INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

ADMINISTRADOR DE POLÍTICAS DE SEGURIDAD, 2NDA VERSIÓN.

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

ANNA VALESKA ALVAREZ BEJAR

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

JUNIO 2006

CONTENIDO.

RESUMEN.

Dado que una política de seguridad es pilar para el control del riesgo informático en una organización, es necesario que sea precisa, concreta y consistente. Desafortunadamente pocas organizaciones tienen un documento de políticas de seguridad y algunas de ellas no se aseguran que dichas políticas cumplan con el objetivo de proteger sus recursos informáticos. A partir de este hecho surge la necesidad de una herramienta que genere políticas y les dé mantenimiento.

Con el propósito de administrar y aplicar efectivamente una política de seguridad, M en C Karen García Gamboa ha desarrollado una herramienta: *Administrador de políticas de seguridad organizacionales*. Si bien el administrador de políticas de seguridad organizacionales ha impactado en su propósito, un análisis arrojó 3 limitaciones importantes: i) falta de una semántica clara y precisa de políticas de seguridad; ii) razonamiento incompleto en el análisis de políticas, especialmente cuando no hay evidencias de que son inconsistentes; iii) una interfaz usuario-máquina que en algunos módulos requiere interacciones tediosas y que no permite una revisión integral de las políticas capturadas.

En la presente tesis, extendemos ésta herramienta desarrollando un nuevo administrador de políticas, el cual subsana las limitaciones anteriores: i) aplicando un lenguaje similar al lenguaje $\mathcal{L}_7$, definido por Halpern y Weissman; ii) complementando el análisis de (in)consistencia usando Mace (un constructor de modelos para la lógica de primer orden); iii) ampliando el número de propiedades que podemos analizar, incluyendo si la organización es fiel al modelo de seguridad Bell-La Padula; y por último iv) realizamos una serie de mejoras y de nuevas capacidades a la interfaz gráfica logrando con esto una interfaz más amigable y funcional.

Y por último, probamos el rendimiento del administrador de seguridad en una empresa ficticia. Hubiéramos deseado evaluar nuestra herramienta en una organización real, sin embargo, no fue posible debido a que las políticas de cualquier organización son de carácter

confidencial. No obstante, consideramos que los resultados obtenidos al evaluar el administrador de políticas son muy satisfactorios por lo que pensamos que si en un futuro el administrador de políticas pueda ser implementado en una empresa real, éste funcionará de forma correcta.

1 PROPUESTA DEL TRABAJO.

1.1 MOTIVACIÓN DEL PROBLEMA.

Los eventos mundiales recientes han generado un mayor sentido de urgencia que antes con respecto a la necesidad de tener mayor seguridad tanto física como lógica. La mayoría de las empresas actualmente han reforzado sus medidas de seguridad, pero nunca se sabe cuándo o cómo pueden estar expuestas. Un factor clave para establecer la seguridad computacional en una empresa es contar con un conjunto de políticas adecuadas [2]. Una política de seguridad es *adecuada* si contribuye en el establecimiento de las metas de seguridad (integridad, disponibilidad, confidencialidad, autentificación y no repudiación), sin obstaculizar la misión de la organización para la que fue desarrollada.

Cuando una compañía no cuenta con un documento de políticas de seguridad de información, dicha compañía 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 posibilidad de que las vulnerabilidades presentes en su plataforma tecnológica sean aprovechadas por una amenaza externa o interna.

Escribir políticas de seguridad no es una tarea sencilla, tanto porque es fácil cometer errores, las políticas pueden ser inconsistentes, ambiguas, etc., como porque consume muchos recursos humanos y materiales, que la mayoría de las veces se refleja en presupuestos muy elevados. Aunado a esto, en la práctica, pocas empresas administran adecuadamente sus políticas de seguridad, por lo que dichas políticas a menudo contienen una gran cantidad de inconsistencias e imprecisiones.

Esto ha provocado la necesidad de contar con herramientas que, no sólo disminuyan los errores de administración, sino que además fomenten la administración misma.

Un esfuerzo por lograr esto, fue el trabajo desarrollado en la tesis de M. en C. Karen García Gamboa [8], el cual consistió en el diseño de una herramienta para administrar políticas de seguridad llamada "*Administrador de políticas de seguridad organizacionales"*.

El presente trabajo surge de la problemática mencionada anteriormente y se basa en el trabajo de Karen A. García Gamboa ya que, al analizarlo, hemos encontrado que se le pueden realizar algunas mejoras que permitirán que la herramienta sea usada ya no como un prototipo sino como un administrador de seguridad.

1.2 ESPECIFICACIÓN DEL PROBLEMA.

Las principales características del administrador de políticas organizacionales incluyen:

1) Una interfaz gráfica utilizada para capturar políticas de seguridad mediante el uso de ventanas que surgen conforme se va construyendo la política.

