
fun Policy(a,b,Value,Ii,Ii+1) ::=
    if a* Value(Ii) <= Value(Ii+1)  Value(Ii+1)<= b * Value(Ii) then
        Policy:=true;
    else
        Policy:=false;
    fi
nuf
```

Cuadro 3.7. Ejemplo del lenguaje utilizado en el modelo KST.

El modelo es aún una propuesta, sin embargo es importante mencionarlo por el hecho de que representa políticas como un proceso semejante al ciclo de desarrollo de software. Mismo proceso que facilita a un desarrollador de sistemas de seguridad implementar el modelo en un administrador de políticas permitiendo su refinamiento. El modelo también está preparado para el

desarrollo de un mecanismo que se use para detectar violaciones en la política que puedan ser expresadas con el modelo.

3.3 CONCLUSIONES

Los lenguajes de especificación de políticas nos ofrecen las herramientas necesarias para expresar las políticas de seguridad de un sistema computacional o dispositivos que protegen la seguridad de la información dentro de cualquier organización. Cada uno de los lenguajes que en la actualidad se usan, tienen sus ventajas y desventajas sobre otros, así como sus características particulares, lo cual no los hace mejores o peores, sino que cada investigación tiene su propósito y por tanto la utilización de cualquiera de ellos dependerá de los propósitos del investigador.

Estas herramientas sirven a los investigadores de seguridad –específicamente de políticas de seguridad–, como base para nuevas investigaciones utilizando las principales ideas y ventajas de cada lenguaje y transportándolas a una herramienta administradora de políticas de seguridad.

La investigación de las herramientas estudiadas en este capítulo nos ayudó a evaluarlas y determinar cuál pudiera ser la más apropiada para la representación de políticas de seguridad organizacionales. Aunque estos lenguajes no representan la mejor alternativa en la construcción de políticas de seguridad, nos proveyó de ideas que fueron tomadas en cuenta e implementadas como parte del prototipo.

El lenguaje en el que nos basamos (idea principal) fue LASCO, que trabaja mediante el uso de grafos para mostrar las políticas a los usuarios e internamente maneja lenguaje de programación para la interpretación computacional correspondiente de la política de seguridad.

4 INTERFAZ GRÁFICA DE USUARIO

4.1 INTRODUCCIÓN

Al escribir un enunciado se produce el proceso de planificar la redacción, y resulta difícil encontrar una formulación adecuada que disminuya el tiempo y la energía usada en el proceso de escritura. Naturalmente el trabajo final deberá ser un texto limpio y esperamos bien formulado.

En esta tesis, para ayudar en el proceso de expresar y administrar un conjunto de políticas adecuadas a las necesidades de una empresa, sugerimos un sistema administrador con las siguientes características:

El sistema debe proporcionar al usuario una forma sencilla de capturar políticas de seguridad y facultar el análisis formal de éstas considerando que el administrador de seguridad típico, no posee entrenamiento especial en métodos formales (lenguaje de primer orden). Esta formalización permitirá al usuario verificar sí las políticas desarrolladas para una organización son correctas. El prototipo se constituye de diferentes componentes que son la interfaz gráfica de usuario, el demostrador de teoremas de primer orden utilizado para el análisis formal de las políticas y por último un analizador que transforma las políticas introducidas en sintaxis de primer orden.

El propósito de que el sistema cuente con una interfaz gráfica es proporcionar al usuario una forma sencilla de capturar políticas de seguridad, pero que al mismo tiempo puedan ser leídas y procesadas por un equipo de cómputo. Una forma sencilla de capturar enunciados correspondería

a un cuadro de diálogo en el que mediante lenguaje coloquial el usuario pudiera introducir cada una de las políticas. Sin embargo esto representa un problema significativo ya que estas políticas introducidas al sistema deberán ser traducidas a sintaxis de lógica formal para su validación -recordemos que todo sistema computacional es la interpretación del conocimiento de un individuo en particular, y cada individuo tendrá su propia interpretación-. Como consecuencia resulta sumamente difícil manejar ideas mediante un proceso semejante ya que podrían encontrarse interpretaciones erróneas de lo que realmente el usuario intentó escribir.

La solución propuesta a este problema fue construir una interfaz mediante sub-ventanas que dirijan y limiten al usuario en el proceso de escritura. El presente capítulo muestra el contexto de captura de políticas de seguridad sobre información y las conclusiones a las que se llegó.

4.2 DESARROLLO

Como vimos en el primer capítulo, las políticas de seguridad se dividen en tres tipos, pero para efectos de esta tesis se limitó a sólo los dos siguientes:

- Generales [1]: Usadas para crear el contexto de la visión de seguridad de los recursos de una organización. El objetivo principal de estas políticas tomando en cuenta que el recurso a proteger es la información se enfoca en la preservación de su integridad, confidencialidad, disponibilidad, y autentificación.

- De tema específico [1]: Éstas tratan puntos específicos de preocupación para la organización y cuya seguridad representa la estabilidad organizacional.

4.2.1 POLÍTICAS DE SEGURIDAD GENERALES

Para la representación de políticas de seguridad generales consideramos las cuatro propiedades que debe cumplir la información y se propusieron un conjunto de políticas que preservan dichas propiedades. Estas políticas pueden incluirse a voluntad del usuario según lo requiera, y puede modificarlas ligeramente en cuanto a su descripción sin embargo el contexto sigue siendo el mismo:

1. Toda información propiedad de *la empresa* deberá estar protegida adecuadamente en cuanto a su integridad, confidencialidad, disponibilidad y autenticación.

Integridad

2. Todo daño deliberado, robo o modificación no autorizada de la información propiedad de *la empresa* deberá ser sancionado.

3. Todo daño causado por negligencia a la información propiedad de *la empresa*, deberá ser sancionado.

4. Los custodios de la información son responsables de proveer un ambiente seguro en el cual pueda ser mantenida con integridad.

Confidencialidad

5. Toda información propiedad de la empresa deberá ser generada, eliminada o modificada únicamente por aquellas personas autorizadas para hacerlo y en aquellos lugares autorizados por *la empresa*.

6. Toda información propiedad de *la empresa* debe clasificarse de acuerdo a su nivel de importancia para el negocio.

7. El acceso a la información de la compañía se restringe solamente a usuarios autorizados por el administrador de seguridad.

Disponibilidad

8. Toda información propiedad de *la empresa* deberá estar disponible sólo a aquellas personas a quienes está destinada.

Autentificación

9. El administrador de seguridad es responsable de proveer un mecanismo de acceso a los usuarios autorizados para manipular la información de la empresa.

4.2.2 POLÍTICAS DE SEGURIDAD PARTICULARES

Para representar políticas de seguridad particulares sobre archivos de información lo principal fue encontrar un esquema constituido por los principales componentes involucrados en el manejo, distribución y mantenimiento de la información:

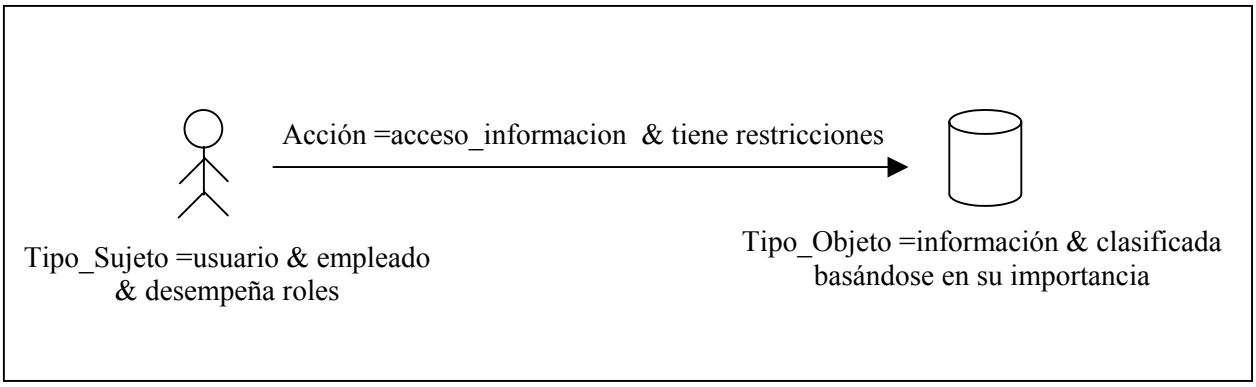


Figura 4.1. Se muestran los componentes de una política: sujeto, objeto y acción. Los sujetos son empleados con ciertos roles que ejecutan una acción con restricciones sobre la información clasificada de acuerdo a su importancia.

Sujetos: Los sujetos principalmente representan una entidad dinámica dentro de la organización como son los empleados –quienes ocupan un nivel jerárquico de acuerdo a la importancia de sus actividades laborales– y la ejecución de procesos o aplicaciones, que influyen directamente en el estado actual de un archivo de información. Estas entidades mediante una acción específica de acceso modifican la estructura o contenido de un archivo de información, el cual debe preservar sus propiedades de integridad, confidencialidad, disponibilidad y autentificación, y para lo cual se requiere de un conjunto de restricciones, que a su vez estarán establecidas en base a los roles que desempeñen los sujetos dentro de la organización.

Objetos: Los objetos representan la información –recurso principal de toda organización– que debe necesariamente clasificarse para la implantación de mecanismos de acceso que protejan su seguridad mediante la restricción a usuarios no autorizados evitando perdidas cuantiosas que repercutan en los objetivos y metas de la empresa. Los elementos de la base de datos que pueden

tomar el papel de objetos son los elementos localizados en la tabla TB_IMPORTANCIA_INFO y en la tabla TB_OBJETOS:

Importancia de la información para la organización
Importancia
Confidencial
Importante
General

Tabla 4.1. Importancia de la información.

Los datos de ésta tabla se recomiendan al administrador, sin embargo pueden ser modificados de acuerdo a sus necesidades. Esta clasificación es general y diversos autores de seguridad informática los proponen.

Información de la empresa	
Información	**Importancia**
Estados financieros	Confidencial
Nomina	Confidencial
Archivo de passwords	Confidencial
Archivos locales	Importante
Politicas generales	General
Informacion generada por personal de la empresa	Importante
Plan de contingencias	Confidencial
Analisis de mercados	Confidencial
Procedimientos de recuperacion de desastres	Confidencial
Documento de analisis de riesgos	Confidencial

Tabla 4.2. Ejemplos de información de una organización.

Los datos introducidos en la tabla de objetos dependerán de la información propiedad de cada empresa y su importancia dentro de las actividades del negocio.

Acciones: Representan la relación que existe entre el sujeto y el objeto. Esta relación depende directamente de los roles de los sujetos y de la clasificación de la información. Basándose en ello, el administrador de seguridad es el encargado de realizar una lista de control de acceso, en la cual se visualiza cada uno de los sujetos y objetos registrados y qué acciones pueden ejecutarse sin poner en riesgo la seguridad de la información. Las acciones que puede ejecutar un empleado sobre la información se encuentran en la tabla TB_MANEJO_SUJETOS y estas son:

Acciones sobre la información	
Cve_Acción	**Acción**
1	Borrar
2	Renombrar
3	Cambiar de directorio
4	Almacenar
5	Respaldar
6	Crear
7	Acceder
8	Copiar
9	Modificar
10	Resguardar
11	Asignar
12	Administrar
13	Mantener
14	Generar
15	Preservar

Tabla 4.3. Acciones sobre la información.

Las acciones antes mencionadas tienen diversos sinónimos que no sería conveniente introducir como parte de las acciones con el objetivo de unificar las descripciones y evitar redundancias. Para tratar ese problema se creo una tabla de sinónimos (tabla 5.4) dentro de la base de datos, cuando el usuario selecciona una de las acciones anteriores a través de su clave se buscarán los sinónimos correspondientes y se mostrarán al usuario mediante una tabla informativa. El usuario

no podrá hacer nada con esa información, simplemente se mostrará para que tenga conocimiento de las posibilidades de una acción.

Sinónimos		
Clave Acción	**Acción**	**Sinónimo**
1	Borrar	Eliminar
1	Borrar	Quitar
2	Renombrar	Cambiar nombre
2	Renombrar	Modificar nombre
3	Cambiar de directorio	Modificar directorio
3	Cambiar de directorio	Cambiar de ruta de acceso
4	Almacenar	Guardar
5	Respaldar	Backups
6	Crear	Insertar
6	Crear	Agregar
7	Acceder	Ingresar
9	Modificar	Cambiar
10	Resguardar	Proteger
8	Copiar	Duplicar
11	Asignar	Dar
11	Asignar	Proporcionar

Tabla 4.4. Sinónimos de acciones que puede realizar un empleado a la información.

Analizando la figura 4.1, las políticas de seguridad se escriben con la ejecución de una acción partiendo de un componente dinámico hacia un estático. Sin embargo, la especificación de políticas estaría limitada para el caso en el cual la información tenga que ocupar el papel de sujeto.

Esto puede ocurrir para el caso en el que los administradores (o usuarios del prototipo) desearán escribir una política a partir de la información. Nosotros resolvimos el problema siguiendo la misma estructura, esquemáticamente la figura 3.2 y 3.3 muestran el tratamiento de este caso:

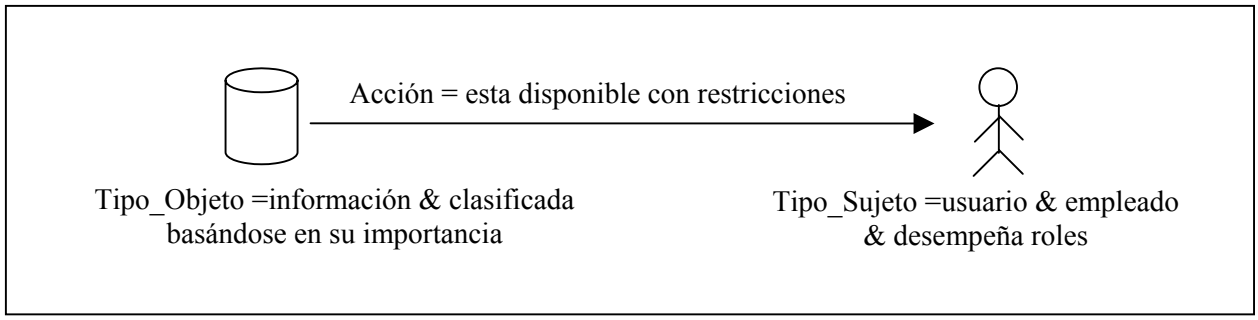


Fig. 4.2. Muestra los elementos de una política de seguridad, tomando como inicio de la descripción la información.

En la figura anterior se muestra esquemáticamente la forma en que se trabaja con la información como elemento de partida en la especificación de una política de seguridad y se sigue mostrando a la información como objeto (elemento estático). Sin embargo, la interfaz gráfica del prototipo para este caso mostrará la información como sujeto, (no objeto) y el usuario o mecanismo de acceso como objeto según sea el caso.

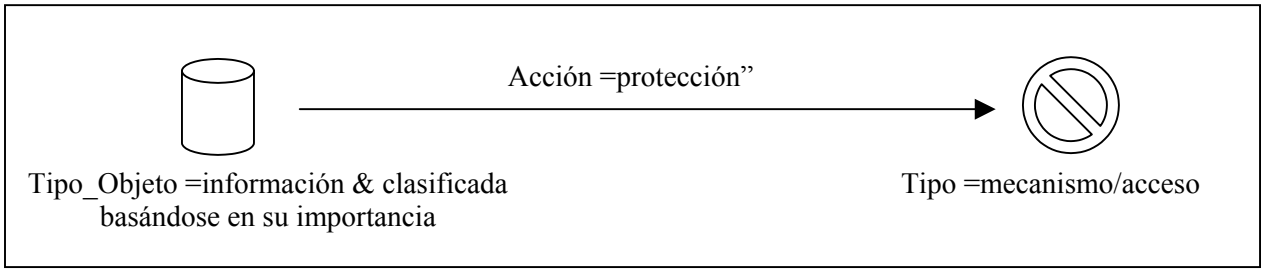


Fig. 4. 3. Se observan los elementos de una política de seguridad cuando se expresa la protección que debe tener la información.

Los esquemas anteriores muestran una forma de especificar políticas de seguridad que hagan referencia a las propiedades de protección de la información, así como la disposición que ésta tiene para ciertos empleados que ocupan roles específicos. Es importante mencionar que las gráficas que se muestran en las figuras 4.1, 4.2 y 4.3 no son las mismas que utiliza la interfaz gráfica, a continuación se muestran las gráficas utilizadas por la interfaz:

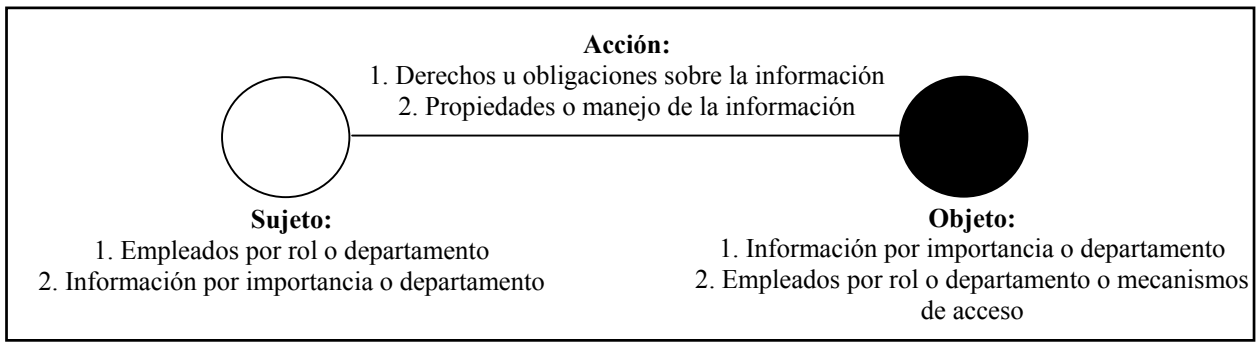


Fig. 4.4. Símbolos gráficos utilizados por la interfaz de usuario para representar una política de seguridad.

En la figura 4.4 se muestra en la parte inferior de cada nodo, los valores que toma cada uno de los elementos de la política y los números indican la correspondencia de valores, por ejemplo:

Al sujeto *1. Empleados por rol o departamento*, la acción y objeto que le corresponden son los etiquetados con el número uno; *1. Derechos y obligaciones sobre la información*, e *1. Información por importancia o departamento*.

4.3 CLASIFICACIÓN DE INFORMACIÓN

El prototipo trabaja mediante una base de datos que funciona como un motor de información sobre el que se apoya la interfaz gráfica. La base de datos incluye la información de los empleados involucrados en la seguridad de los archivos, los roles específicos que cada uno de ellos ocupa, el departamento al que pertenecen y una lista de la información –clasificada– con los privilegios de acceso establecidos por el administrador basándose en el rol que ocupa cada empleado involucrado.

Para asegurar una protección apropiada de la información corporativa, es importante clasificar de acuerdo a su importancia a los empleados y a la información. Clasificar a los empleados dependerá del administrador de seguridad y de las funciones que desempeñen. Mientras que la información se protege clasificándola de acuerdo a su sensibilidad, importancia crítica y valor organizacional, sin importar el medio en el que se almacena, ni los sistemas en los que se procesa o la forma en que se distribuye. La base de datos incluye como valores predeterminados la siguiente clasificación de información:

- Pública o general: Esta información es de distribución pública a través de canales autorizados de la corporación [2].

- Uso interno o importante: Información para uso exclusivo de empleados que dirigen los negocios de la corporación principalmente, pero no necesariamente, todo dependerá de la organización y administración de seguridad que se maneje [2].

- Confidencial: Información que si es publicada podría violar la privacidad organizacional, causando daños sustanciales a la corporación reduciendo su grado competitivo y afectando su desempeño [2].

Clasificando la información de la empresa es posible manipularla correcta y fácilmente disminuyendo los riesgos a los que está expuesta. En las siguientes secciones se presenta la información almacenada en la base de datos.

4.4 BASE DE DATOS

El objetivo de usar una base de datos es proporcionar flexibilidad al usuario en la descripción de las políticas de seguridad. Basándose en las necesidades de cada empresa el administrador de seguridad podrá realizar modificaciones a la información almacenada en ella. En la figura 4.4 se muestran las principales tablas utilizadas y la relación que tienen entre ellas.

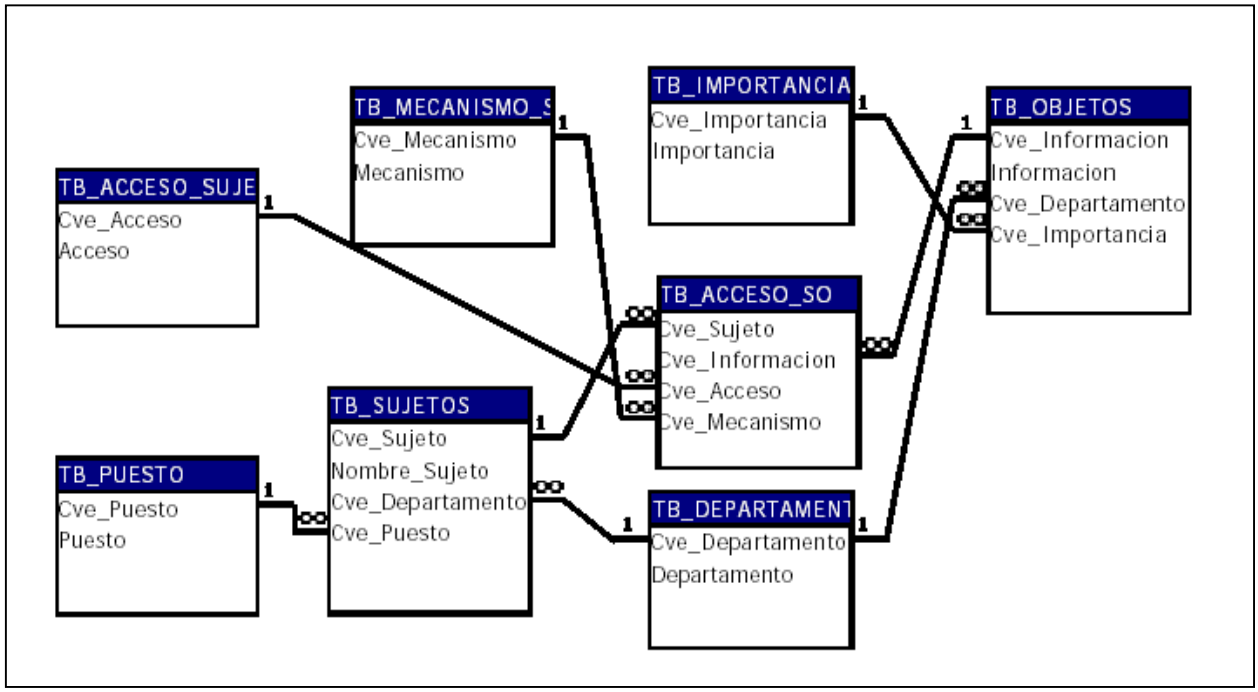


Fig. 4.5. Tablas principales de la base de datos y sus relaciones.

Las tablas de la base de datos forman parte necesaria en la construcción de una política de seguridad, para entender su importancia los siguientes sub-capítulos describen su función.

4.4.1 TB_SUJETOS

La tabla de sujetos le permite al administrador encargado de desarrollar las políticas de seguridad llevar el registro de los empleados que trabajan en la empresa y que tienen impacto sobre la seguridad de la información incluyendo el puesto que ocupen y el departamento al que pertenecen, para efectos de restringir sus accesos. Cuando el usuario en la interfaz gráfica seleccione la opción sujetos, si el tipo de sujeto es empleados la interfaz se conectará a la tabla *TB_SUJETOS* de la base de datos.

4.4.2 TB_ACCESO_SO

La tabla de sujetos se relaciona con la tabla de acceso a objetos *TB_ACCESO_SO}*, en la cual se enlistan los empleados y la información a la que tienen permitido accesar de acuerdo a su importancia –general, importante, confidencial–, su restricción –que puede ser acceso total, de lectura y restringido– y el mecanismo de acceso asignado a cada usuario necesario para preservar la seguridad en la información a la que tiene acceso.

4.4.3 TB_OBJETOS

La tabla de objetos contiene la información de la empresa, cada registro representa un elemento de información que incluye el departamento al que pertenece y la importancia que tiene para la organización –clasificación de la información de acuerdo a su importancia–. Cuando un usuario describe una política de seguridad específica, si la opción es objetos/información, la interfaz se conectará a la tabla *TB_OBJETOS* de la base de datos para que el usuario seleccione la opción deseada.

Las tablas *TB_DEPARTAMENTO, TB_PUESTO, TB_IMPORTANCIA, TB_ACCESO_SUJETOS, TB_MECANISMO_SEG* son catálogos de información necesarios para evitar redundancias de datos en la base de datos sin consumir recursos de memoria extra. Estos catálogos son necesarios porque incluyen los departamentos de la empresa, los puestos existentes, la clasificación de la información de acuerdo a su importancia, los tipos de acceso a la información, y los mecanismos de acceso a la información.

La interfaz gráfica requiere conectarse a la base de datos para ir agregando los datos necesarios para construir una política de seguridad particular mediante el uso de sub-ventanas.

El objetivo de la base de datos no es el de representar una política de seguridad, sino proporcionar al usuario información que requiere para la construcción de una política de seguridad…

Ejemplos de políticas particulares:

1. El administrador de seguridad debe otorgar contraseñas a cualquier empleado de la organización.
2. Todos los empleados del departamento de contabilidad deben tener acceso total a la nómina mediante Contraseñas.
3. Todos los empleados del departamento de sistemas deben respaldar los archivos locales.
4. Todos los programadores del departamento de sistemas no pueden otorgar acceso total a la información importante.
5. Todos los gerentes de la empresa deben tener acceso de lectura a la información importante mediante contraseñas.
6. Toda la información confidencial debe protegerse mediante contraseñas.
7. Toda la información general debe estar disponible a cualquier empleado de la organización.
8. Toda la información importante debe estar protegida mediante un mecanismo de acceso.

4.5 CAPTURA DE POLÍTICAS DE SEGURIDAD MEDIANTE LA INTERFAZ GRÁFICA

En este sub-capítulo mostraremos como se capturan las políticas de seguridad en el prototipo administrador (objetivo de esta tesis) mediante un ejemplo, la política es la siguiente:

Todos los empleados de la empresa deben acceder a sus archivos locales mediante passwords.

4.5.1 PANTALLA PRINCIPAL

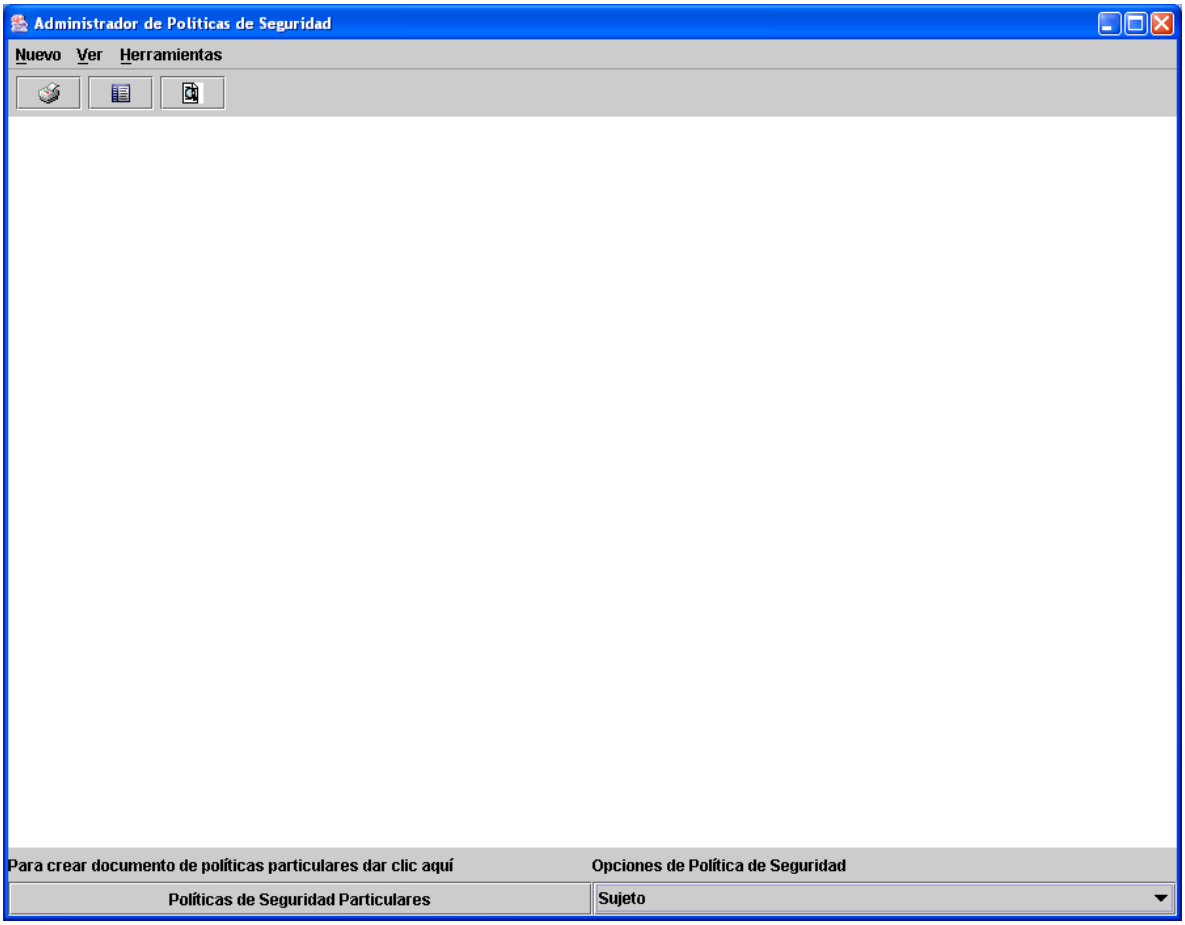


Fig. 4.5. Pantalla principal de la interfaz gráfica (prototipo administrador de políticas de seguridad).

La pantalla principal del prototipo como se muestra en la figura 4.5 se forma de tres secciones, la primera es la barra de menú con las siguientes opciones:

Nuevo	**Ver**	**Herramientas**

• Empleados
• Información
• Catálogos
  ▪ Departamentos
  ▪ Puestos
  ▪ Mecanismos de acceso
  ▪ Niveles de información
• Salir
	• Empleados
	• Información
	• Catálogos
	  ▪ Departamentos
	  ▪ Puestos
	  ▪ Mecanismos de acceso
	  ▪ Niveles de información
	• Documentos
	  ▪ Políticas particulares
	  ▪ Políticas generales
		• Generar
		  ▪ Políticas particulares
		  ▪ Políticas generales
		• Eliminar
		  ▪ Políticas particulares
		  ▪ Políticas administrador

El menú *nuevo* ofrece al usuario la posibilidad de dar de alta empleados con sus respectivos niveles de acceso a la información, información de la empresa en base a su importancia y los diferentes catálogos de información que incluyen departamentos y roles de la empresa, mecanismos de acceso implantados y niveles de información establecidos.

El menú *ver* presenta los datos almacenados en las principales tablas de la base de datos como mecanismo informativo evitando que el usuario acceda a la base. Por último el menú herramientas nos ofrece opciones para generar o eliminar políticas de seguridad.

4.5.2 DESCRIPCIÓN DEL SUJETO

En la barra inferior de la interfaz del lado derecho se encuentran las opciones de los elementos de una política de seguridad específica: sujeto, objeto, acciones…como se muestra en la figura 4.6. El administrador deberá seleccionar la opción deseada.

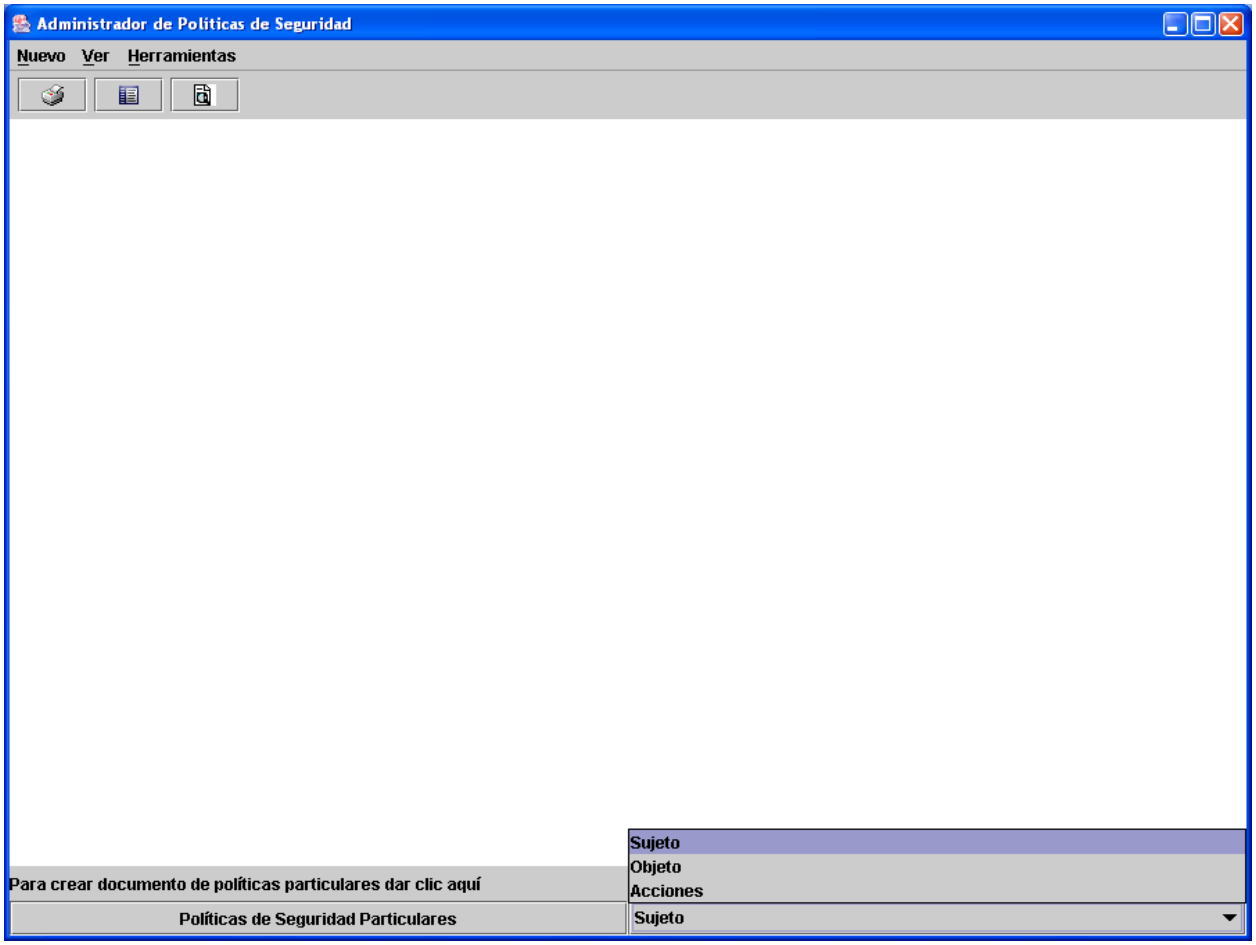


Fig. 4.6. Barra inferior de la interfaz gráfica.