2) Un traductor de políticas que formaliza éstas en lógica de primer orden con sintaxis de *Otter* (demostrador automático de teoremas para un lenguaje de primer orden).

3) Una liga, con una consola gráfica, a *Otter* para manipular su uso ya que éste trabaja mediante comandos del sistema operativo.

Este administrador de políticas de seguridad fue evaluado mediante una encuesta aplicada a personas que laboran en el área de sistemas de Instituto Tecnológico y de Estudios Superiores de Monterrey. Esta encuesta sirvió para demostrar que su herramienta cumple con los objetivos de proporcionar a los usuarios una interfaz gráfica para capturar políticas de seguridad correctamente así como la validación formal de las mismas. Sin embargo, dicha encuesta nos muestra que su interfaz gráfica presenta dos limitaciones principales. La primera limitación consiste en que algunos módulos que se utilizan en la captura de políticas son confusos y tediosos. Y la segunda nos muestra que el diseño de la interfaz gráfica no permite una revisión integral de las políticas de seguridad.

Por otro lado, nosotros también realizamos un análisis objetivo sobre dicha el administrador de políticas organizacionales y encontramos las siguientes dos limitaciones:

- Realiza una formalización de las políticas de seguridad, con una semántica no clara e imprecisa.

- Se tiene un razonamiento incompleto en el análisis de las políticas, especialmente cuando no hay evidencia de que son inconsistentes.

Debido a estas limitaciones (las limitaciones obtenidas de la encuesta y de nuestras observaciones), surge la presente tesis y se plantea extender la herramienta para la administración de políticas de seguridad, que de ahora en adelante llamaremos simplemente *"administrador de políticas"*. El administrador de políticas resuelve estas limitaciones y además se le han adicionado nuevas capacidades que permitirán una mejor administración de las políticas de seguridad.

Es importante mencionar que una nueva capacidad de nuestro administrador de políticas es que se incluye el modelo de seguridad de Bell-La Padula. Esto tiene como objetivo el permitir al administrador de seguridad investigar si sus políticas se ajustan o no a este modelo.

Para una mejor comprensión de esta tesis, en la siguiente sección se da una breve explicación de los conceptos principales de una política de seguridad. Empezaremos por definir qué es una política de seguridad, en qué consiste y sobre todo hablaremos de la importancia que tiene el implementar políticas de seguridad en una empresa.

Posteriormente, en las siguientes secciones explicaremos en qué consiste cada una de las limitantes del administrador de políticas de seguridad organizacionales y explicaremos brevemente cómo nuestro administrador de políticas las resuelve.

Al final de este capítulo comentaremos el procedimiento que utilizamos para evaluar nuestra herramienta y los resultados que obtuvimos.

1.3 MARCO TEORICO.

1.3.1 POLÍTICA DE SEGURIDAD.

El término **política de seguridad** se define como el conjunto de reglas de carácter obligatorio dadas por los responsables directos o indirectos de un sistema, que indica en términos generales qué está y qué no está permitido durante la operación general de dicho sistema [1]. Al tratarse de "términos generales", aplicables a situaciones o recursos muy diversos, es necesario refinar los requisitos de la política para convertirlos en indicaciones precisas de qué es lo permitido y lo denegado en cierta parte de la operación del sistema, lo que se denomina **política de aplicación específica** [2].

También, se puede definir una política de seguridad como "*una declaración de intenciones de alto nivel que cubre la seguridad de los sistemas informáticos y que proporciona las bases para definir y delimitar responsabilidades para las diversas actuaciones técnicas y organizativas que se requieran*" [3]

De acuerdo a la anterior definición, una política de seguridad se refleja en una serie de normas, reglamentos y protocolos a seguir, donde se definen las medidas a tomar para proteger la seguridad del sistema; pero ante todo, una política de seguridad es una forma de comunicación. Ya que siempre hay que tomar en cuenta que la seguridad empieza y termina con las personas [4].

Una política debe ser holística[1] ya que no tiene sentido proteger el acceso con una puerta blindada si a ésta no se le ha cerrado con llave. También, debe adecuarse a las necesidades y recursos y el tiempo en que se aplica no debe influir en su eficacia y eficiencia.

Una política de seguridad puede ser **prohibitiva**, si todo lo que no está expresamente permitido está denegado, o **permisiva**, si todo lo que no está expresamente prohibido está permitido. Evidentemente la primera opción es mejor que la segunda si el objetivo es mantener la seguridad de un sistema; en este caso la política contemplaría todas las

[1] La holística se refiere a la manera de ver las cosas enteras, en su totalidad, en su conjunto, en su complejidad, pues de esta forma se pueden apreciar interacciones, particularidades y procesos que por lo regular no se perciben si se estudian los aspectos que conforman el todo, por separado.

actividades que se pueden realizar en los sistemas, y el resto, las no contempladas, serían consideras ilegales.