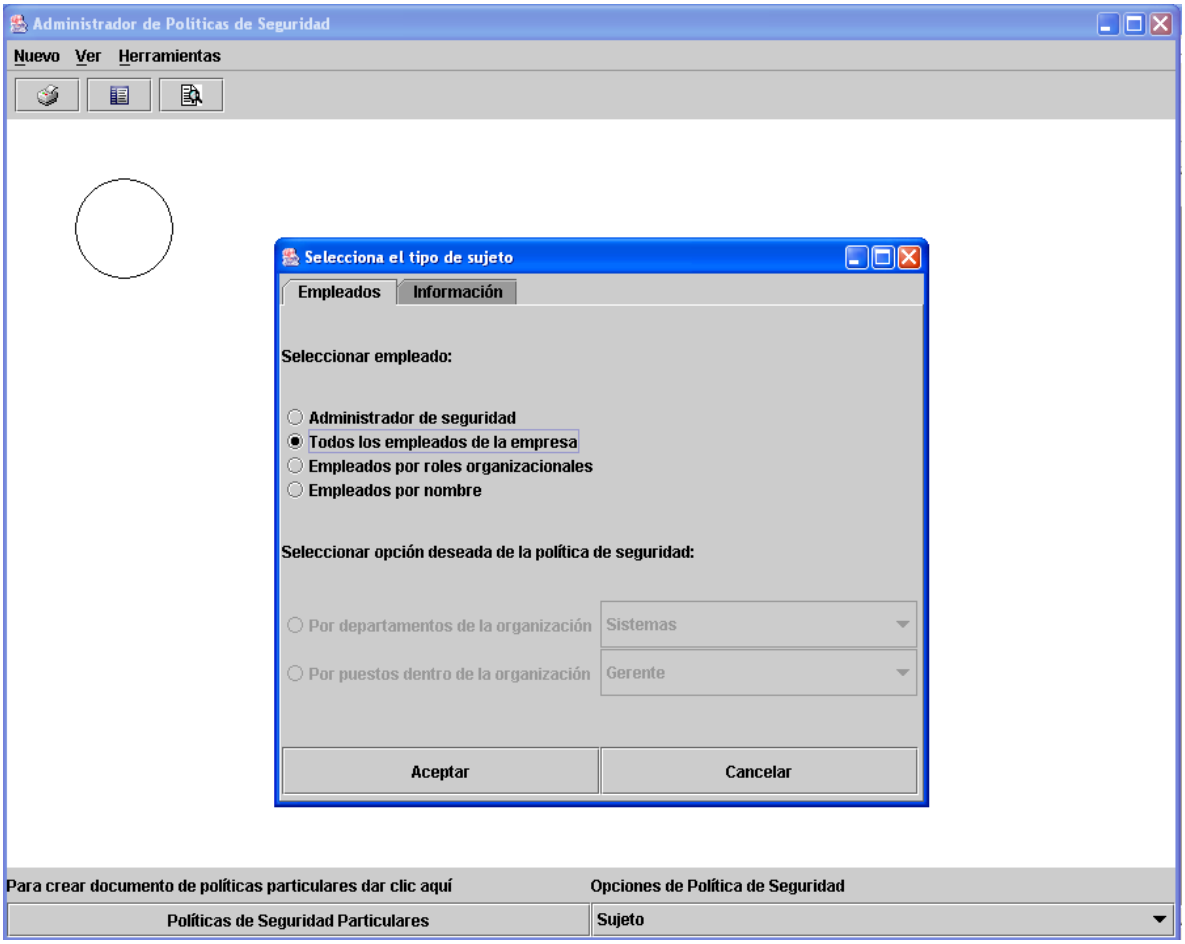


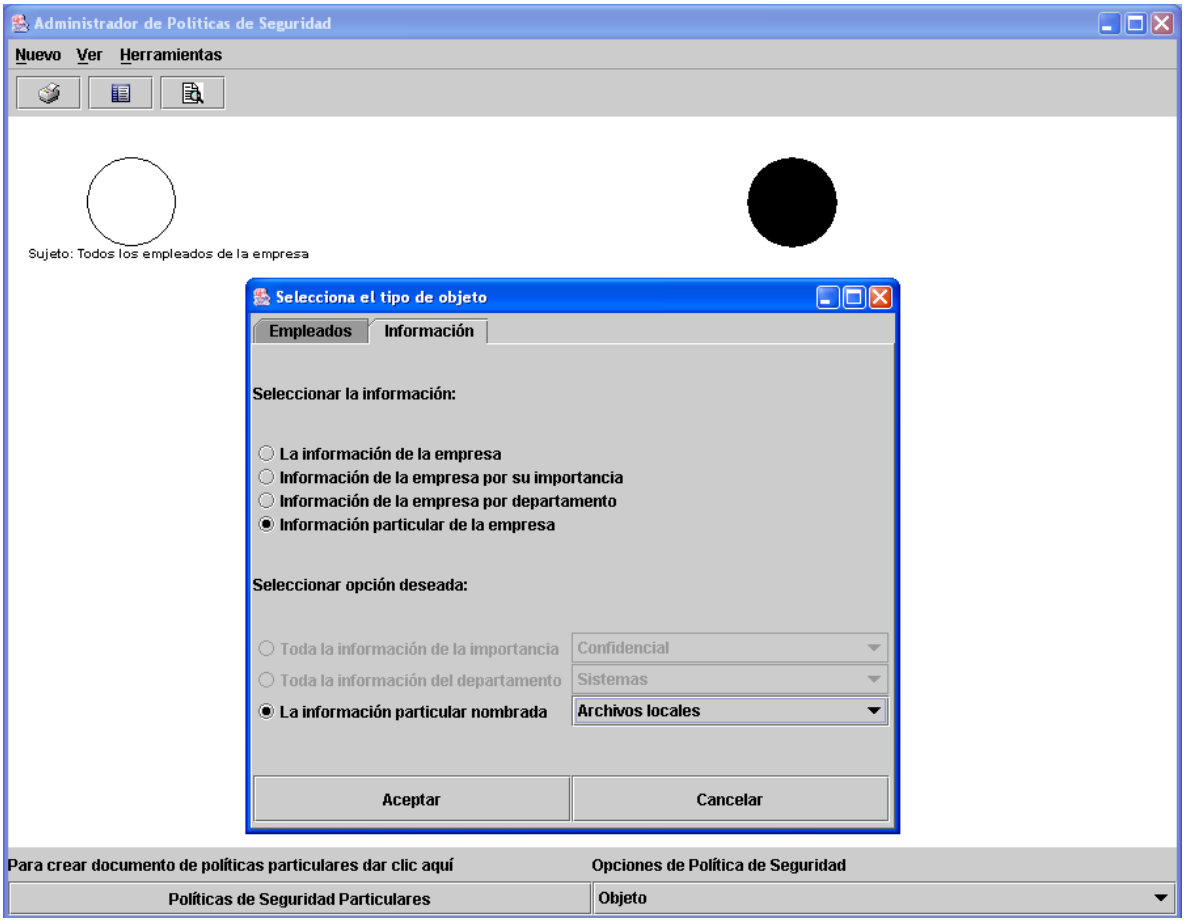
Fig. 4.7. Descripción del sujeto de la política.

Una vez seleccionada la opción *sujeto*, el usuario arrastrando el ratón dibujará el nodo correspondiente al elemento seleccionado y se abrirá una subventana en la que el usuario deberá seleccionar el valor que desea aplicar al primer elemento de la política (sujeto).

En la figura 4.7 el usuario seleccionó la opción *todos los empleados de la empresa*, este enunciado se insertará en la tabla *políticas* de la base de datos en la columna *sujetos*. De esta forma queda descrita la primera parte de una política de seguridad.

4.5.3 DESCRIPCIÓN DEL SUJETO

El segundo elemento de una política de seguridad es el objeto sobre el cual se ejecuta una acción, para insertar el valor de este elemento la interfaz gráfica como ya lo vimos en secciones anteriores se conectará a la tabla objetos de la base de datos. A continuación se muestran las imágenes respectivas para su descripción.



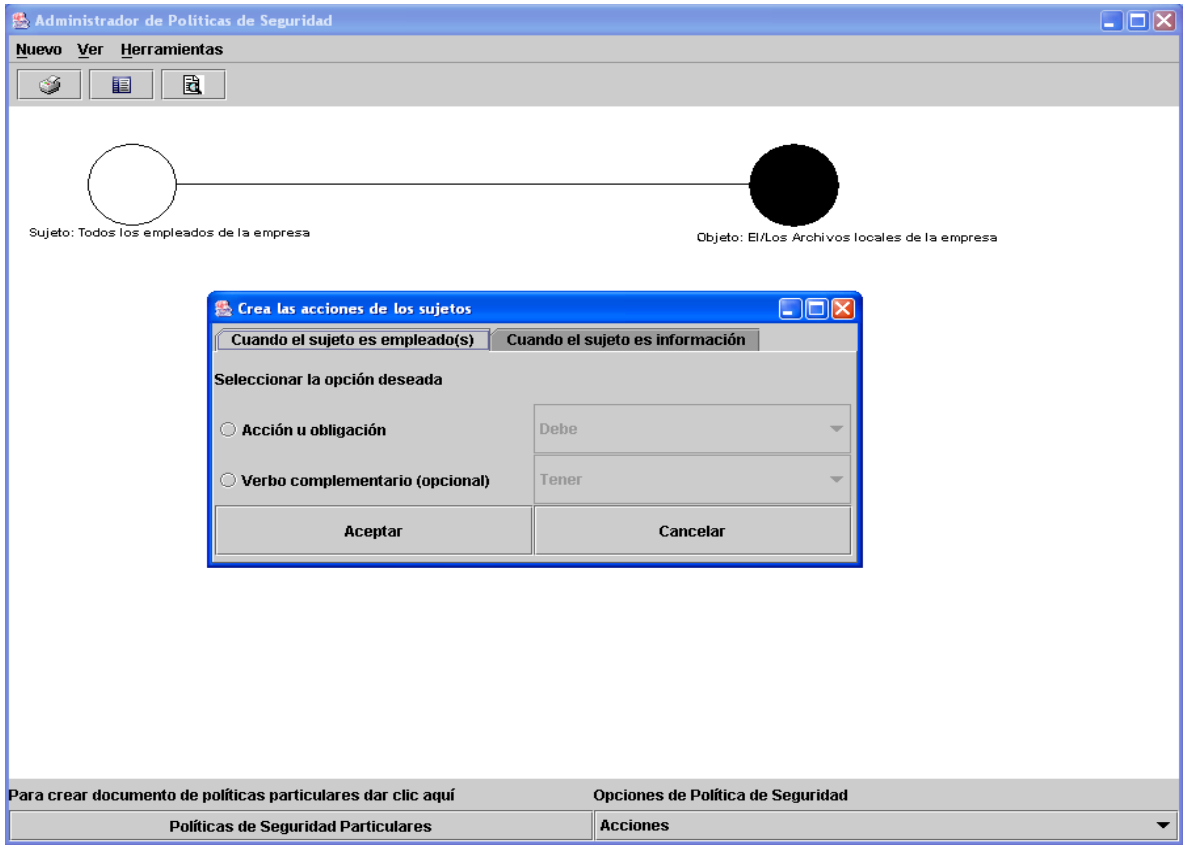
4.8. Descripción del objeto de la política.

Los valores que el objeto puede tomar son información u empleados para lo cual se muestran en la subventana unas pestañas que deberán seleccionarse por el usuario para agregar el valor deseado.

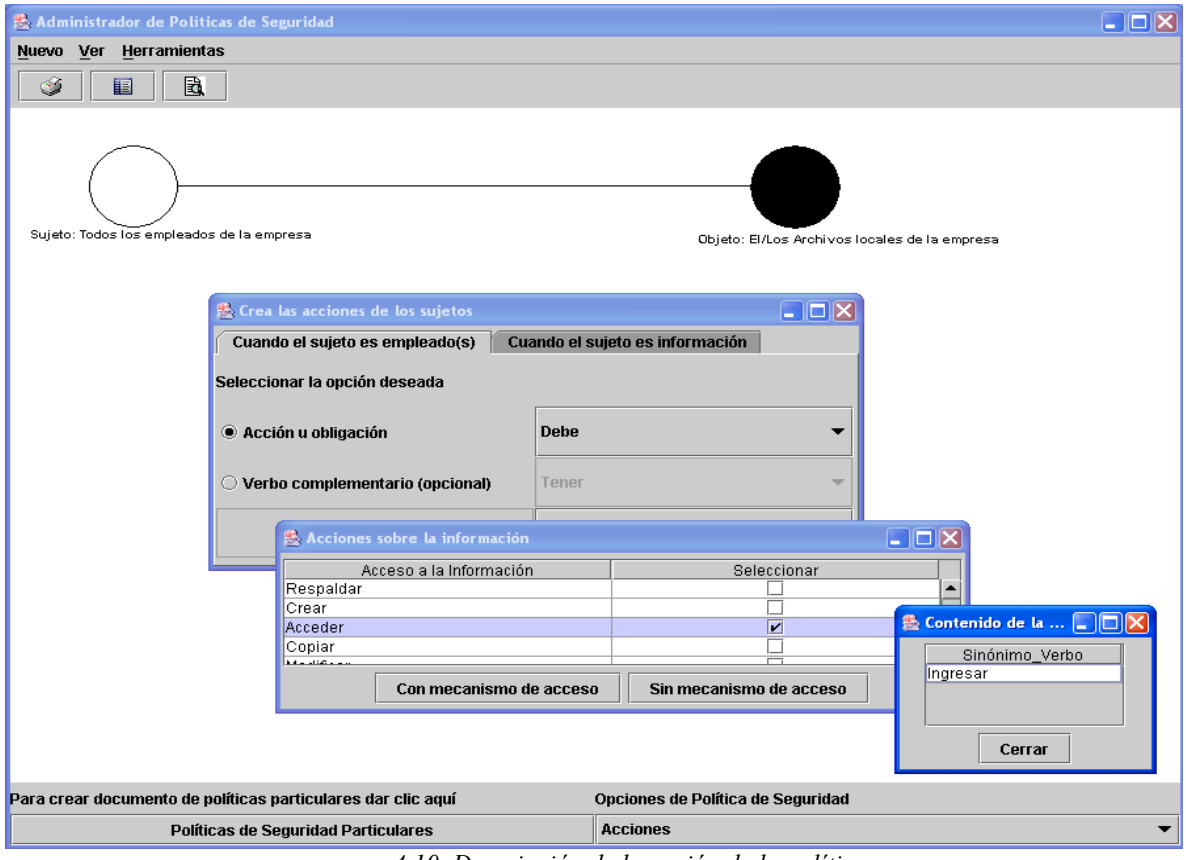
De la misma forma que el usuario creo el sujeto, lo hará para el objeto y las acciones de la política de seguridad. El valor que puede tomar el objeto cuando el sujeto es empleados es información clasificada por su importancia, información clasificada por el departamento al que pertenece o información especifica. En la figura 4.8 el valor seleccionado fueron los *archivos locales*.

4.5.4 DESCRIPCIÓN DEL SUJETO

Como última fase en la descripción de políticas de seguridad específicas, se encuentra la definición de la acción que puede hacer o no el sujeto sobre el objeto. A continuación se muestran las ventanas de la información que puede capturarse cuando el sujeto es empleados.



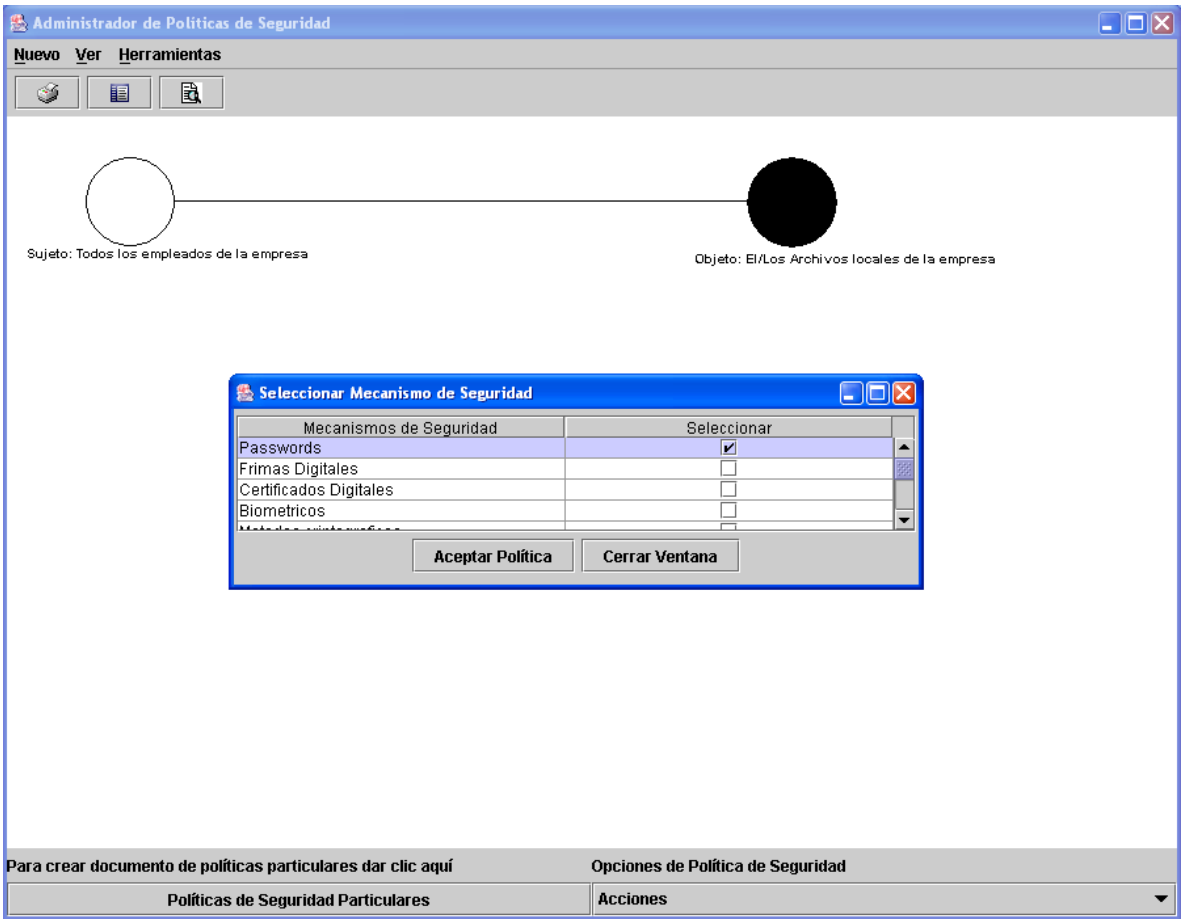
4.9. Descripción de la acción de la política.



4.10. Descripción de la acción de la política.

Las figuras 4.9 y 4.10 representan gráficamente la forma en que trabaja la interfaz en la descripción de acciones de una política de seguridad. El valor seleccionado para ejemplificar es:
…debe acceder…

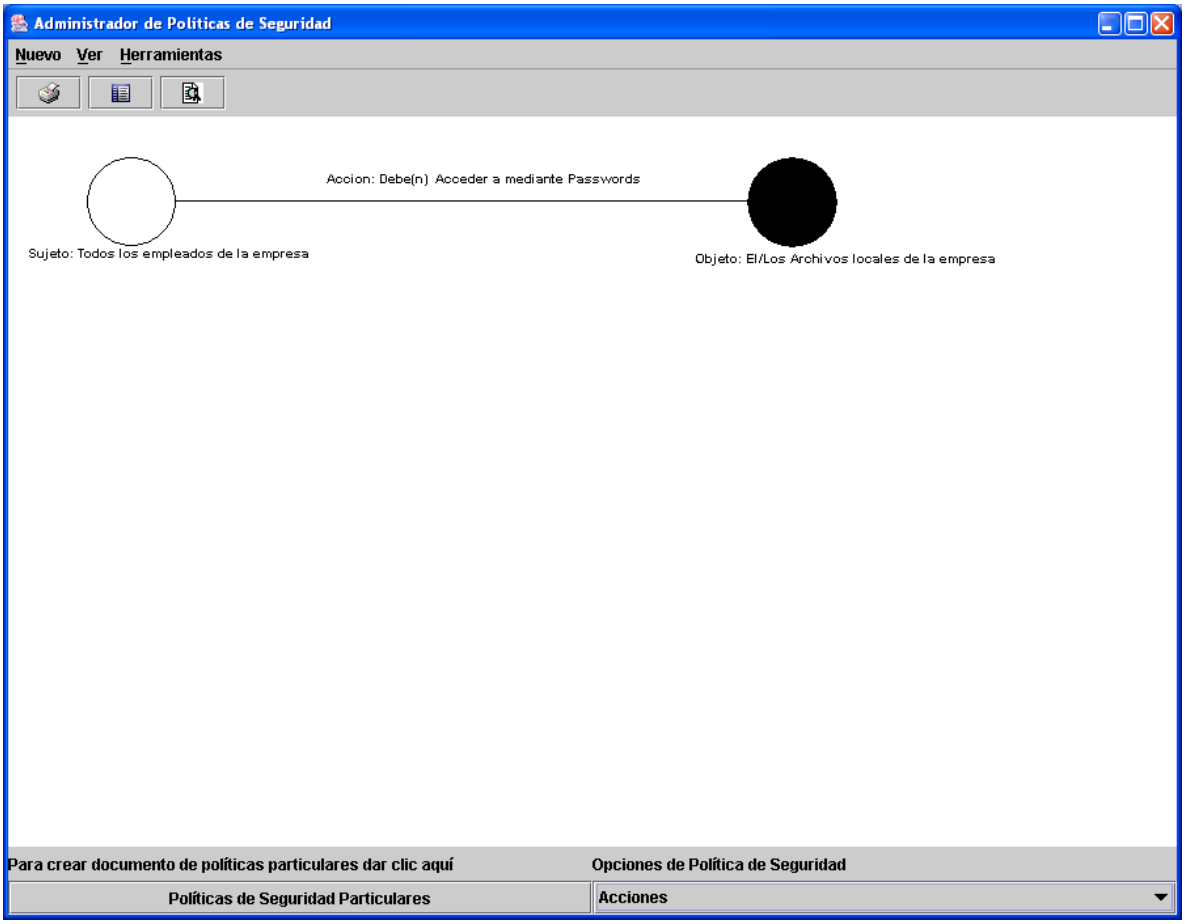
Como parte del elemento acción se encuentra el mecanismo de acceso requerido en algunas políticas de seguridad, la siguiente imagen representa un ejemplo en el que se selecciona *passwords* como complemento de la acción quedando la descripción:
…debe acceder mediante passwords…



4.11. Descripción del mecanismo de acceso requerido por la acción.

Las posibles acciones pueden tomar dos caminos: acceso a la información y manejo de la información. Cuando hablamos de *acceso a la información*, los datos que se presentan son: *acceso total, acceso de lectura, acceso restringido*. Para el caso de la opción *manejo de la información*, los datos que se presentan son los elementos vistos en la tabla 4.3.

La figura 4.12 finaliza el ejemplo mostrando la vista de la política en la interfaz gráfica: *Todos los empleados de la empresa deben acceder a sus archivos locales mediante passwords.*



4.12. Interfaz gráfica. Descripción de una política de seguridad.

4.6 CONCLUSIONES

El uso de una interfaz gráfica capaz de facilitar la captura de políticas de seguridad representa un reto importante en la preservación de la seguridad de los recursos informáticos de una empresa.

En las secciones anteriores se mostró una herramienta gráfica diseñada para facilitar y mejorar la captura de políticas de seguridad. Si bien esta herramienta muestra una serie de limitantes como son: el enfoque exclusivo a la protección de la información, y la forma en que una política es capturada (la cual podría no ser la mejor solución al problema de uso fácil), es un hecho que el primer trabajo siempre abre nuevas ideas y oportunidades y esta tesis constituye las bases para nuevas aportaciones que ofrezcan una mejor solución en el diseño de normas de seguridad.

5 FORMALIZACIÓN DE POLÍTICAS DE SEGURIDAD

5.1 INTRODUCCIÓN

En capítulos anteriores hemos hablado de la importancia de especificar políticas de seguridad sobre información, las cuáles además de contribuir al establecimiento de metas de seguridad para proteger los recursos, no deben obstaculizar la misión de la organización para la que fueron desarrolladas. Para que ello no suceda es importante validar que estas políticas no se contradigan unas con otras.

La solución adoptada en esta tesis aboga el uso de formalismos que consiste en expresar cada una de las políticas en términos de especificación formal y cuya interpretación será validada por medio del demostrador de teoremas de primer orden conocido como *Otter*.

Como se mencionó en el capítulo anterior, la interfaz gráfica escribe en un archivo de texto las políticas de seguridad que el administrador introduce. En la capa interna del prototipo, se ha desarrollado un programa que toma como entrada el archivo de políticas y el cual traslada a sintaxis del demostrador de teoremas que se guarda en otro archivo de texto. A continuación, extendemos el tema de los demostradores de teoremas y el trabajo desarrollado para el prototipo.

En la siguiente sección se estudia el demostrador de teoremas utilizado para formalizar y validar las políticas de seguridad. Si el lector tiene conocimientos generales de lógica de primer orden y de las principales características del demostrador de teoremas *Otter* tiene la opción de omitir la lectura de dicha sección.

5.2 DEMOSTRADOR DE TEOREMAS DE PRIMER ORDEN

La lógica de primer orden es en la actualidad una de las herramientas más importantes para la solución de problemas. Su enfoque corresponde a lo que se ha denominado como la *demostración automática de teoremas* que se define como, los métodos que prueban por refutación que un conjunto de cláusulas es insatisfacible. Los sistemas demostradores de teoremas trabajan en la búsqueda de una prueba que satisfaga o no una fórmula dada. La sintaxis de la lógica de primer orden requiere del estudio de las siguientes definiciones:

Definición 1. La cadena □ es un término si

a. □ es una constante,
b. □ es una variable,
c. □ tiene la forma ß(□1, □2,…, □n) donde ß es una función y cada uno de los □i (i □ {1, 2,…,n}) es un término. [10]

Definición 2. La cadena Ø es una fórmula de *L* si:

a. Ø tiene la forma □(□1, □2, …, □n), donde □ es una relación de n-términos y cada uno de los □i son un término (llamado fórmulas atómicas),
b. Ø tiene una de las siguientes formas: ¬□; (□ ∧ □); (□ ∨ □); (□ ⇒ □); (□ ⇔ □) donde □ y □ son fórmulas recursivamente,
c. Ø tiene la forma ∃μ□ o ∀μ□, donde μ es una variable y □ es una fórmula. (Este es el tipo de fórmulas que se ocupan en el prototipo de políticas de seguridad). [10]

Definición 3. La ocurrencia de una variable μ dentro de una fórmula Ø está acotada si se encuentra dentro de una ocurrencia en Ø de la forma ∃μ□ o ∀μ□. Una ocurrencia que no está acotada se conoce como libre. Por ejemplo, se tiene una fórmula como (femenino(x) ∨ ∃x masculino(x)), la primera ocurrencia de x es libre, y la segunda está acotada. [10]

Definición 4. Una fórmula Ø es cerrada si cada ocurrencia de una variable en Ø está acotada. Como contraparte, la fórmula Ø es abierta en las variables que aparecen libres. [10]

Existen varios demostradores de teoremas en lógica de primer orden como son: Otter, Spass, Vampire, Waldmeister y Meteor, todos ellos con evidencias suficientes para ser considerados como eficientes. Sin embargo, se eligió el demostrador de teoremas cuya documentación fuese más extensa y tuviese más pruebas realizadas y éste es *Otter*.

Otter es la cuarta generación de los sistemas de deducción del Laboratorio Nacional de Argonne. Este demostrador está codificado en ANSI C y tiene la característica de portabilidad, puede usarse en sistemas Unix y Windows.

El funcionamiento general de *Otter* [11] es el siguiente:

1. Las reglas de inferencia de *Otter* toman un pequeño conjunto de cláusulas[2] e infieren una cláusula, si la cláusula inferida es nueva y puede usarse, ésta es almacenada y podrá estar disponible para inferencias subsecuentes. Los métodos utilizados por *Otter* para aplicar las reglas de inferencia son resolución binaria, hyper-resolución, UR-Resolución y Paramodulación binaria.

2. Otter permite al usuario seleccionar los métodos que desee utilizar en la prueba y las cuales se especifican al inicio de las cláusulas de entrada. La palabra reservada para dicha tarea es *set(regla_inferencia)*. Esta palabra va acompañada del identificador de la regla de inferencia y los valores que puede tomar son: *binary_res, hyper_res, neg_hyper_res, ur_res, para_into, para_from, demod_inf* y *auto*. La bandera *auto* determina las reglas de inferencia y la estrategia de búsqueda utilizadas por Otter en la prueba, es decir, Otter seleccionará de entre todos los métodos que maneja, el que mejor se acomode en la demostración de teoremas.

Nota: Es recomendable utilizar el modo automático (auto) que ofrece Otter, principalmente cuando el usuario es inexperto en el manejo de este demostrador. Por otro lado, cada uno de los métodos para aplicar las reglas de inferencia requiere que las cláusulas involucradas en la búsqueda de una prueba cumplan ciertos parámetros. Por ejemplo, para aplicar UR- Resolución se requiere que una de las cláusulas involucradas en la prueba sea no unitaria y las otras unitarias[3].

[2] Definición de cláusula: es una disyunción de literales ($P \vee Q \vee R$). Una literal es una fórmula atómica o la negación de una fórmula atómica.
[3] Cláusula unitaria: es aquella que solamente tiene una literal.

3. Los enunciados del problema pueden ser capturados como fórmulas de primer orden o como cláusulas. Si se ingresan fórmulas de primer orden, *Otter* transforma éstas a cláusulas.

4. El demostrador en una de sus capacidades re-escribe y simplifica cláusulas inferidas con un conjunto de igualdades, y también usa una igualdad inferida para re-escribir todas las cláusulas existentes.

5. *Otter* trabaja con clasificación hacia delante y hacia atrás, cuando se trata de la primera el demostrador borra una cláusula inferida si ésta es clasificada por cualquier cláusula existente, y con clasificación hacia atrás borra todas las cláusulas que son clasificadas por una cláusula inferida.

La sintaxis de *Otter*, utilizada en el prototipo es la siguiente:

LPO	*Otter*	Descripción
$\wedge$	&	And, unión, conjunción
$\vee$	|	Or, disyunción
$\neg$	-	Negación
$\Rightarrow$	->	Implicación
$\Leftrightarrow$	<->	Doble implicación
$\exists$	exists	Cuantificador, existencial
$\forall$	all	Cuantificador, para todo

Tabla 5.1. Simbología principal de lógica de primer orden y *otter*.

5.3 FORMALIZACIÓN DE POLÍTICAS DE SEGURIDAD

Nuestro lenguaje de primer orden para expresar las políticas de seguridad formalmente contiene los siguientes símbolos:

SÍMBOLOS	
Símbolo	**Descripción**
Símbolos de un solo término	
información(x)	*x* es información
empleado(y)	*y* es un empleado
* Empleado_XXX(y)	*y* es un empleado de cierto departamento donde *XXX* es el identificador del departamento (ej. Empleado_Sistemas(y)).
* Rol(x)	*y* es un empleado con cierto rol (ej. Gerente(y)).
* Rol_XXX(y)	*y* es un empleado de cierto rol que pertenece a un departamento donde *XXX* es el identificador del departamento (ej. Gerente_Sistemas(y)).
* Informacion_XXX(x)	*x* es información de cierto departamento donde *XXX* es el identificador del departamento (ej. Informacion_Sistemas(x)).
mecanismo_acceso(s)	*s* es un mecanismo de acceso
privilegio_acceso(a)	*a* es un privilegio de acceso
* Informacion_XXX(x)	*x* es un nivel de importancia de la información donde *XXX* es el identificador de la importancia (ej. Informacion_General(x)).
caracteristica(w)	*w* es una característica de la información (ej. Integridad(w)).
Símbolos de dos términos	
disponible(x,y)	*x* está disponible al empleado *y*
Posee(y,s)	*y* posee el mecanismo de acceso *s*
protege(y,x)	*y* protege la información *x*
autoriza(y1, y)	*y1* autoriza a *y* para realizar cierta actividad
dana(y,x)	*y* daña la información *x*
Símbolos de tres y más términos	
provee(y, x, s)	El empleado *y* provee al empleado *x* el mecanismo de acceso *s*
ad_preserva(y,w,x)	El empleado *y* preserva la propiedad *w* de la información *x*
ad_otorga(y1,p,y)	El empleado *y1* otorga el mecanismo de acceso *p* al empleado *y*

*Tabla 5.2. Simbología general en la formalización de políticas de seguridad. Las celdas marcadas por un * representan un esquema, no es lógica de primer orden.*

Los símbolos de dos términos se obtienen de la tabla *TB_MANEJO_SUJETOS* de la base de datos que contiene una serie de acciones que puede ejecutar un empleado sobre la información de la empresa.

En la tabla 5.3 se muestran los símbolos utilizados para formalizar estas acciones.

Símbolos de dos términos	
Símbolo	**Descripción**
Borrar(y, x)	El empleado *y* borra la información *x*
Renombrar(y, x)	El empleado *y* renombra la información *x*
Cambiar_de_directorio(y, x)	El empleado *y* cambia de directorio la información *x*
Almacenar(y, x)	El empleado *y* almacena la información *x*
Respaldar(y, x)	El empleado *y* respalda la información *x*
Crear(y, x)	El empleado *y* crea la información *x*
Acceder(y, x)	El empleado *y* accede a la información *x*
Copiar(y, x)	El empleado *y* copia la información *x*
Modificar(y, x)	El empleado *y* modifica la información *x*
Resguardar(y, x)	El empleado *y* resguarda la información *x*
Asignar(y, x)	El empleado *y* asigna la información *x*
Administrar(y, x)	El empleado *y* administra la información *x*
Mantener(y, x)	El empleado *y* mantiene la información *x*
Generar(y, x)	El empleado *y* genera la información *x*
Preservar(y, x)	El empleado *y* preserva la información *x*
Acceso_Total(y, x)	El empleado *y* tiene acceso total a la información *x*
Acceso_Lectura(y, x)	El empleado *y* tiene acceso de lectura a la información *x*
Acceso_Restringido(y, x)	El empleado *y* tiene acceso restringido a la información *x*

Tabla 5.3. Simbología de dos términos para representar acciones sobre la información.

Para la formalización de las políticas de seguridad el prototipo administrador se conecta a la base de datos donde encontramos una tabla llamada POLITICAS con una columna para cada elemento

de la construcción de una política de seguridad (sujeto, objeto, acción, y mecanismo de seguridad). El funcionamiento de la formalización es mediante un programa analizador escrito en java que obtiene cada uno de esos elementos y los traduce a sintaxis del demostrador de teoremas *Otter* y los almacena en un archivo de salida para posteriormente enviarlo al demostrador. El prototipo se conecta al demostrador mediante una herramienta gráfica para evitar trabajar en línea de comandos de MS-DOS. A continuación se estudia la formalización de políticas de seguridad generales y particulares.

5.3.1 FORMALIZACIÓN DE POLÍTICAS DE SEGURIDAD GENERALES

Las políticas generales de seguridad que protegen la integridad, confidencialidad, disponibilidad y autenticación como ya vimos en el capítulo anterior quedan sentadas en el documento como el prototipo las describe, es decir son fijas, por lo tanto su formalización también es fija y una vez que el usuario selecciona las políticas deseadas, éstas se almacenan en una tabla de la base de datos llamada, POLITICAS_GRALES. Cuando el usuario quiere hacer la validación formal del documento el programa analizador escrito en java se conectará a la base de datos y va a extraer las políticas elegidas por el usuario quedando su formalización de la siguiente forma:

1. Toda información propiedad de *la empresa* deberá estar protegida adecuadamente en cuanto a su integridad, confidencialidad, disponibilidad y autenticación.

all y all x (empleado(y) & informacion(x) & protege(y,x) ->
(exists a (Integridad(a) & preserva(x,a)))
| (exists b (Confidencialidad(b) & preserva(x,b)))
| (exists c (Disponibilidad(c) & preserva(x,c)))
| (exists d (Autenticacion(d) & preserva(x,d)))).

Integridad

2. Todo daño deliberado, robo o modificación no autorizada de la información propiedad de *la empresa* deberá ser sancionado.

3. Todo daño causado por negligencia a la información propiedad de *la empresa*, deberá ser sancionado.

all x all y (empleado(y) & informacion(x) & dana(y,x) ->
(exists y1 (empleado(y1) & sanciona(y1,y)))).

Nota: La política 2 y 3 tienen la misma formalización.

El símbolo *dana(y,x)* que significa que el empleado *"y"* daña la información *"x"*, se ocupa en los siguientes casos:

- daño deliberado o por negligencia
- eliminación o modificación de la información sin autorización
- eliminación o modificación de la información con autorización pero sin propósito (o accidentalmente).

4. El administrador de seguridad es responsable de proveer un ambiente seguro en el cual la información pueda ser mantenida con integridad.

all y all x (Administrador_seguridad(y) & informacion(x) ->
ad_preserva(y, Integridad, x) &
ad_preserva(y, Disponibilidad, x) &
ad_preserva(y, Confidencialidad, x)).

Confidencialidad

5. Toda información propiedad de la empresa deberá ser generada, eliminada o modificada únicamente por aquellas personas autorizadas para hacerlo y en aquellos lugares autorizados por *la empresa*.

all y all x (empleado(y) & informacion(x) & Generar(y, x) ->
exists z (Administrador_seguridad(z) & autoriza(z, y, crear))).

all y all x (empleado(y) & informacion(x) & Borrar(y, x) ->
exists z (Administrador_seguridad(z) & autoriza(z, y, eliminar))).

all y all x (empleado(y) & informacion(x) & Modificar(y, x) ->
exists z (Administrador_seguridad(z) & autoriza(z, y, cambiar))).

6. Toda información propiedad de *la empresa* debe clasificarse de acuerdo a su nivel de importancia para el negocio.

all x (informacion(x) -> (exists z (importancia(z) & clasifica(x,z)))).

7. El acceso a la información de la compañía se restringe solamente a usuarios autorizados por el administrador de seguridad.

all y all x (empleado(y) & informacion(x) & Acceder(y, x) ->
exists z (Administrador_seguridad(z) & autoriza(z, y, ingresar))).

Disponibilidad

8. Toda información propiedad de *la empresa* deberá estar disponible sólo a aquellas personas a quienes está destinada.

all x all y (informacion(x) & empleado(y) & Manipular(y, x) -> disponible(x, y)).
all y all x (Acceder(y, x) | Generar(y, x) | Borrar(y, x) | Modificar(y, x) -> Manipular(y, x)).

Autentificación

9. El administrador de seguridad es responsable de proveer un mecanismo de acceso a los usuarios autorizados para manipular la información de la empresa.

all y all x (empleado(y) & informacion(x) & Manipular(y, x) ->
(exists z (Administrador_seguridad(z) &
(exists s (mecanismo_acceso(s) & provee(z, y, s))))).

5.3.2 FORMALIZACIÓN DE POLÍTICAS DE SEGURIDAD ESPECÍFICAS

Debido a que las políticas de seguridad particulares siguen una estructura de sujeto, acción y objeto, la formalización de éstas resulta más fácil. Esta es una de las principales ventajas que se observan en la descripción de políticas siguiendo un esquema de desarrollo.

La primera parte de la política (sujeto) puede consistir ya sea de los empleados de la empresa, del administrador de seguridad que no se incluye en la base de datos de empleados (se maneja como un elemento diferente por sus obligaciones). Por último se da el caso para el cual la información ocupa la primera parte de la política.