Cualquier política ha de contemplar seis elementos claves en la seguridad de un sistema informático [5]:

- Disponibilidad: es necesario garantizar que los recursos del sistema se encontrarán disponibles cuando se necesiten, especialmente la información crítica.
- Utilidad: Los recursos del sistema y la información manejada en el mismo han de ser útiles para alguna función.
- Integridad: La información del sistema ha de estar disponible tal y como se guardó por un agente autorizado.
- Autenticidad: El sistema ha de ser capaz de verificar la identidad de sus usuarios, y los usuarios la del sistema.
- Confidencialidad: La información sólo ha de estar disponible para agentes autorizados, especialmente su propietario.
- Posesión: Los propietarios de un sistema han de ser capaces de controlarlo en todo momento; perder este control en favor de un usuario malicioso compromete la seguridad del sistema hacia el resto de usuarios.

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 [6] que son:

- 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.

Una política de seguridad debe ser fácil de entender y aplicable. 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 Además se debe asegurar que todas las políticas se escribe reúnen las necesidades de cada organización en específico.

Antes de desarrollar un documento de políticas de seguridad es necesario realizar un análisis de riesgos. En un análisis de riesgos se debe calcular la posibilidad de que ocurran ciertos sucesos y determinar cuales tienen el potencial para causar daño. Esta medición se podrá utilizar para contrastar el costo de la protección de la información en análisis contra el costo de recuperarla en caso de un incidente.

1.3.2 IMPORTANCIA DE IMPLANTAR POLÍTICAS DE SEGURIDAD EN UNA ORGANIZACIÓN.

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. Existen estudios que nos indican que las empresas que no cuentan con un documento de políticas de seguridad están expuestas en un 80% más a sufrir un ataque interno ó externo [7]. Por tal motivo, la generación de políticas de seguridad permite reducir el riesgo de pérdida de información.

El implementar políticas de seguridad en una organización permite que ésta tome el control de su funcionamiento y reduce la probabilidad de que las vulnerabilidades en las instalaciones donde reside la información sean aprovechadas causando severos daños a la organización.

1.4 LIMITACIONES DEL ADMINISTRADOR DE POLÍTICAS DE SEGURIDAD ORGANIZACIONALES.

En esta sección nos centraremos en el análisis que realizamos sobre la herramienta desarrollada en la tesis de Karen A. García Gamboa [8]. Se expondrán brevemente las mejoras realizadas. Y además la utilidad que se espera de la solución.

Para lograr una mejor explicación del tema, esta sección será divida en tres partes: la primera es referente a el aspecto lógico y matemático usado en la representación de las políticas, la segunda se refiere al razonamiento que podemos realizar sobre las políticas de seguridad, y por último, en la tercera sección hablaremos sobre la interfaz gráfica del administrador de políticas.

1.4.1 FORMALIZACIÓN DE POLÍTICAS.

Analizando el administrador de políticas organizacionales realizado por M en C Karen García Gamboa observamos que formaliza los sujetos y los objetos de las políticas de seguridad utilizando la siguiente regla: Sustantivo_XXX(x) por ejemplo: Informacion_Sistemas(x) que quiere decir que: "*x* es información del departamento sistemas". Este tipo de formalización no nos permite evaluar al sujeto o el objeto de la política en sí, debido a que no están siendo considerados como parte de la estructura.

Para subsanar esta limitante, primero nos dimos a la tarea de realizar una investigación para ver si existen otros lenguajes que nos permitan expresar las políticas de seguridad con una sintaxis más apropiada en donde se acepte un modelo arbitrario de política de acceso de la información.

Además, analizamos el trabajo de Halpern y Weissman [14] en el cual se muestra un lenguaje para especificar políticas cuya formulación es más sencilla y nos permite evaluar todas las variables que tiene la expresión lógica.

En el capitulo 2 de esta tesis se describe más a detalle los lenguajes encontrados en nuestra investigación. Además, se explica el lenguaje de formalización de Halpern y Weissman, mostrando qué sub-conjunto de la lógica de primer orden estamos usando y cuál es su complejidad computacional.

En el siguiente capítulo se expone cómo usando un lenguaje similar al lenguaje de especificación elegido se pueden expresar las políticas de seguridad relacionadas con el uso de sus recursos.

1.4.2. RAZONAMIENTO INCOMPLETO EN EL ANÁLISIS DE POLÍTICAS.

El administrador de políticas de seguridad organizacionales desarrollada por M en C Karen García Gamboa únicamente realizaba la validación de políticas utilizando Otter. Sin embargo, en algunos casos, Otter puede tardar mucho tiempo (infinito) en responder debido a que no cuenta con los elementos necesarios para hacerlo.

Para resolver esta limitante, se le ha adicionado a la herramienta la utilización de Mace, un constructor de modelos para lógica de primer orden. Mace construirá un modelo si las políticas son consistentes. Por lo tanto, el utilizar Mace en nuestra herramienta nos permite responder con mayor precisión cuándo una política de seguridad es o no consistente.

Al contar con estas dos herramientas (Otter/Mace) nuestro administrador de políticas va a realizar las siguientes validaciones: i) verificar qué sujeto(s) tiene(n) ciertos permisos sobre un objeto en especial, ii) verificar si un sujeto puede o no realizar una acción *x*, iii) verificar si las políticas escritas son consistentes y iv) verificar si las políticas escritas satisfacen el modelo de Bell – La Padula.

En el capítulo 4 se explica más a detalle cómo se realizan estas cuatro validaciones usando ambas herramientas (Otter/Mace).

1.4.3 MEJORAS Y CAPACIDADES ADICIONALES EN LA INTERFAZ GRÁFICA.

Como se mencionó anteriormente, la primera limitante que arrojó la encuesta realizada al administrador de políticas organizacionales [8], fue que la captura de las políticas es complicada y tediosa. Esto se debe a que para escribir las políticas se debe seleccionar mediante la interfaz gráfica cada uno sus elementos (sujeto-acción-objeto). Sin embargo, antes de seleccionar cada uno de estos elementos se debe primero dibujar un grafo representativo, lo que genera pérdida de tiempo en la creación de políticas. En nuestra herramienta, se remedia este problema, eliminando la necesidad de dibujar el grafo para escribir la política.

La segunda limitante que se encontró fue que la consulta, modificación y eliminación de políticas así como de los elementos necesarios para construirlas (departamentos, empleados, etc.) resulta un trabajo difícil para el usuario, ya que estas operaciones se encontraban en menús separados en la interfaz. Nuestro administrador de políticas resuelve lo anterior al permitir que el usuario realice dichas operaciones en una misma ventana, sin necesidad de navegar en diferentes menús.

En el Capítulo 5 de esta tesis, se habla más a detalle sobre las mejoras realizadas a la interfaz y de la serie de capacidades adicionadas. Entre otras tenemos la capacidad de revisión y edición de la política de seguridad; una mejor navegación a través de ventanas sencillas para construir una política; la generalización de políticas y el cambio del manejador de bases de datos para poder manipular bases de datos grandes.

1.5 EVALUACIÓN DEL RENDIMIENTO DEL ADMINISTRADOR DE POLÍTICAS.

Para probar el rendimiento del administrador de seguridad, desarrollamos tres ambientes. El primero fue un ambiente juguete, constituido por una empresa con 7 empleados y 4 departamentos. El segundo fue una empresa ficticia de mayor tamaño la cual consistió en 114 empleados y 16 departamentos. Y el tercer ambiente es un ambiente militar. Con este último ambiente examinamos el desempeño de la herramienta al implementar el modelo de seguridad de Bell-La Padula.

Para evaluar el rendimiento del administrador de políticas medimos los tiempos de procesamiento que nos arrojó Otter y Mace al evaluar las políticas de estos tres ambientes. Al realizar un análisis de estos tiempos de procesamiento pudimos observar que se incrementan conforme la organización y su número de políticas son mayores. Por ejemplo, al analizar el ambiente juguete contra la empresa ficticia, vemos que los tiempos de procesamiento aumentan de microsegundos a segundos.

En el capítulo 6 se explican más a detalle las pruebas que realizamos para evaluar el rendimiento de nuestra herramienta y se reportan los resultados. Además, en este mismo capítulo se exponen las conclusiones a las que llegamos.

1.6 CONCLUSIONES.

Concluyendo, la falta de administración 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, por lo que una política de seguridad adecuada representa un elemento fundamental dentro de toda organización para establecer la seguridad de sus recursos informáticos valiosos.

El subsanar todas las limitantes que encontramos en el administrador de políticas de seguridad organizacionales desarrollado por [8], nos permitió desarrollar una nueva herramienta con la cual podemos realizar un análisis más extenso de las políticas de seguridad que se implementarán en una organización.

2 LENGUAJE Y HERRAMIENTAS USADAS EN EL DESARROLLO DE LA TESIS.

2.1 INTRODUCCIÓN.

El objetivo del presente capítulo es explicar el lenguaje y las herramientas que utilizamos para resolver las limitantes que encontramos en el administrador de políticas de seguridad organizacionales presentado por [8].

De acuerdo a lo que se expuso en el capítulo anterior, nos dimos a la tarea de realizar una investigación para encontrar un lenguaje que tuviera una semántica precisa y nos permitiera expresar políticas de seguridad. En el presente capítulo exponemos los lenguajes encontrados.

Además, realizamos un análisis del trabajo presentado por Halpern y Weissman [14] y mostramos que el lenguaje expuesto en este trabajo es el mejor para formalizar las políticas de seguridad.

Por otro lado, para tener un razonamiento más completo en el análisis de las políticas de seguridad, en el presente capítulo se da una introducción de Mace, explicándose claramente qué es, cómo funciona y para qué sirve.