Un empleado es identificado por su rol, por el departamento al que pertenece o ambos, de esta forma solo da opción a que la formalización de la primera parte de la política cuando el sujeto es un empleado, se forme con los siguientes símbolos:

1. *all y (empleado(y) -> ...) o exists y (empleado(y) & ...)*

2. *all y (Empleado_XXX (y) ->...) o exists y (Empleado_XXX (y) &...), donde XXX es un departamento. Ejemplo " all y (Empleado_Sistemas(y) -> ...)"*

3. *all y (Rol(y) -> ...) o exists y (Rol(y) & ...). Ejemplo " all y (Gerente(y) -> ...)"*

4. *all y (Rol_XXX(y) -> ...) o exists y(Rol_XXX(y) & ...), donde XXX es un departamento. Ejemplo "all y (Gerente_Sistemas(y) -> ...)"*

Los valores departamento y rol se obtienen de los datos obtenidos de la base de datos cuya información se localiza en la tabla de departamentos y puestos.

Para el caso en el que la primera parte de la política se forme por el administrador de seguridad, o custodio de la información (como lo manejan diferentes autores de seguridad), la formalización queda de la siguiente forma:

all y (Administrador_seguridad(y) -> ...) o *exists y (Administrador_seguridad(y) & ...)*

Finalizando la primera parte de la formalización cuando se refiere a información, ésta puede identificarse por su nivel de importancia, departamento al que pertenece, o por su identidad dentro de la empresa:

1. *all x (informacion(x) &…) o exists x (informacion(x) &…)*

2. *all x (Informacion_XXX(x) & …) o exists x (Informacion_XXX(x) &…), donde XXX es un nivel de importancia. Ejemplo "all x (Informacion_Confidencial(x) & …)"*

3. *all x (Informacion_XXX(x) &…) o exists x (Informacion_XXX(x) &…), donde XXX es un departamento. Ejemplo "all x (Informacion_Sistemas(x) & …)"*

4. *all x (Informacion_Particular(x) & …) o exists x (Informacion_Particular(x) & …). Ejemplo "all x (Nomina(x) & …)"*

Nota: De esta forma la programación resultó sencilla, mediante el uso de sentencias recursivas (for, if then else, while).

Para la formalización de las políticas no se sigue el mismo orden utilizado en la descripción formada por la interfaz gráfica, pero los elementos son los mismos (sujeto, objeto y acción); una vez que se formalizó la primera etapa de la política (sujeto), la segunda etapa (objeto) es similar a la primera, ya que los elementos que la conforman pueden ser la información, los empleados de la empresa o el administrador de seguridad.

La siguiente fase en la formalización incluye las acciones que el sujeto realiza sobre el objeto y los símbolos utilizados para representar formalmente dichas acciones se muestran en el cuadro 5.3. Como último elemento en formalización de políticas de seguridad específicas se encuentra el mecanismo de acceso necesario para que el sujeto realice cierta acción sobre el objeto.

(… exists s (mecanismo_acceso(s) & posee(y,s) …), donde y es un empleado.

El mecanismo de acceso puede ser cualquiera de los elementos que se encuentran en la tabla correspondiente de la base de datos (TB_MECANISMOS_SEG) ej.

(… exists s (Passwords(s) & posee(y,s) …), donde y es un empleado.

Una vez formalizadas cada una de las partes que forman una política de seguridad, se tiene cada uno de los elementos en términos de lógica de primer orden, para que éstas sean probadas mediante el demostrador de teoremas *Otter*.

En la siguiente sección se muestran dos ejemplos representativos con la formalización de un conjunto de políticas, las cuales fueron validadas con *Otter* encontrando inconsistencias en el primer conjunto de políticas. El segundo ejemplo se demostró que las políticas introducidas son consistentes entre ellas.

5.3.3 DE LENGUAJE NATURAL A LENGUAJE SIMBÓLICO

Para trasladar una política de seguridad de lenguaje natural a simbólico, es importante tomar en cuenta que una política es una proposición o un enunciado declarativo; partiendo de este hecho se construyó un analizador que lee de la base de datos cada uno de los elementos de las políticas descritas por el usuario y se utilizaron los siguientes símbolos de las conectivas lógicas:

1. El símbolo all (∀) denominado cuantificador universal representa la expresión "todos", "todas". Cuando la política habla del administrador de seguridad el lenguaje natural maneja la expresión "el"; sin embargo este se usará como cuantificador universal.

2. El símbolo exists (∃) denominado cuantificador existencial, es utilizado para representar a la expresión "algunos", "el", "la"... excepto cuando la política se refiera a las obligaciones del administrador de seguridad.

3. Cada término con que se nombra a un objeto o sujeto se representa por los valores almacenados en la base de datos. Por ejemplo, si la política habla sobre la información importante de la empresa, esté objeto se representará mediante la palabra *Informacion_Importante*. Conforme el programa analizador especifica un término, éste se compara con los datos de la base de datos con el objetivo de disminuir errores en la traducción.

4. Las literales minúsculas a, b, c,… se utilizan para representar variables individuales. Por ejemplo, *Informacion_Importante(a)*.

5. Los conectores lógicos que se utilizan como parte de la simbolización de políticas de seguridad son: *and* o *y* ($\wedge$), *or* u *o* ($\vee$) e *implicación* o *si y solo si* ($\rightarrow$).

El programa analizador traduce las políticas de seguridad separando cada uno de los elementos que la componen (sujeto, objeto, acción y mecanismo de acceso). Éste abre el archivo de políticas de seguridad y lee caracter por caracter de cada una de las políticas escritas por el usuario; estos caracteres se almacenan en una variable de tipo cadena que cambia de valor cuando se encuentra el caracter vacío (espacio).

Mediante el uso de ciclos repetitivos se hacen comparaciones del valor de la variable obtenida con la información leída de la base de datos y se genera la formalización correspondiente; dicha formalización se almacena en una variable. Como el analizador trabaja con los elementos de una política por separado (sujeto, objeto, acción y mecanismo de acceso si se requiere), cuando se ha terminado de analizar una política se integran cada una de sus partes obteniendo como resultado una política formalizada.

5.4 CONCLUSIONES

Las políticas de seguridad no solamente deben estar bien formuladas sino que deben ser consistentes unas con otras. Una posible solución a la validación de políticas de seguridad es el uso de un demostrador automático de teoremas el cual trabaja mediante la formalización de enunciados en lógica de primer orden. En este capítulo vimos los conceptos básicos utilizados en lógica de primer orden, así como los conceptos del demostrador de teoremas utilizado para la validación de las políticas de seguridad y la forma en que fueron integrados en esta tesis.

6 EJEMPLOS CON EL PROTOTIPO ADMINISTRADOR DE POLÍTICAS DE SEGURIDAD

6.1 INTRODUCCIÓN

En capítulos anteriores hemos hablado teóricamente de las características del prototipo administrador de políticas de seguridad entre las que encontramos la demostración automática de teoremas utilizada en la validación de dichas políticas. En las siguientes secciones se muestran dos ejemplos simples pero representativos de las capacidades del prototipo. En ellos se consideran dos empresas ficticias con algunos departamentos, puestos y empleados. El primer ejemplo se tomo del curso de seguridad computacional 1 de la maestría en ciencias computacionales impartido por el Dr. Jesús Vázquez y obtiene deliberadamente como resultado un conjunto de políticas inconsistentes, mientras que en el segundo ejemplo los resultados demuestran validez en las políticas capturadas después de un proceso de refinamiento.

6.2 EJEMPLO 1

6.2.1 ESPECIFICACIÓN DE POLÍTICAS

La “empresa X” dedicada a la comercialización de materias primas acaba de iniciar operaciones con cinco computadoras, una de ellas utilizada para respaldar la información, ésta última debe protegerse no importando el medio de almacenamiento en que se encuentra. La empresa inicia con un total de 10 empleados quienes ocupan los siguientes roles:

1. Gerente
2. Contador
3. Administrador
4. Recepcionista
5. Analista de mercados
6. Administrador de seguridad
7. Secretaria
8. Mensajero
9. Barrendero
10. Portero

Nota: Debido a que cuenta con pocos empleados, la empresa no está seccionada por departamentos.

Para proteger la información de la empresa el administrador de seguridad divide ésta de acuerdo a su importancia en general, importante y confidencial. Partiendo de estos datos el administrador describe políticas de seguridad mediante el prototipo desarrollado para facilitar su captura.

6.2.2 REPRESENTACIÓN DE POLÍTICAS DE SEGURIDAD

En este primer ejemplo, consideramos que el administrador[4] diseña las siguientes políticas:

1. *Todos los empleados pueden acceder a la información de la empresa.*
2. *Todos los empleados deben acceder a la información importante mediante passwords.*
3. *Toda la información importante debe estar protegida por medio de passwords.*
4. *Todos los barrenderos no pueden acceder a la información importante de la empresa.*

6.2.3 VALIDACIÓN DE POLÍTICAS DE SEGURIDAD

El prototipo realiza la tarea de traducción a través de un botón localizado en la pantalla principal, acción que requiere algunos segundos de tiempo por parte del administrador. En la siguiente sección se presenta la formalización de las políticas de seguridad capturadas por el administrador las cuales podrán ser leídas e interpretadas por *Otter*.

[4] Administrador de seguridad: persona encargada de proteger la seguridad de los recuros de una organización.

Nota: El tiempo de procesamiento en la traducción de políticas de seguridad escritas en lenguaje natural a sintaxis de *Otter* depende de la cantidad de políticas a traducir.

6.2.3.1 POLÍTICAS DE SEGURIDAD FORMALIZADAS EN SINTAXIS DE *OTTER*

set(auto).
formula_list(usable).

all x (Informacion_Confidencial(x) -> informacion(x)).
all x (Informacion_Importante(x) -> informacion(x)).
all x (Informacion_General(x) -> informacion(x)).
all x (Gerente(x) -> empleado(x)).
all x (Portero(x) -> empleado(x)).
all x (Contador(x) -> empleado(x)).
all x (Administrador(x) -> empleado(x)).
all x (Recepcionista(x) -> empleado(x)).
all x (Analista(x) -> empleado(x)).
all x (Telefonista(x) -> empleado(x)).
all x (Secretaria(x) -> empleado(x)).
all x (Mensajero(x) -> empleado(x)).
all x (Barrendero(x) -> empleado(x)).
all x (Passwords(x) -> mecanismo_acceso(x)).

exists x informacion(x).
exists x empleado(x).
exists x mecanismo_acceso(x).
exists x Informacion_Confidencial(x).
exists x Informacion_Importante(x).
exists x Informacion_General(x).
exists x Gerente(x).
exists x Portero(x).
exists x Contador(x).

exists x Administrador(x).
exists x Recepcionista(x).
exists x Analista(x).
exists x Telefonista(x).
exists x Secretaria(x).
exists x Mensajero(x).
exists x Barrendero(x).
exists x Passwords(x).

% Todos los empleados pueden acceder a la información de la empresa.
all y all x (empleado(y) & informacion(x) -> Acceder(y,x)).

% Todos los empleados deben acceder a la información importante mediante passwords.
all y all x (empleado(y) & Informacion_Importante(x) & Acceder(y,x) ->
(exists s (Passwords(s) & posee(y,s)))).

% Toda la información importante debe estar protegida por medio de passwords.
all y all x (Barrendero(y) & Informacion_Importante(x) -> -Acceder(y,x)).

% Todos los barrenderos no pueden acceder a la información importante de la empresa.
all x (Informacion_Importante(x) -> (exists s (Passwords(s) & protegida(x,s)))).

end_of_list.

6.2.3.2 EJECUCIÓN DE *OTTER*

Una vez creado el archivo de entrada de *Otter* con las políticas formalizadas, el administrador de seguridad mediante un botón de la interfaz gráfica del prototipo abre una consola que manipula el demostrador de teoremas. El tiempo de procesamiento ocupado por el demostrador en la verificación de las políticas es de 0.25 segundos (para este ejemplo).

Nota: El tiempo de procesamiento del demostrador de teoremas ocupado en la validación de enunciados formales depende del tamaño del archivo de entrada.

La verificación se detuvo en el momento en que fue encontrada la cláusula vacía, resultado que nos lleva a la conclusión de que hay inconsistencias en las políticas de seguridad capturadas y éstas deben ser modificadas o eliminadas según se requiera.

6.2.3.2.1 RESULTADOS DE *OTTER*

En esta sección se muestra la salida producida por el demostrador de teoremas *Otter* y se analizan los resultados:

----- Otter 3.3, August 2003 -----
The process was started by a Windows user on a Windows machine,
Thu Nov 18 21:44:02 2004
The command was "C:\Documents and Settings\Azurim\Escritorio\MCC-3er Semestre\tesis\OtterFace8_Win32\otter33".

set(auto).
dependent: set(auto1).
dependent: set(process_input).
dependent: clear(print_kept).
dependent: clear(print_new_demod).
dependent: clear(print_back_demod).
dependent: clear(print_back_sub).
dependent: set(control_memory).
dependent: assign(max_mem, 12000).
dependent: assign(pick_given_ratio, 4).
dependent: assign(stats_level, 1).
dependent: assign(max_seconds, 10800).

SCAN INPUT: prop=0, horn=1, equality=0, symmetry=0, max_lits=4.
.

dependent: set(hyper_res).

dependent: clear(order_hyper).

---------- PROOF ----------

2 [] -Informacion_Importante(x)|informacion(x).

13[] -Barrendero(x)|empleado(x).

15 [] -empleado(x)| -informacion(y)|Acceder(x,y).

20 [] -Barrendero(x)| -Informacion_Importante(y)| -Acceder(x,y).

25 [] Informacion_Importante($c5).

36 [] Barrendero($c16).

42 [hyper,25,2] informacion($c5).

53 [hyper,36,13] empleado($c16).

96 [hyper,53,15,42] Acceder($c16,$c5).

119 [hyper,96,20,36,25] $F.

-----> EMPTY CLAUSE at 0.25 sec ----> 143 [hyper,118,24,43,32] $F.

---------- end of proof ----------

---------- statistics ----------

clauses given	*92*
clauses generated	*94*
clauses kept	*142*
clauses forward subsumed	*0*
clauses back subsumed	*0*
Kbytes malloced	*255*

---------- times (seconds) ----------

user CPU time	*0.25*	*(0 hr, 0 min, 0 sec)*
system CPU time	*0.00*	*(0 hr, 0 min, 0 sec)*
wall-clock time	*0*	*(0 hr, 0 min, 0 sec)*
hyper_res time	*0.00*	
for_sub time	*0.00*	
back_sub time	*0.00*	
conflict time	*0.00*	
demod time	*0.00*	

That finishes the proof of the theorem.

Process 0 finished Thu Nov 18 21:44:02 2004

6.2.3.2.2 ANALIZANDO LOS RESULTADOS DE *OTTER*

La cláusula vacía fue encontrada con los siguientes símbolos:

1) Acceder($c16, $c5)
2) –Barrendero(x) | –Informacion_Importante(y) | –Acceder(x,y)
3) Barrendero($c16)
4) Informacion_Importante($c5)
5) Acceder(Barrendero, Información_Importante)… sustitución en (1) con (3) y (4)

Analizando los símbolos sobre los que se encontró la cláusula vacía, se observa que el barrendero está involucrado con la información importante mediante un acceso, por lo tanto deben verificarse aquellas políticas que incluyen éstos elementos (ver política no. 4 y 2).

La política cuatro afirma que ningún barrendero puede acceder a la información importante, sin embargo un barrendero es un empleado por lo tanto contradice la regla número dos en la que se dice que todos los empleados pueden acceder a la información importante.

Nota[5]: Al analizar los datos arrojados por *otter* no es necesario entender exactamente que se hizo, sino ir directamente al párrafo *PROOF* y revisar que elementos están involucrados en la obtención de la cláusula vacía.

[5] El tiempo que el usuario ocupará en la interpretación de los resultados de *Otter* dependerá de la cantidad de políticas que involucren los elementos que llevan a la cláusula vacía.

6.3 EJEMPLO 2

La “empresa Y” dedicada a la comercialización de materias primas, está preocupada por su información y necesita un conjunto de políticas de seguridad que apoyen su protección. La información de la empresa fue dividida de acuerdo a su importancia en general, importante y confidencial. La tabla 6.1 muestra los departamentos que constituyen la empresa y los empleados de cada uno de ellos y en la tabla 6.2 se enlista la información de la empresa.

Departamentos y sus empleados	
Departamento	**Empleados**
Sistemas	1 Gerente (o encargado de departamento), 1 Programador, 1 Analista
Contabilidad	1 Gerente (o encargado de departamento), 1 Contador
Administración	1 Gerente (o encargado de departamento), 1 Administrador
Finanzas	1 Gerente (o encargado de departamento), 1 Contador
Mercadotecnia	1 Gerente (o encargado de departamento), 1 Administrador
Seguridad	1 Gerente (o encargado de departamento), 1 Telefonista
Recepción	1 Secretaria (o encargado de departamento), 1 Mensajero
Independiente	1 Administrador de seguridad

Tabla 6.1. Departamentos y empleados.

Información de la empresa		
Información	**Departamento**	**Importancia**
Estados financieros	Contabilidad	Confidencial
Nomina	Contabilidad	Confidencial
Archivo de passwords	Sistemas	Confidencial
Archivos locales	Todos	Importante
Políticas generales	Sistemas	General
Información generada por personal de la empresa	Todos	General
Plan de contingencias	Sistemas	Importante

Información de la empresa		
Información	**Departamento**	**Importancia**
Análisis de mercados	Mercadotecnia	Importante
Procedimientos de recuperación de desastres	Sistemas	Importante
Documento de análisis de riesgos	Sistemas	Importante

Tabla 6.2. Información de la empresa.

Antes de escribir las políticas de seguridad, es necesario analizar los elementos involucrados en la seguridad de la información de la empresa (empleados clasificados por su rol y departamento, información clasificada por su importancia, mecanismos de acceso implantados para la protección de la información, entre otros.).

La tabla 6.3 muestra qué personal tiene acceso autorizado a la información sensible de la empresa especificando el mecanismo de acceso. Esta información también será formalizada y validada por *Otter*.