Y por último, debido a que implementamos en el administrador de políticas el modelo de Bell- La Padula, en este capítulo, exponemos qué es un modelo de seguridad y en qué consiste el modelo de Bell- La Padula.

2.2 LENGUAJES PARA ESPECIFICAR POLÍTICAS DE SEGURIDAD.

Existen varios lenguajes que nos permiten expresar políticas de seguridad, en el trabajo realizado por Karen García Gamboa vemos que realizó una investigación de algunos de estos lenguajes con la finalidad de ver si se podían utilizar para expresar políticas de seguridad y sobre todo, si le servían para realizar un razonamiento de las políticas. Sin embargo, los lenguajes presentados por [8] no sirvieron para lograr este objetivo.

De igual forma nosotros nos dimos a la tarea de investigar también algunos lenguajes que nos permitan expresar estas políticas. En las siguientes secciones se describen los lenguajes encontrados.

2.2.1 EXTENSIBLE RIGHTS MARKUP LANGUAGE (XRML).

XrML (Extensible rights Markup Language) [10] es una propuesta al manejo de derechos estándares digitales (DRM) ó servicios Web. Este lenguaje soporta el proceso automático de grabar reglas implícitas en el hipertexto y ayudar a la comprensión del navegador. Su principal aplicación es en la compra y venta sobre Internet ya que ayuda a proteger el contexto de accesos no autorizados. Compañías como ContentGuard, Hewlett-Packard, Microsoft, Reuters y VeriSingn entre otras han promovido el uso de XrML

XrML fue desarrollado a partir del lenguaje *Digital Property Rights Language (DPRL)* versión 2.0, el cual a su vez está basado en el lenguaje LISP desarrollado por el centro de investigación de Xerox. DPRL es un lenguaje cuyo objetivo es definir reglas de acceso y procedimientos para los servicios Web. XrML cumple con este objetivo y además, es un lenguaje más extensible e interoperable que DPRL ya que permite crear estándares y se adapta fácilmente a los cambios.

XrML requiere de tres componentes: *Rule Identification Markup Language (RIML), Rule Structure Markup Language (RSML) y Rule Triggering Markup Language (RTML).*

En Rule Identification Markup Language (RIML) el meta-dato expresado en RIML podrá ser identificado para la existencia de reglas implícitas en el hipertexto en la Web. La asociación formal representa una estructura de reglas que deberán también ser especificadas.

En cambio, en Rule Structure Markup Language (RSML) las reglas tienen que ser representadas en una estructura formal para ser procesadas en la Web. Finalmente, en Rule Triggering Markup Language (RTML) se definen las condiciones en que ciertas reglas serán ejecutadas. Así que el RTML es grabado en la aplicación tal como los sistemas expertos, agentes de software, etc.

2.2.2 OPEN DIGITAL RIGHTS LANGUAGE (ODRL).

ODRL [11] es un lenguaje para la gestión de derechos y permisos sobre contenidos digitales. Soporta gran cantidad de idiomas con los que puede expresar los términos y condiciones de estos permisos. Este lenguaje es extensible y tiene un diccionario para expresiones y condiciones sobre un contexto.

ODRL provee mecanismos flexibles e interoperables para soportar de forma transparente e innovadora el uso de recursos digitales en la publicación, distribución y consumo de medios a través de todos los sectores incluyendo el sector educación, entretenimiento, comunicación móvil y software. ODRL también soporta la protección y especificaciones en su contexto digital.

Este lenguaje ha sido oficialmente aceptado en la Alianza Móvil Abierta (OMA) como el lenguaje estándar para expresar políticas en un contexto móvil. ODRL está basado en un modelo en el cual su objetivo principal es la búsqueda y análisis de requerimientos específicos tales como aspirar a ser compatibles con una amplia comunidad móvil [11].

De acuerdo a lo expuesto anteriormente, vemos que los lenguajes investigados no tienen una semántica formal y cálculo por lo que no nos sirven en la herramienta de seguridad. Sin embargo, el lenguaje de lógica de primer orden si cumple con estas características.

Por tal motivo, tanto en el trabajo de Karen García Gamboa [8] como en el nuestro, optamos por formalizar las políticas utilizando el lenguaje de lógica de primer orden. No obstante, como ya se mencionó anteriormente, una limitante que encontramos en el lenguaje utilizado por M en C Karen García fue que la semántica no era muy clara y precisa, lo que no permitía realizar un análisis completo de las políticas de seguridad. Para resolver lo anterior, nos basamos en el trabajo realizado por Halpern y Weissman [14]

En la siguiente sección se describe en qué consiste el trabajo de Halpern y Weissman y se muestra como este lenguaje es mejor que la formalización utilizada en el administrador de políticas presentado por Karen García Gamboa.

2.3 TRABAJO PRESENTADO POR HALPERN Y WEISSMAN.

El objetivo principal del trabajo de Halpern y Weissman es describir un lenguaje que nos permita expresar políticas de seguridad utilizando un fragmento de la lógica de primer orden. Halpern y Weissman plantean como objetivos parciales que dicho lenguaje debe ser: a) lo suficientemente expresivo para capturar en forma fácil y natural dichas políticas y b) lo suficientemente tratable como para permitir realizar preguntas interesantes sobre las políticas y que puedan responderse en un tiempo razonable. Por lo tanto, dicho lenguaje debe poder ser usado fácilmente por personas que no requieran conocimientos previos de lógica de primer orden.

En su artículo se muestra que la verificación de las políticas se podrá realizar de dos formas. Primero se verificará si dado un conjunto de políticas donde existen acciones permitidas y denegadas tiene concordancia con su *ambiente*, entendiendo por ambiente todos los hechos relevantes y verdaderos que crean el contexto en donde las políticas se aplican. En otras palabras, si se tiene un conjunto de políticas $S\ \{P_1, P_2, P_3, ...,Pn\}$ y E es su ambiente, la conjunción de E con S $(E \wedge S)$ debe ser válida.

Segundo, se podrá verificar si el conjunto de políticas de un sujeto en particular es consistente con el total de políticas, por ejemplo, supongamos que Andrea es estudiante de la universidad y está escribiendo políticas sobre un nuevo programa de la universidad. Si la unión de sus políticas con las políticas de la universidad es consistente, entonces sabemos que las políticas escritas por Andrea no se contradicen con el conjunto de políticas de la universidad.

Para escribir las políticas Halpern y Weissman utilizan el predicado *Puede(t,t')* donde t y t' son términos para denotar al sujeto y las acciones realizadas respectivamente, para las políticas válidas y $\neg$ *Puede(t, t')* para las políticas denegadas.

En general, si se traducen las políticas al lenguaje de lógica de primer orden usando el predicado *Puede(t,t')*, y se desea realizar alguna de las dos tareas de razonamiento que se describieron anteriormente, Halpern y Weissman exponen que las respuestas que se obtienen no son satisfactorias. Esto se debe a que como sabemos la validación de un problema en lógica de primer orden es indecidible.

Para poder realizar un análisis más completo de las políticas, como por ejemplo, validar la consistencia de las políticas entre otras, se requiere que el lenguaje de lógica que sea utilizado sea decidible. Para lograr esto, Halpern y Weissman exponen tres características importantes que debe tener el lenguaje que se utilice.

La primera característica consiste en que para que se pueda determinar si una política implica permisos o no, su ambiente (contexto) no debe ser vacío. Lo anterior es correcto, ya que para verificar las políticas necesitamos conocer su ambiente. Por tal motivo, Halpern y Weissman definen a un *ambiente estándar* como un ambiente que es lo suficientemente expresivo para capturar toda la información representativa y verdadera del contexto en cuestión.

Dicho ambiente estándar puede estar divido en dos conjuntos E_0 y E_1. E_0 es llamado *ambiente básico [14]* el cual es la conjunción de literales fijas y no contienen el predicado *Puede(t,t')* mientras que E_1 contiene fórmulas con cuantificadores universales.

La segunda característica se refiere a que las políticas utilicen sólo cuantificadores universales. Esto se debe a que las políticas permiten que un individuo haga una acción basada en los atributos de ese individuo y la acción.

Y la tercera característica define a una política de seguridad como una oración de la forma:

$$\forall x \ldots\ldots xm ((C \rightarrow (\neg)\ Puede(t,t'))$$

Donde *C* representa las literales fijas que no pueden contener el predicado *Puede(t,t')*, *t* es un término que representa al sujeto y *t'* un término que representa la acción que realiza. Una política de la forma anterior es llamada una política estándar [14]. De acuerdo con Halpern y Weissman, una política estándar es en general suficiente para expresar cualquier tipo de política de seguridad.

Al realizar lo anterior, Halpern y Weissman obtienen un lenguaje que ya es decidible pero aún no es tratable. Un primer paso para obtener un lenguaje tratable es considerar sólo ambientes estándares. Sin embargo, en lo que se refiere a las políticas aún es necesario realizar una restricción más [14].

La restricción que deben tener las políticas es la *bipolaridad*. Una literal ℓ se dice que es bipolar en una fórmula F, escrita en la forma normal conjuntiva, si ℓ está en F y si hay otra literal ℓ' en F tal que ℓ y $\neg\ell'$ son unificables. Esto es, $\exists\sigma.\ (l' \equiv \neg l')\sigma$, donde σ es una sustitución de variables. Si $(l' \equiv \neg l')\sigma$ sigue de un conjunto de oraciones de igualdad de E entonces ℓ se dice que tiene una bipolaridad relativa en la fórmula F.