Accesos a la información			
Puesto-Departamento	**Informacion**	**Acceso**	**Mecanismo**
Gerente de sistemas	Archivo de passwords	Total	Passwords
Gerente de sistemas	Políticas generales	Total	Passwords
Gerente de sistemas	Plan de contingencias	Lectura	Passwords
Gerente de sistemas	Procedimientos recuperación de desastres	Lectura	Passwords
Gerente de sistemas	Documento de análisis de riesgos	Lectura	Passwords
Programador de sistemas	Políticas generales	Lectura	Cualquiera
Programador de sistemas	Plan de contingencias	Lectura	Passwords
Programador de sistemas	Procedimientos recuperación de desastres	Lectura	Passwords
Programador de sistemas	Documento de análisis de riesgos	Lectura	Passwords
Gerente de contabilidad	Estados financieros	Total	Passwords
Gerente de contabilidad	Nómina	Total	Passwords

Accesos a la información			
Puesto-Departamento	**Informacion**	**Acceso**	**Mecanismo**
Contador de contabilidad	Estados financieros	Lectura	Passwords
Contador de contabilidad	Políticas generales	Lectura	Cualquiera
Gerente de administración	Políticas generales	Lectura	Cualquiera
Administrador de administración	Políticas generales	Lectura	Cualquiera
Contador de finanzas	Políticas generales	Lectura	Cualquiera
Gerente de mercadotecnia	Análisis de mercados	Total	Passwords
Administrador de mercadotecnia	Análisis de mercados	Lectura	Passwords

Tabla 6.3. Accesos autorizados a la información de la empresa por parte de los empleados de la empresa, mediante un mecanismo de acceso.

6.3.1 REPRESENTACIÓN DE POLÍTICAS DE SEGURIDAD

Al escribir las políticas de seguridad el administrador no revisó exhaustivamente que éstas fueran inconsistentes, simplemente usando su intuición se dio a la tarea de capturar las políticas mediante el prototipo administrador[6] y estas se dividen en un conjunto de políticas específicas que incluye las responsabilidades del administrador de seguridad y otro conjunto de políticas generales.

6.3.1.1 POLÍTICAS DE SEGURIDAD ESPECÍFICAS

1. Todos los empleados de la empresa deben respaldar los archivos locales de la empresa.

2. Todos los gerentes de la empresa deben acceder a la información importante mediante passwords.

3. El gerente de sistemas debe acceder al archivo de passwords mediante cualquier mecanismo de acceso.

[6] Prototipo Administrador de Políticas de Seguridad: Herramienta propuesta en esta tesis para administrar políticas de seguridad.

4. Todos los programadores de la empresa deben acceder a los archivos locales de la empresa mediante passwords.

5. Todos los empleados del departamento de sistemas deben acceder a la información importante de la empresa mediante passwords.

6. Todos los programadores del departamento de sistemas no pueden acceder a la información confidencial.

7. Todos los empleados del departamento de sistemas deben respaldar la información del departamento de sistemas.

8. Todos los empleados del departamento de contabilidad deben administrar la información del departamento de contabilidad mediante passwords.

9. El gerente del departamento de contabilidad debe acceder a la nómina de la empresa mediante passwords.

10. Toda la información confidencial debe protegerse mediante passwords.

11. Toda la información general debe estar disponible a cualquier empleado de la empresa.

12. Toda la información del departamento de sistemas debe estar disponible a los empleados del departamento de sistemas mediante passwords.

13. Toda la información importante debe generarse por empleados autorizados.

14. Los archivos locales de la empresa deben eliminarse por empleados autorizados.

15. Los estados financieros de la empresa deben estar disponibles al gerente del departamento de contabilidad mediante cualquier mecanismo de acceso.

16. Los archivos de passwords de la empresa no deben estar disponibles a cualquier empleado de la empresa.

17. Toda la información importante debe estar disponible a los gerentes de la empresa mediante cualquier mecanismo de acceso.

18. Toda la información del departamento de contabilidad debe eliminarse por los gerentes del departamento de contabilidad mediante passwords.

19. Todos los administradores de la empresa no pueden acceder a los estados financieros de la empresa.

20. Todos los empleados de la empresa deben tener acceso de lectura a las políticas generales de la empresa.

21. Todos los empleados de la empresa no pueden tener acceso total a las políticas generales de la empresa.

22. La nómina de la empresa debe estar protegida mediante passwords.

6.3.1.2 RESPONSABILIDADES DEL ADMINISTRADOR DE SEGURIDAD

23. El administrador de seguridad debe otorgar passwords a cualquier empleado de la empresa.

24. El administrador de seguridad debe resguardar la información confidencial de la empresa.

25. El administrador de seguridad debe resguardar la información importante de la empresa.

26. El administrador de seguridad debe generar los procedimientos de recuperación de desastres de la empresa.

27. El administrador de seguridad debe otorgar mecanismos de acceso a todos los empleados de la empresa.

28. El administrador de seguridad debe administrar el archivo de passwords de la empresa.

29. El administrador de seguridad debe preservar la integridad de la información confidencial de la empresa.

30. El administrador de seguridad debe preservar la integridad de la información importante de la empresa.

31. El administrador de seguridad debe resguardar la información confidencial de la empresa.

32. El administrador de seguridad debe resguardar la información importante de la empresa.

6.3.1.3 POLÍTICAS DE SEGURIDAD GENERALES

33. Toda información propiedad de la empresa deberá estar protegida adecuadamente en cuanto a su integridad, confidencialidad, disponibilidad y autenticación.

34. Todo daño deliberado, robo o modificación no autorizada de la información propiedad de la empresa deberá ser sancionado.

35. Todo daño causado por negligencia a la información propiedad de la empresa, deberá ser sancionado.

36. Los administradores de seguridad son responsables de proveer un ambiente seguro en el cual pueda ser mantenida con integridad.

37. Toda información propiedad de la empresa deberá ser generada, eliminada o modificada solamente por las personas autorizadas y en los lugares autorizados por la empresa.

38. Toda información propiedad de la empresa debe clasificarse de acuerdo a su nivel de importancia para el negocio.

39. El acceso a la información de la compañía se restringe solamente a usuarios autorizados por el administrador de seguridad.

40. Toda información propiedad de la empresa deberá estar disponible en el momento que se requiera sólo a aquellas personas a quienes está destinada.

41. El administrador de seguridad es responsable de proveer un mecanismo de acceso a los usuarios autorizados para manipular la información de la empresa.

6.3.2 VALIDACIÓN DE POLÍTICAS DE SEGURIDAD

Las políticas fueron capturadas, a través de una sesión con nuestro prototipo, en un tiempo de treinta minutos aproximadamente. Se realizó después la tarea de validación, e inmediatamente *Otter* detectó una inconsistencia entre las políticas capturadas y la tabla de accesos 6.3. El error consiste en que la política de seguridad afirma que todos los empleados de la empresa tenían acceso a la información general; sin embargo, en la tabla de accesos el gerente del departamento de contabilidad sólo tenía acceso restringido a las políticas generales de la empresa y éstas se encuentran en el rubro de información general.

Al obtener los resultados del demostrador y con conocimientos básicos de lógica de predicados, encontramos el error después de 10 minutos y modificamos la tabla de accesos, donde se encontraba el problema. Tras la modificación, realizamos nuevamente la tarea de validación generando y evaluando el archivo de políticas formalizadas, nuevamente se encontró otra inconsistencia con dos de las políticas, una de ellas afirmaba que todos los gerentes de cada departamento tenían acceso restringido a la información importante de la empresa, mientras que la otra decía todo lo contrario, se realizaron nuevamente los cambios correspondientes y se volvió a evaluar el archivo de políticas. El documento tuvo que modificarse y evaluarse un par de veces más hasta que no se encontró ningún problema con las políticas capturadas.

En la siguiente sección se muestran el conjunto de políticas de seguridad formalizadas con las modificaciones correspondientes de acuerdo a los últimos resultados obtenidos por el demostrador de teoremas.

6.3.2.1 POLÍTICAS DE SEGURIDAD FORMALIZADAS EN SINTAXIS DE *OTTER*

set(auto).

formula_list(usable).

all y (Gerente_Sistemas(y) -> Empleado_Sistemas(y)).

all y (Empleado_Sistemas(y) -> empleado(y)).

all y (Programador_Sistemas(y) -> Empleado_Sistemas(y)).

all y (Gerente_Contabilidad(y) -> Empleado_Contabilidad(y)).

all y (Empleado_Contabilidad(y) -> empleado(y)).

all y (Contador_Contabilidad(y) -> Empleado_Contabilidad(y)).

all y (Gerente_Administracion(y) -> Empleado_Administracion(y)).

all y (Empleado_Administracion(y) -> empleado(y)).

all y (Administrador_Administracion(y) -> Empleado_Administracion(y)).

all y (Empleado_Administracion(y) -> empleado(y)).

all y (Contador_Finanzas(y) -> Empleado_Finanzas(y)).

all y (Empleado_Finanzas(y) -> empleado(y)).

all y (Gerente_Mercadotecnia(y) -> Empleado_Mercadotecnia(y)).

all y (Empleado_Mercadotecnia(y) -> empleado(y)).

all y (Administrador_Mercadotecnia(y) -> Empleado_Mercadotecnia(y)).

all x (Administrador_seguridad(x) -> empleado(x)).

all x (Informacion_Confidencial(x) -> informacion(x)).

all x (Informacion_Importante(x) -> informacion(x)).

all x (Informacion_General(x) -> informacion(x)).

all x (Gerente(x) -> empleado(x)).

all x (Programador(x) -> empleado(x)).

all x (Contador(x) -> empleado(x)).

all x (Administrador(x) -> empleado(x)).

all x (Recepcionista(x) -> empleado(x)).

all x (Analista(x) -> empleado(x)).

all x (Telefonista(x) -> empleado(x)).

all x (Secretaria(x) -> empleado(x)).

all x (Mensajero(x) -> empleado(x)).

all x (Estados_financieros(x) -> Informacion_Confidencial(x)).

all x (Nomina(x) -> Informacion_Confidencial(x)).

all x (Archivo_de_passwords(x) -> Informacion_Confidencial(x)).

all x (Archivos_locales(x) -> Informacion_Importante(x)).

all x (Politicas_generales(x) -> Informacion_General(x)).

all x (Informacion_generada_por_personal_de_la_empresa(x) -> Informacion_General(x)).

all x (Plan_de_contingencias(x) -> Informacion_Confidencial(x)).

all x (Analisis_de_mercados(x) -> Informacion_Confidencial(x)).

all x (Procedimientos_de_recuperacion_de_desastres(x) -> Informacion_Confidencial(x)).

all x (Documento_de_analisis_de_riesgos(x) -> Informacion_Confidencial(x)).

all x (Integridad(x) -> propiedad_informacion(x)).

all x (Seguridad(x) -> propiedad_informacion(x)).

all x (Disponibilidad(x) -> propiedad_informacion(x)).

all x (Confidencialidad(x) -> propiedad_informacion(x)).

all x (No_repudiacion(x) -> propiedad_informacion(x)).

all x (Passwords(x) -> mecanismo_acceso(x)).

all x (Gerente_Sistemas(x) -> Empleado_Sistemas(x)).

all x (Empleado_Sistemas(x) -> empleado(x)).

all x (Programador_Sistemas(x) -> Empleado_Sistemas(x)).

all x (Empleado_Sistemas(x) -> empleado(x)).

all x (Empleado_Contabilidad(x) -> empleado(x)).

all x (Gerente_Contabilidad(x) -> Empleado_Contabilidad(x)).

all x (Informacion_Sistemas(x) -> informacion(x)).

all x (Informacion_Contabilidad(x) -> informacion(x)).

all x (Empleado_Sistemas(x) -> empleado(x)).

all x (Empleado_Autorizado(x) -> empleado(x)).

all x (Empleado_Contabilidad(x) -> empleado(x)).

exists y (Gerente_Sistemas(y)).

exists y (Programador_Sistemas(y)).

exists y (Gerente_Contabilidad(y)).

exists y (Contador_Contabilidad(y)).

exists y (Gerente_Administracion(y)).

exists y (Administrador_Administracion(y)).

exists y (Contador_Finanzas(y)).

exists y (Gerente_Mercadotecnia(y)).

exists y (Administrador_Mercadotecnia(y)).

exists x informacion(x).

exists x empleado(x).

exists x mecanismo_acceso(x).

exists x Administrador_seguridad(x).

exists x Informacion_Confidencial(x).

exists x Informacion_Importante(x).

exists x Informacion_General(x).

exists x Gerente(x).

exists x Programador(x).

exists x Contador(x).

exists x Administrador(x).

exists x Recepcionista(x).

exists x Analista(x).

exists x Telefonista(x).

exists x Secretaria(x).

exists x Mensajero(x).

exists x Estados_financieros(x).

exists x Nomina(x).

exists x Archivo_de_passwords(x).

exists x Archivos_locales(x).

exists x Politicas_generales(x).

exists x Informacion_generada_por_personal_de_la_empresa(x).

exists x Plan_de_contingencias(x).

exists x Analisis_de_mercados(x).

exists x Procedimientos_de_recuperacion_de_desastres(x).

exists x Documento_de_analisis_de_riesgos(x).

exists x Integridad(x).

exists x Seguridad(x).

exists x Disponibilidad(x).

exists x Confidencialidad(x).

exists x No_repudiacion(x).

exists x Passwords(x).

exists x Gerente_Sistemas(x).

exists x Empleado_Sistemas(x).

exists x Programador_Sistemas(x).

exists x Empleado_Contabilidad(x).

exists x Gerente_Contabilidad(x).

exists x Informacion_Sistemas(x).

exists x Informacion_Contabilidad(x).

exists x Empleado_Autorizado(x).

% Accesos autorizados de los empleados (ver tabla 6.3)

exists y exists x (Gerente_Sistemas(y) & Politicas_generales(x) & Acceso_Total(y,x)
& (exists s (Passwords(s) & posee(y,s))))).

exists y exists x (Gerente_Sistemas(y) & Plan_de_contingencias(x) & Acceso_Lectura(y,x)
& (exists s (Passwords(s) & posee(y,s))))).

exists y exists x (Gerente_Sistemas(y) & Procedimientos_de_recuperacion_de_desastres(x)
& Acceso_Lectura(y,x) & (exists s (Passwords(s) & posee(y,s))))).

exists y exists x (Gerente_Sistemas(y) & Documento_de_analisis_de_riesgos(x)
& Acceso_Lectura(y,x) & (exists s (Passwords(s) & posee(y,s))))).

exists y exists x (Gerente_Sistemas(y) & Archivo_de_passwords(x) & Acceso_Total(y,x)
& (exists s (Passwords(s) & posee(y,s))))).

exists y exists x (Programador_Sistemas(y) & Plan_de_contingencias(x)
& Acceso_Lectura(y,x) & (exists s (Passwords(s) & posee(y,s))))).

exists y exists x (Programador_Sistemas(y) & Politicas_generales(x)
& Acceso_Lectura(y,x)).

exists y exists x (Programador_Sistemas(y) &
Procedimientos_de_recuperacion_de_desastres(x) & Acceso_Lectura(y,x)
& (exists s (Passwords(s) & posee(y,s))))).

exists y exists x (Programador_Sistemas(y) & Documento_de_analisis_de_riesgos(x)
& Acceso_Lectura(y,x) & (exists s (Passwords(s) & posee(y,s))))).

exists y exists x (Gerente_Contabilidad(y) & Nomina(x) & Acceso_Total(y,x)
& (exists s (Passwords(s) & posee(y,s))))).

exists y exists x (Gerente_Contabilidad(y) & Estados_financieros(x) & Acceso_Total(y,x)
& (exists s (Passwords(s) & posee(y,s))))).

exists y exists x (Contador_Contabilidad(y) & Estados_financieros(x)
& Acceso_Lectura(y,x) & (exists s (Passwords(s) & posee(y,s))))).

exists y exists x (Contador_Contabilidad(y) & Politicas_generales(x)
& Acceso_Lectura(y,x)).

exists y exists x (Gerente_Administracion(y) & Politicas_generales(x)
& Acceso_Lectura(y,x)).

exists y exists x (Administrador_Administracion(y) & Politicas_generales(x)
& Acceso_Lectura(y,x)).

exists y exists x (Contador_Finanzas(y) & Politicas_generales(x) & Acceso_Lectura(y,x)).

exists y exists x (Gerente_Mercadotecnia(y) & Analisis_de_mercados(x)
& Acceso_Total(y,x) & (exists s (Passwords(s) & posee(y,s))))).

exists y exists x (Administrador_Mercadotecnia(y) & Analisis_de_mercados(x)
& Acceso_Lectura(y,x) & (exists s (Passwords(s) & posee(y,s))))).

% Políticas generales formalizadas

all y all x (empleado(y) & informacion(x) & protege(y,x) ->(exists a (Integridad(a) & preserva(x,a))) | (exists b (Confidencialidad(b) & preserva(x,b))) | (exists c (Disponibilidad(c) & preserva(x,c))) | (exists d (Autenticacion(d) & preserva(x,d)))).

all x all y (empleado(y) & informacion(x) & dana(y,x) ->
(exists y1 (empleado(y1) & sanciona(y1,y)))).

all y all x (Administrador_seguridad(y) & informacion(x) -> (exists a (Integridad(a) & ad_preserva(y, a, x))) & (exists b (Disponibilidad(b) & ad_preserva(y, b, x)) & (exists c (Confidencialidad(c) & ad_preserva(y, c, x)))).

all y all x (empleado(y) & informacion(x) & Generar(y, x) ->
(exists z (Administrador_seguridad(z) & autoriza(z, y, crear)))).

all y all x (empleado(y) & informacion(x) & Borrar(y, x) ->
(exists z (Administrador_seguridad(z) & autoriza(z, y, eliminar)))).

all y all x (empleado(y) & informacion(x) & Modificar(y, x) ->
(exists z (Administrador_seguridad(z) & autoriza(z, y, cambiar)))).

all x (informacion(x) -> (exists z (importancia(z) & clasifica(x,z)))).

all y all x (empleado(y) & informacion(x) & Acceder(y, x) ->
(exists z (Administrador_seguridad(z) & autoriza(z, y, ingresar)))).

all x all y (informacion(x) & empleado(y) & Manipular(y, x) -> disponible(x, y)).

all y all x (Acceder(y, x) | Generar(y, x) | Borrar(y, x) | Modificar(y, x) -> Manipular(y, x)).

all y all x (empleado(y) & informacion(x) & Manipular(y, x) -> (exists z (Administrador_seguridad(z) & (exists s (mecanismo_acceso(s) & provee(z, y, s))))).

% Políticas particulares formalizadas

all y exists x (empleado(y) & Archivos_locales(x) -> Respaldar(y,x)).

all y all x (Gerente(y) & Informacion_Importante(x) & Acceder(y,x) ->
(exists s (Passwords(s) & posee(y,s)))).

exists y exists x (Gerente_Sistemas(y) & Archivo_de_passwords(x) & Acceder(y,x) &
(exists s (mecanismo_acceso(s) & posee(y,s)))).

all y exists x (Programador(y) & Archivos_locales(x) & Acceder(y,x) ->
(exists s (Passwords(s) & posee(y,s)))).

all y all x (Empleado_Sistemas(y) & Informacion_Importante(x) & Acceder(y,x) ->
(exists s (Passwords(s) & posee(y,s)))).

exists y all x (Programador_Sistemas(y) & Informacion_Confidencial(x) & -Acceder(y,x)).

all y all x (Empleado_Sistemas(y) & Informacion_Sistemas(x) -> Respaldar(y,x)).

all y all x (Empleado_Contabilidad(y) & Informacion_Contabilidad(x) & Administrar(y,x) -> (exists s (Passwords(s) & posee(y,s)))).

exists y exists x (Gerente_Contabilidad(y) & Nomina(x) & Acceder(y,x) &
(exists s (Passwords(s) & posee(y,s)))).

all x (Informacion_Confidencial(x) -> (exists s (Passwords(s) & protegida(x,s)))).

all x exists y (Informacion_General(x) & empleado(y) -> disponible(x,y)).

all x all y (Informacion_Sistemas(x) & Empleado_Sistemas(y) & disponible(x,y) ->
(exists s (Passwords(s) & posee(y,s)))).

all x exists y (Informacion_Importante(x) & Empleado_Autorizado(y) -> generada(x,y)).

exists x exists y (Archivos_locales(x) & Empleado_Autorizado(y) & eliminada(x,y)).

exists x y (Estados_financieros(x) & Empleado_Contabilidad(y) & disponible(x,y) &
(exists s (mecanismo_acceso(s) & posee(y,s)))).

exists x exists y (Archivo_de_passwords(x) & empleado(y) & -disponible(x,y)).

all x all y (Informacion_Importante(x) & Gerente(y) & disponible(x,y) ->
(exists s (mecanismo_acceso(s) & posee(y,s)))).

all x y (Informacion_Contabilidad(x) & Empleado_Contabilidad(y) & eliminada(x,y) ->
(exists s (Passwords(s) & posee(y,s)))).

all y exists x (Administrador(y) & Estados_financieros(x) -> -Acceder(y,x)).

all y exists x (empleado(y) & Politicas_generales(x) -> Acceso_Lectura(y,x)).

all y exists x (empleado(y) & Politicas_generales(x) -> -Acceso_Total(y,x)).

exists x (Nomina(x) & (exists s (Passwords(s) & protegida(x,s)))).

% Políticas de seguridad especificas - obligaciones del encargado o administrador de
% seguridad

all y (Administrador_seguridad(y) -> (exists y1 (empleado(y1) &
(exists s (Passwords(s) & ad_otorga(y,s,y1)))))).

all y (Administrador_seguridad(y) ->

(all x (Informacion_Confidencial(x) -> ad_protege(y,x))).

all y (Administrador_seguridad(y) ->
(all x (Informacion_Importante(x) -> ad_protege(y,x))).

all y (Administrador_seguridad(y) ->
(exists x (Procedimientos_de_recuperacion_de_desastres(x) &
ad_genera(y,x))).

all y (Administrador_seguridad(y) -> (all y1 (empleado(y1) ->
(exists s (mecanismo_acceso(s) & ad_otorga(y,s,y1)))))).

all y (Administrador_seguridad(y) ->
(exists x (Archivo_de_passwords(x) & ad_administra(y,x))).

all y (Administrador_seguridad(y) ->
(all x (Informacion_Confidencial(x) -> (exists w (Integridad(w) & ad_preserva(y,w,x)))))).

all y (Administrador_seguridad(y) ->
(all x (Informacion_Importante(x) -> (exists w (Integridad(w) & ad_preserva(y,w,x)))))).

all y (Administrador_seguridad(y) ->
(all x (Informacion_Confidencial(x) -> ad_protege(y,x)))).

all y (Administrador_seguridad(y) ->
(all x (Informacion_Importante(x) -> ad_protege(y,x)))).

end_of_list.

La validación de las políticas se detuvo en el momento en que ya no se encontró ninguna forma de inferir nuevas cláusulas, lo que significa que la cláusula vacía no es encontrada por lo tanto el conjunto de políticas de seguridad establecidas por el administrador de seguridad son consistentes

unas con otras y pueden ser implantadas en la organización. (Ver resultados de *Otter* en los anexos al final del documento)

6.4 CONCLUSIONES

Como parte del experimento de validación, las políticas de seguridad generales de los dos ejemplos mostrados en secciones anteriores fueron negadas deliberadamente una vez que el documento no presentaba inconsistencias. Sin embargo, los resultados arrojados por el demostrador *Otter* no reflejaron inconsistencias. Esto se debe a que las políticas generales representan metas de seguridad que cumplen las políticas particulares especificadas. Si una política particular contradice cualquier meta de seguridad, el demostrador encuentra inconsistencias

El proceso de formar políticas de seguridad correctas es más fácil contando con un sistema como el que se presenta en esta tesis, ya que a pesar de que el tiempo utilizado en la captura y validación de las políticas de seguridad es relativamente grande no rebasa el tiempo utilizado en la descripción de políticas de manera tradicional. Además de que la herramienta garantiza que las políticas capturadas realmente protegen la seguridad de la información ya que no hay contradicciones entre ellas que puedan perjudicar su seguridad.

7 EVALUACIÓN DEL PROTOTIPO ADMINISTRADOR DE POLÍTICAS DE SEGURIDAD

7.1 INTRODUCCIÓN

Toda investigación o desarrollo de un sistema requiere de la evaluación de los resultados obtenidos. Existen algunas técnicas de evaluación desarrolladas por el área de mercadotecnia, entre las que se encuentran con buenos resultados, las encuestas[7].

Para complementar la investigación de este trabajo, se formuló un cuestionario que fue aplicado a seis personas del área de sistemas, tres de ellas laboran directamente en el departamento de seguridad. Antes de aplicar el cuestionario se dio a conocer el prototipo administrador de políticas mediante una exposición de las capacidades del sistema. En la siguiente sección se analizan los resultados obtenidos.

7.2 POBLACIÓN ENCUESTADA

• Lic. Ricardo González Vargas Director de Seguridad Computacional ITESM – CEM	• Ing. Damián Guerra Farias Ingeniero de Seguridad Computacional ITESM – CEM

[7] Las encuestas son procedimientos utilizados en la investigación de mercados para obtener información relevante que evalúe un proyecto, un sistema, o cualquier tipo de producto, mediante preguntas dirigidas a una muestra de personas representativa de la población para la que va dirigido el producto.

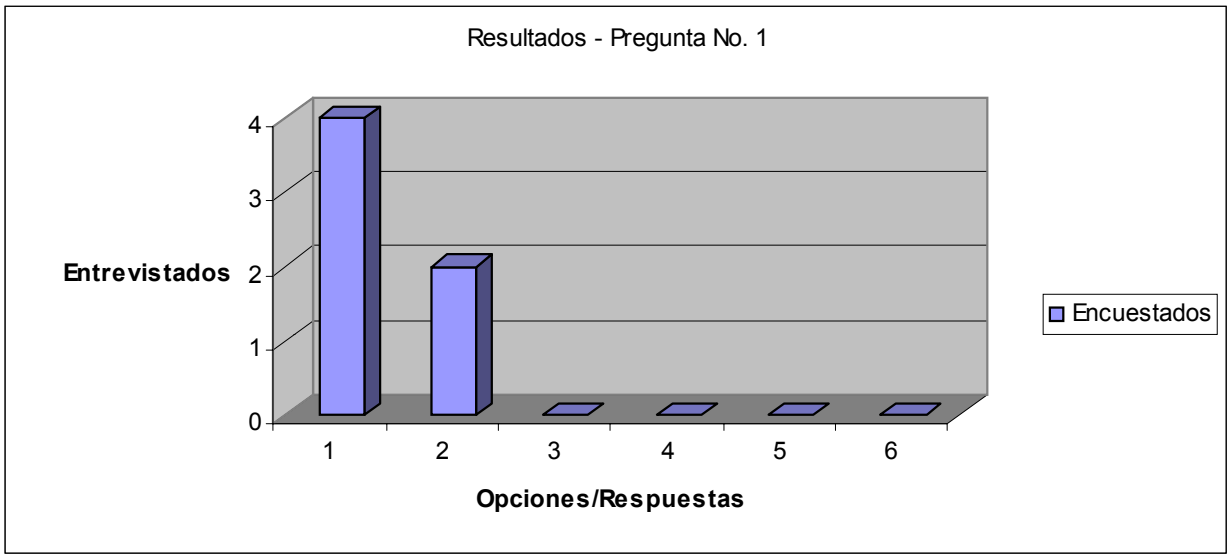
• Dr. Roberto Gómez Cátedra de Seguridad Computacional ITESM – CEM	• Ing. Edgar Gutiérrez Martínez Seguridad Informática MAKYMAT[8], S.A. DE C.V.
• Ing. Yene Mejía Alcauter Analista en Sistemas QA-CONSULTING GROUP[9]	• MCC. Alfonso Estrada Díaz Líder de Proyectos QA-CONSULTING GROUP

Nota: El orden de la población encuestada es en base a la aplicación del cuestionario.

7.3 RESULTADOS DE LAS PREGUNTAS CERRADAS

La escala de calificación va desde 1 que es totalmente de acuerdo hasta 6 que es totalmente en desacuerdo, excepto la pregunta número 3 que se califica al revés.

1. ¿Es fácil de entender el uso del prototipo?



De las 6 personas encuestadas:

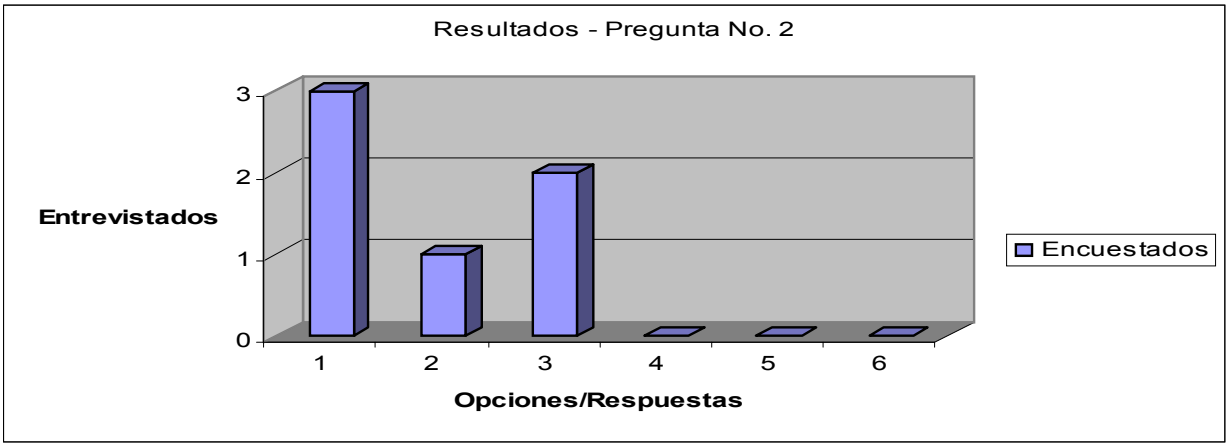
4 respondieron con la opción número 1, y

2 con la opción número 2.

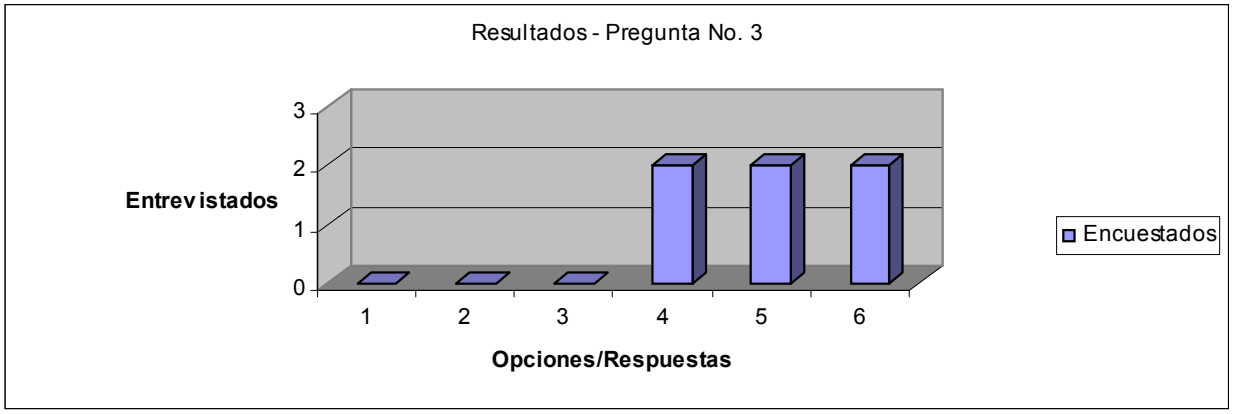
Nota: La interpretación a las siguientes gráficas es de la misma forma.

[8] Makymat S.A. DE C.V., empresa dedicada a la producción de materias primas.
[9] QA Consulting Group, consultoría en sistemas computacionales.

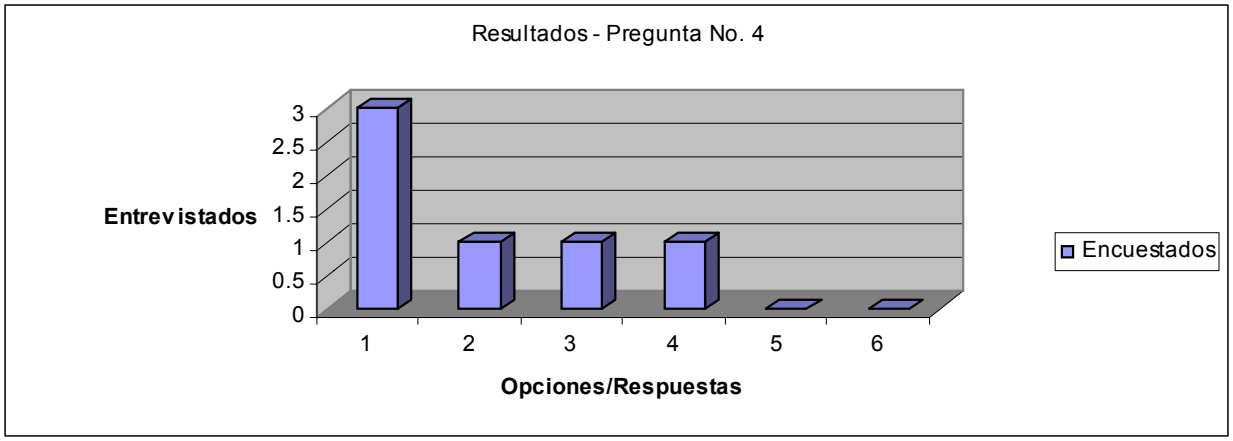
2. ¿El prototipo se usa fácilmente?



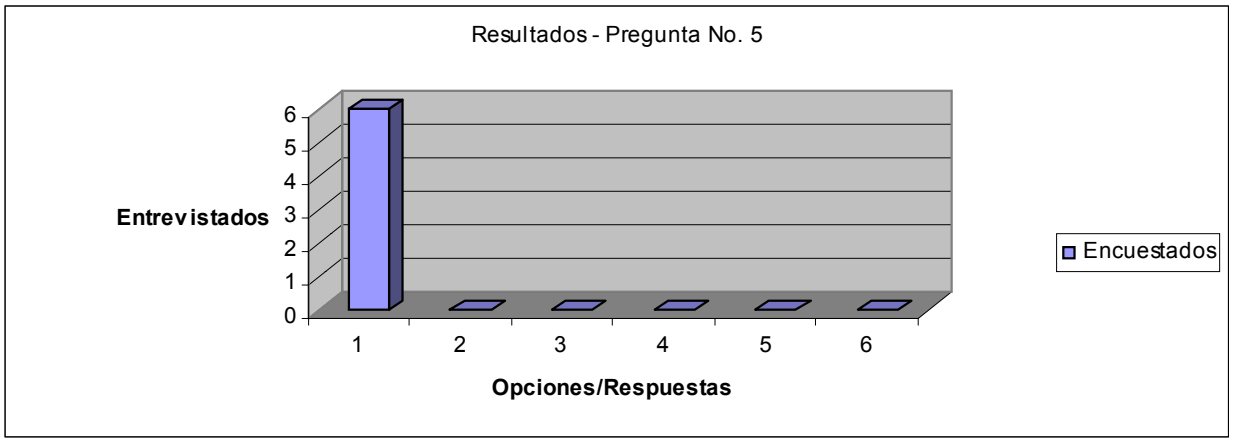
3. ¿Encontró dificultades al capturar políticas de seguridad?



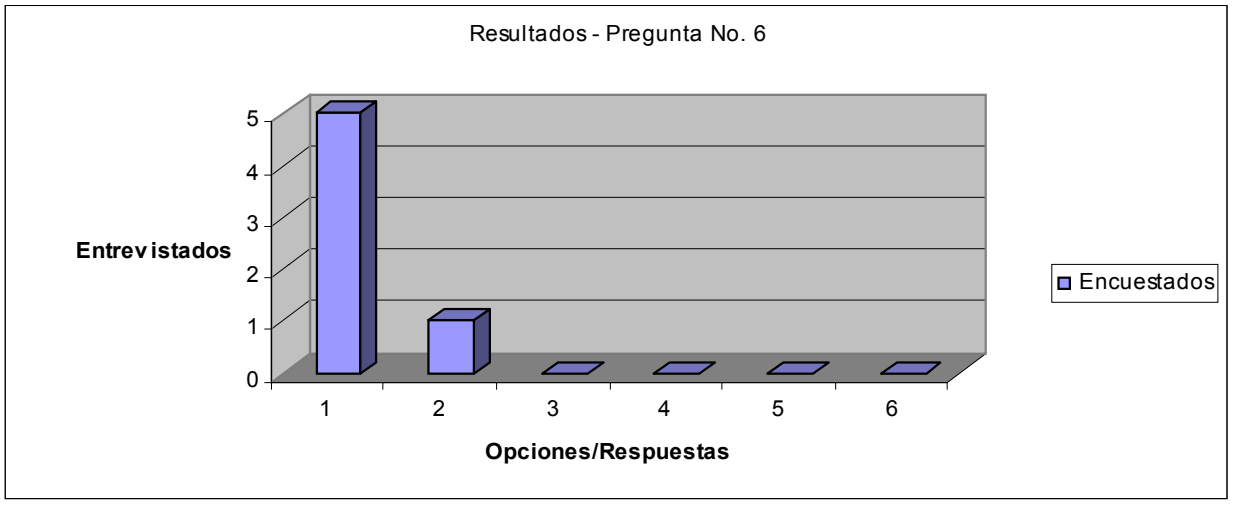
4. ¿Desde su punto de vista, el prototipo incluye todas las opciones necesarias para administrar una política?