Tomando esta restricción, Halpern y Weissman definen un lenguaje el cual llaman $\mathcal{L}_7$ [14]. Este lenguaje ya es tratable y puede ser utilizado para expresar políticas de seguridad. Dicho lenguaje es descrito en el siguiente teorema:

Teorema 4.3[14]: Sea ϕ un lenguaje que contiene *Puede(t,t')* (y la posibilidad de otros predicados, constantes y funciones). Sea $\mathcal{L}_7$ el lenguaje que contiene todas las fórmulas cerradas F *en* $\mathcal{L}^{F0}(\phi)$ de la forma $E_0 \wedge E_1 \wedge P_1 \wedge \ldots \wedge P_n \rightarrow$ *Puede(t, t')*. En la cual, E_0 es un ambiente básico, E_1 es la conjunción de fórmulas universales, $P_1 \wedge \ldots \wedge P_n$ es la conjunció n de políticas estándares y t es el sujeto que se le permite o niega realizar una acción t'. $\mathcal{L}_7$, tiene las siguientes restricciones:

a.- E_0 tiene m constantes.
b.- La conjunción de $E_1 \wedge P_1 \wedge \ldots \wedge P_n$ no puede tener desigualdades en los antecedentes,
c.- Cada conjunción en $E_1 \wedge P_1 \wedge \ldots \wedge P_n$ tiene a lo más una literal que tiene una bipolaridad relativa en $E_1 \wedge P_1 \wedge \ldots \wedge P_n$ a los argumentos de igualdad en E_0.

Se determina la validación en el tiempo de la siguiente forma $O(|E_1 \wedge (P_1 \wedge \ldots \wedge P_n)|log|E_1 \wedge (P_1 \wedge \ldots \wedge P_n)| + b|C_l| + T)$ donde b es el número de pares bipolares en f relativos a las igualdades [14] en E_0, C_l es la mayor conjunción (unión) en f, T se define: si cada literal que aparece en la conjunción $E_1 \Lambda P_1 \wedge \ldots \wedge P_n$ tiene al menos una variable que no aparece en el predicado *Puede(t,t')* entonces T es igual a $(|E_0| + m(|E_1 \Lambda P_1 \wedge \ldots \wedge P_n| + b|C_l|))log|E_0|$. En caso contrario T es $(|E_0| + m^k(|E_1 \Lambda P_1 \wedge \ldots \wedge P_n| + b|C_l|))log|E_0|$, donde k es número máximo

de variables que aparecen en una simple conjunción en donde no aparecen argumentos en el predicado *Puede(t,t')* .

2.3.1 LENGUAJE SELECCIONADO PARA LA FORMALIZACIÓN DE POLÍTICAS EN EL ADMINISTRADOR DE POLÍTICAS.

Después de realizar el análisis del trabajo de Halpern y Weissman, expuesto en la sección anterior. Concluimos que el lenguaje en que se basa nuestra herramienta para formalizar las políticas de seguridad es un lenguaje que cumple con las condiciones del lenguaje $\mathcal{L}_{7,}$ excepto con la condición 2 Y 3.

Lo anterior se debe a que nosotros estamos forzando a incluir desigualdades en los antecedentes de las políticas para completarlas y poderlas validar. Además de que manejamos más de una desigualdad en el ambiente E_0 y tenemos muchos pares bipolares en E_1. Dichos pares bipolares los podemos encontrar en las abreviaturas que usamos en los diferentes modos de acceso.

El administrador de políticas organizacionales realizado por M en C Karen García Gamboa validaba la consistencia de las políticas utilizando Otter (consultar apéndice A). No obstante, en algunas ocasiones Otter no puede darnos una respuesta por falta de evidencias.

Para resolver este problema, en nuestra herramienta hemos incluido la utilización de Mace. Para una mejor comprensión de este tema, en la siguiente sección, se explica qué es Mace, cómo funciona y para que sirve.

2.4 MACE.

Mace (Modelos y contraejemplos) es un programa que construye un modelo de lógica de primer orden. Este normalmente se utiliza como un constructor de modelos ya que por ejemplo con Otter se busca una prueba y con Mace se construye un modelo de un conjunto de teoremas. Ambos programas aceptan el lenguaje de lógica de primer orden. A continuación se describe el funcionamiento general de Mace [15]:

Con tres clases de entradas trabaja Mace. Primero cláusulas ó fórmulas las cuales contienen los elementos necesarios para que Mace construya un modelo. Segundo, comandos especiales en el archivo de entrada que limiten el modelo a buscar. Y tercero, se tiene la opción de comandos en línea para limitar la búsqueda.

Mace lee un archivo de entrada y toma las fórmulas y cláusulas que se encuentren en la lista de usable, sos, demoduladoras y pasivas tal y de igual manera que Otter[2]. Las fórmulas las transforma en cláusulas para poder buscar un modelo que satisfaga a este conjunto de cláusulas.