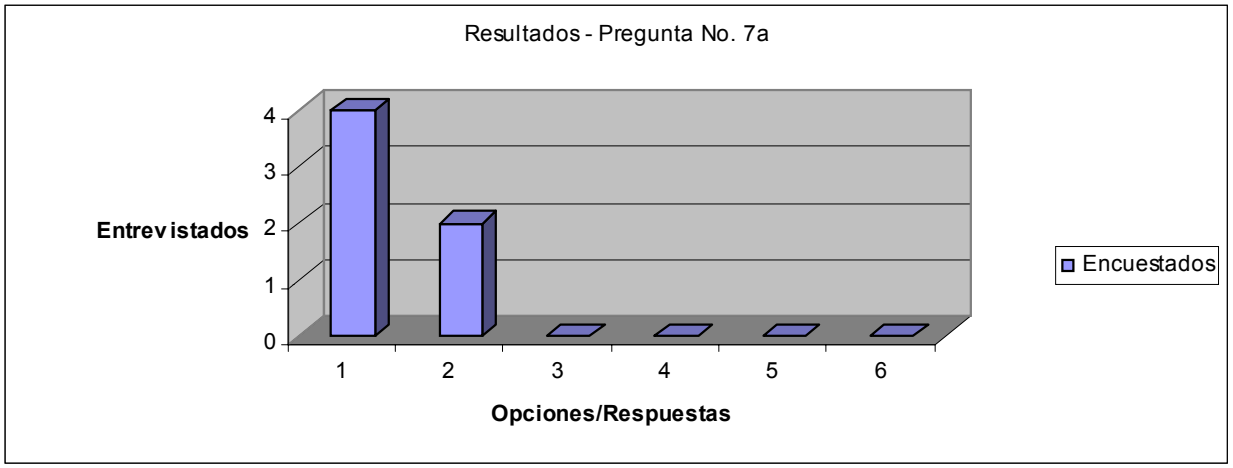
5. ¿Le parece útil el prototipo?



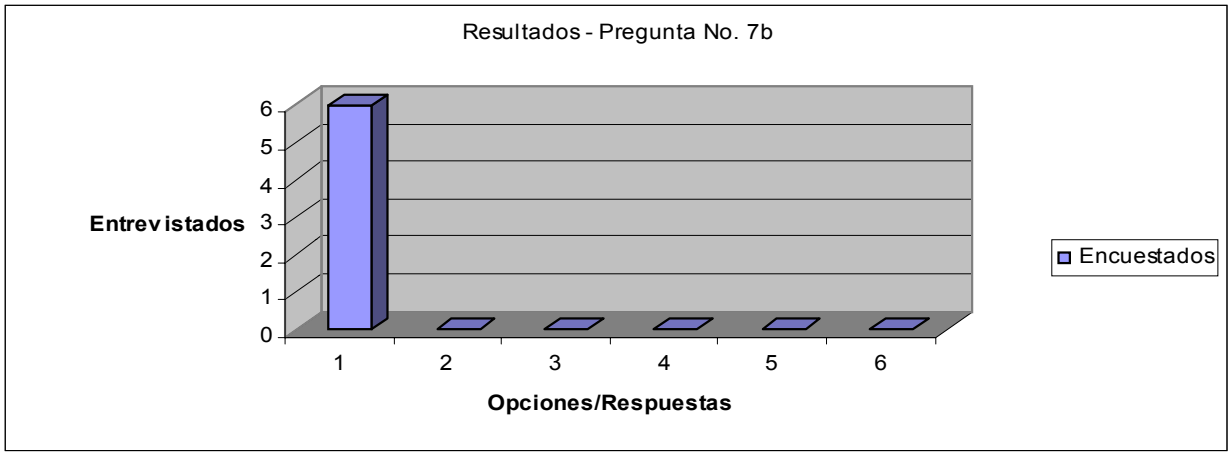
6. ¿Le encuentra beneficios a la herramienta?



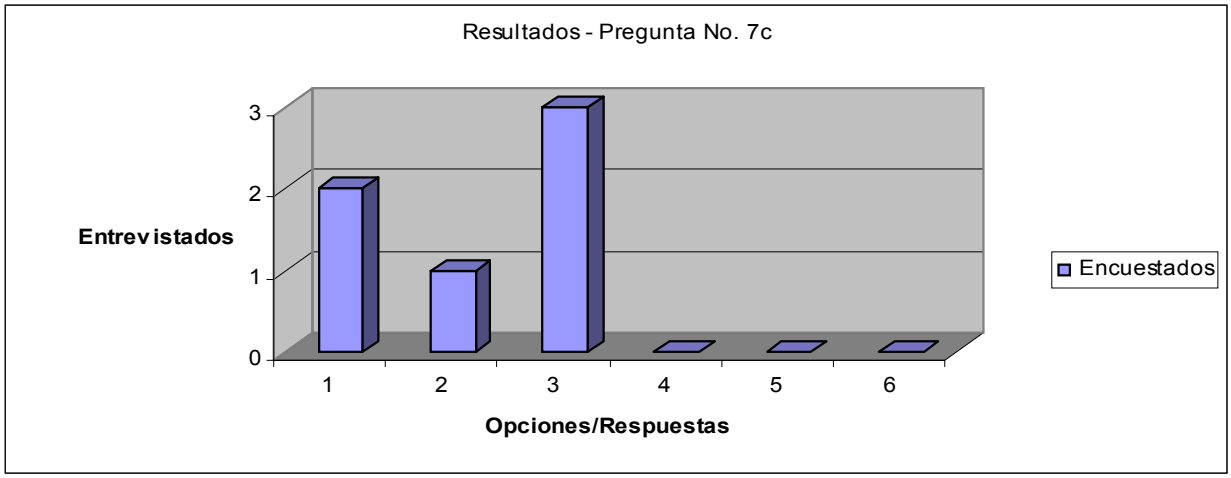
7. ¿Considera que las políticas que se pueden capturar con la herramienta son…?
   a) adecuadas



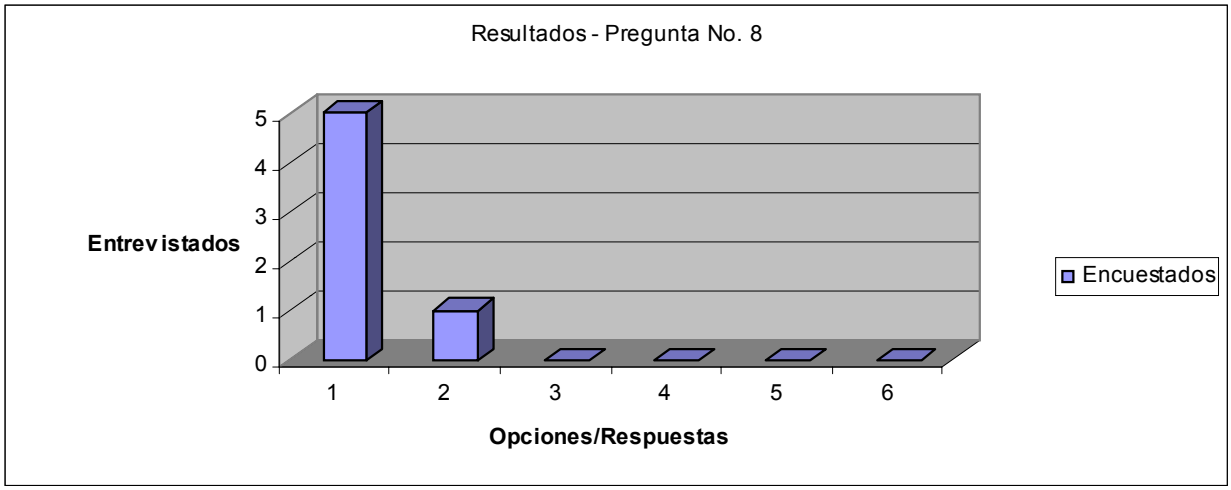
b) correctas



c) satisfacen sus necesidades



8. ¿Le gustaría contar con una herramienta coma ésta?



7.4 RESULTADOS DE LAS PREGUNTAS ABIERTAS

3a. En caso de haber encontrado dificultades al capturar políticas de seguridad, ¿Cuáles son?

Dos de los encuestados coinciden en que la captura de políticas de seguridad es un trabajo algo tedioso. Por otro lado, al capturar la acción de una política el segundo nivel de sub-ventanas confunde al usuario, ya que no se ve claramente la opción que debe seleccionarse.

4a. En caso de no estar de acuerdo con la pregunta 4, ¿Cuáles opciones del prototipo esperaba o hacen falta?

1. El prototipo debe evaluar las políticas capturadas en tiempo real, (validarlas conforme se van agregando sin tener que abrir el demostrador).
2. Interpretar los resultados arrojados por *Otter*.

7a. ¿Qué beneficios encuentra a la herramienta?

1. La generación rápida de políticas de operación.
2. La capacidad de determinar la inconsistencia de políticas.
3. El ahorro de tiempo en las actividades que rodean a la administración de políticas.

9. ¿Qué cosas cambiaría o eliminaría?

1. Son innecesarios los nodos gráficos, es suficiente con las sub-ventanas ya que mediante éstas se lleva a cabo la captura de las políticas.
2. El manejo de las políticas generales de seguridad debe ser más flexible.
3. La interfaz gráfica debe permitir la captura de políticas para cualquier recurso de la empresa, no solo para la información.

10. ¿Opiniones o sugerencias del prototipo administrador de seguridad?

1. La herramienta propuesta representa una solución óptima al problema del desarrollo y mantenimiento de políticas de seguridad.
2. La herramienta debe probarse con una base de datos grande para evaluar su rendimiento, y conocer su flexibilidad al capturar políticas.
3. Como parte de la planeación del documento de políticas de seguridad debería proporcionarse la opción de generar el análisis de riesgos de los recursos de la empresa.

7.5 CONCLUSIONES

En este capítulo hemos mostrado un pequeño experimento del mundo real que además de evaluar el trabajo realizado hasta el momento, nos proporciona una serie de ideas que deberán tomarse en cuenta para desarrollos posteriores (modificaciones) de la herramienta aquí presentada.

Los resultados arrojados por los cuestionarios aplicados a una mínima parte de la población para la cual va dirigido el proyecto, son alentadores en materia de seguridad. Demuestran que el prototipo administrador de políticas de seguridad cumple con los objetivos planteados al inicio del proyecto que son, proporcionar a los usuarios una interfaz gráfica para capturar políticas de seguridad correctamente tanto sintáctica como semánticamente así como la validación formal de las mismas.

Como todo trabajo inicial tiene una serie de limitaciones como son el enfoque único a la información, la falta de interpretación de los resultados obtenidos por *Otter*, y la validación de políticas en tiempo real, mismas que serán eliminadas en trabajos futuros.

En el siguiente apartado, se dará las conclusiones generales y los trabajos futuros propuestos para la continuidad de este trabajo.

8 CONCLUSIONES Y TRABAJO FUTURO

Al inicio del presente trabajo se mostró la importancia de contar con un conjunto de políticas de seguridad dentro de una organización para proteger sus recursos importantes. Nos enfocamos en la información, debido a que diversos autores de seguridad mencionan a este recurso como el de mayor importancia para continuar con las operaciones comerciales de la empresa.

Las políticas son el punto de partida para establecer una infraestructura apropiada. Se mencionó también la necesidad de diseñar un análisis de riesgos de la información y de los sistemas de información antes de comenzar con el desarrollo de las políticas. Desafortunadamente pocas organizaciones tienen un documento de políticas de seguridad y algunas de ellas no garantizan su validez. A partir de este hecho surge la necesidad de un herramienta que generé políticas y les de mantenimiento.

Administrar políticas de seguridad es una tarea complicada, hasta la fecha no existen sistemas que ayuden en la realización de dicha tarea, de ahí que el trabajo presentado en esta tesis es la pauta para investigaciones futuras. Nuestra contribución se enfoca solo a una parte del problema de administrar políticas de seguridad, debido al tiempo y los recursos con los que se contaba, sin embargo proporcionan herramientas suficientes para concluir la importancia de un sistema como el presentado en capítulos anteriores.

El desarrollo de la interfaz gráfica involucró un trabajo sumamente ingenioso y el cual consumió la mayor cantidad de tiempo, ya que no se ideaba la forma más sencilla para capturar una política de seguridad. Una vez que se pensó en la política como un enunciado con los elementos: sujeto, verbo y predicado (sujeto, acción y objeto), algunos Doctores en Ciencias Computacionales del

Tecnológico de Monterrey con especialidad en gráficas computacionales dieron la opinión de utilizar *wizards* o pequeñas ventanas que van surgiendo con una serie de opciones fijas (posibles valores de cada elemento de una política), de tal forma que el usuario sea limitado para evitar errores en la captura. Para dar flexibilidad a las opciones de los wizards se creó una base de datos que puede ser manipulada a voluntad del usuario según lo requiera.

La traducción de las políticas a sintaxis de *Otter* fue un trabajo exhaustivo en el que se investigaron y estudiaron brevemente analizadores léxicos sin llegar a un resultado debido a que la mayor parte del tiempo lo consumió el desarrollo de la interfaz gráfica. A partir de ese antecedente se decidió programar un analizador en java que funciona exclusivamente para este objetivo.

Para ejemplificar el trabajo desarrollado se utilizaron dos prácticas una de ellas tomada del curso de seguridad computacional 1. Esa sección concluye con resultados positivos sobre el manejo del prototipo, sin embargo los datos utilizados en la base de datos son relativamente pocos y es conveniente que para evaluaciones futuras del trabajo se utilice una cantidad de información grande, comparable con los datos que se manejan en una empresa mediana-grande.

La evaluación del prototipo realizada por algunas personas resultó sumamente útil para concluir que para toda organización sin importar el giro de su negocio es necesario contar con un administrador de políticas de seguridad. Además esta evaluación nos proporcionó ideas y sugerencias para trabajos futuros.

Como trabajo futuro existen varios puntos a tratar entre los que encontramos primordialmente:

a) la interpretación de los resultados obtenidos por el demostrador de teoremas una vez que fueron introducidas las políticas formalizadas; b) la validación de políticas en tiempo real evitando que el usuario tenga que oprimir un botón o ejecutar cualquier proceso, además podría eliminarse el trabajo de interpretar los resultados de *Otter* ya que las políticas son evaluadas conforme se van capturando y si *Otter* encuentra la cláusula vacía, la política involucrada en el error sería la última que fue capturada.

Otras características importantes que deben considerarse para investigaciones posteriores son:

- Proporcionar los elementos suficientes para analizar los recursos de la empresa y diseñar fácilmente el análisis de riesgos, y en base a ese estudio ayudar al usuario en el diseño de sus políticas de seguridad.
- Administrar políticas de seguridad para proteger todos los recursos de la empresa (no solo la información).
- Dar a conocer el documento de políticas de seguridad (o solamente algunas de ellas) a las personas indicadas con solo oprimir un botón.

Para finalizar el trabajo futuro, el prototipo puede implementar modelos de seguridad[10] mediante la definición de elementos relevantes, como son: a) datos con restricción; b) datos sin restricción; c) procedimientos de transformación, mismos que corresponden a entidades autorizadas para modificar un dato con restricción; d) procedimientos de verificación de integridad, los cuáles controlan que todos los datos con restricción del sistema se apeguen a la política de integridad; e) reglas de certificación que especifiquen las validaciones que deben realizarse sobre las entidades del sistema antes de que sea utilizado y; f) reglas de implementación que especifiquen los controles que deben realizarse por el sistema sobre las acciones de los usuarios.

Con esto concluimos que aún existen muchas cosas por hacer para obtener la mejor solución al problema que enfrentan muchas empresas, que es la administración de políticas de seguridad. Es evidente que cada día que pasa, las necesidades aumentan y se vuelven más complejas, por lo tanto, es difícil asegurar que llegará la herramienta perfecta que abarque las necesidades de las empresas en materia de políticas de seguridad, más aún, es muy probable que pasen algunos años antes de que se llegué al sistema más óptimo que administre políticas de seguridad.

[10] Un modelo de seguridad es una formalización de una política de seguridad. Esta formalización permite probar propiedades para verificar si se está haciendo respetar la política de seguridad.

REFERENCIAS

[1] WALKER, K.M. CROSWHITE C.L., *Computer Security Policies and SunScreen-Firewalls*. Sun Microsystems Press: Prentice Hall, 1998. 1-17 p.

[2] PELTIER, T. R. *Information Security Policies, Procedures, and Standard: Guidelines for Effective Information Security Management.* Auerbach Publications: imprint of CRC Press LLC, 2002. 1-11, 21-50 p.

[3] http://www.information-security-policies.com/download.htm

[4] PELTIER, T. R. *Information Security Policies and Procedures: a Practitioner's Reference*, Auerbach, c1999.

[5] KANGASLUOMA, M. Policy Specification Languages. Department of Computer Science, Helsinki University of Technology, 1999.

[6] HOAGLAND, J.; PANDEY, R.; LEVITT, K. Specifying and Enforcing Policies Using LaSCO: the Language for Security Constraints on Objects, *Policy Workshop*, 1999.

[7] KRSUL, I.; SPAFFORD, E.; TUGLULAR, T. A New Approach to the Specification of General Computer Security Policies. COAST *Techical Report 97-13*., 1998, West Lafayette, IN 47907–1398.

[8] HOWARD, P.D. The Security Policy Life Cycle: Functions and Responsibilities, Information Security Management Handbook, Edited by Tipton & Krause, CRC Press LLC, 2003.

[9] BRUCE.A. *Library Security and Safety Handbook: Prevention, Policies and Procedures*, American Library Association, 1999.

[10] DAVIS E., *Representation of Commonsense Knowledge*, Courant Institute for Mathematical Sciences, 1990.

[11] McCUNE, W. Otter 3.3 Reference Manual, Technical *Memorandum No. 263*, 2003, ANL/MCS-TM-263.

[12] KANGASLUOMA, M. Expressing Security, Helsinki University of Technology, Departament of Computer Science, Article in T-110.501 Seminar on Network Security, 2001.

[13] SMULLYAN, R. *First-order logic*, New York : Dover Publications, c1995.

[14] MONIN J., *Understanding Formal Methods*, Springer, 2000.

[15] HOAGLAND, J.; PANDEY, R.; LEVITT, K. Security Policy Specification Using a Graphical Approach, *Technical Report CSE-98-3*, 1998.

[16] http://www.symantec.com/region/mx/enterprisesecurity/content/expert/LAM_1155.html

[17] http://www.symantec.com/region/mx/enterprisesecurity/content/expert/LAM_1179.html

[18] http://www.symantec.com/region/mx/enterprisesecurity/content/expert/LAM_1165.html

[19]http://www.symantec.com/region/mx/enterprisesecurity/content/framework/LAM_1128.html

[20] http://www.monografias.com/trabajos11/seguin/seguin.shtml

[21] http://www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5007/0300.HTM