Uno de los parámetros que Mace necesita para construir el modelo es el tamaño del dominio. Se denomina "dominio" a un fragmento de todo el universo, el cual es acotado para encontrar un modelo. Mace transforma las cláusulas de entrada en un problema proposicional equivalente. Y relaciona cada elemento con el dominio, obteniéndose el universo de Herbrand. Para comprender mejor qué es el universo de Herbrand partiremos explicando en qué consiste el teorema de Herbrand en la siguiente subsección.

2.4.1 EL TEOREMA DE HERBRAND.

El teorema de Herbrand es muy importante en lógica simbólica. Es la base para los más modernos procedimientos de prueba, en la demostración mecánica de teoremas. Este teorema tiene la finalidad de probar si un conjunto de cláusulas es insatisfactible, para ello sólo considera las interpretaciones sobre el universo de Herbrand. Si el conjunto de cláusulas es falso, bajo todas las interpretaciones, entonces se puede concluir que las cláusulas son insatisfactibles. Debido a que hay usualmente un número infinito de interpretaciones, se deben organizar las cláusulas en forma sistemática. Esto se puede realizar mediante el uso de árboles semánticos.

El Teorema de Herbrand se enuncia de la siguiente forma:

[2] Para mayor detalle de estos términos consultar el apéndice A de esta tesis.

Un conjunto S de cláusulas es insatisfactible si y sólo si existe un conjunto insatisfactible infinito S' de instancias de las cláusulas aterrizadas de S [12].

Un ejemplo muy sencillo y conocido es el siguiente: Sea *S*, un conjunto insatisfactible tal que *S = {P(x), ~ P(f(y))}*. Entonces de acuerdo con el teorema de Herbrand, existe un conjunto insatisfactible infinito *S´* de instancias de las cláusulas aterrizadas de *S*. Uno de estos conjuntos es *S´ = {P(f(a)),~ P(f(a))}*.

El teorema de Herbrand sugiere un procedimiento de demostración basado en la refutación. En otras palabras, dado un conjunto insatisfactible de cláusulas a probar, si existe un procedimiento mecánico que pueda generar sucesivamente conjuntos S_1..... S_n, de instancias de cláusulas en el Universo de Herbrand, y se puede verificar la insatisfactibilidad de cada uno de estos conjuntos, entonces, como lo garantiza el teorema de Herbrand, este procedimiento puede detectar en un número infinito de iteraciones la insatisfactibilidad de *S*.

Una vez que Mace obtiene el universo de Herbrand, éste es introducido al procedimiento DPLL **(**Davis-Putnam-Loveland-Logeman**)** [17]. A continuación se describe brevemente como funciona este procedimiento.

2.4.2 PROCEDIMIENTO DPLL (DAVIS-PUTNAM-LOVELAND-LOGEMAN).

Este método consiste en el uso iterativo de cuatro reglas que permiten realizar la comprobación de la insatisfactibilidad de un conjunto de cláusulas. Estas reglas permiten ir reduciendo la cantidad de cláusulas hasta llegar al punto en que se tiene una fórmula que es válida o que es insatisfactible y que contiene el mínimo número de cláusulas [26]. A continuación se describe cada una de estas reglas:

Sea *S* un conjunto de cláusulas instanciadas en el Universo de Herbrand.

1. Regla de la Tautología: Borra todas las cláusulas aterrizadas de *S* que son tautologías. El conjunto resultante *S´* es insatisfactible si y sólo si *S* lo es.

2. Regla de la Literal Unitaria: Si existe una cláusula aterrizada unitaria[3] L en S, obtenga S' de S al borrar aquellas cláusulas en S que contengan L. Si S' está vacío, S es satisfactible. De lo contrario, se obtiene un conjunto S'' de S' al borrar $\neg L$. S'' es insatisfactible si y sólo si S lo es. Note que si $\neg L$ es una cláusula aterrizada unitaria, entonces la cláusula se convierte a una cláusula vacía cuando $\neg L$ es borrada de la cláusula.

3. Regla de la Literal Pura: Una literal L en una cláusula aterrizada de S, se dice que es pura en S si y sólo si la literal $\neg L$ no aparece en ninguna cláusula aterrizada en S. Si una literal L es pura en S, borra todas las cláusulas aterrizadas que contengan L. El conjunto S' que resulta de esto, es insatisfactible si y sólo si S lo es.

4. Regla de Expansión: Si el conjunto S puede ser puesto en la forma: $(A_1 \vee L) \wedge \ldots \wedge (A_m \vee \_L) \wedge (B_1 \vee \neg L) \wedge \ldots \wedge (B_n \vee \neg L) \wedge R$, donde A_i, B_i, y R están libres de L y $\neg L$, entonces obtenemos los conjunto $S_1 = A_1 \wedge \ldots \wedge A_m \wedge R$ y $S_2 = B_1 \wedge \ldots \wedge B_n \wedge R$. S es insatisfactible si y sólo si $(S_1 \vee S_2)$ es insatisfactible, esto significa que S_1 y S_2 también lo son.

Si mediante este procedimiento Mace encuentra un modelo, entonces el problema es transformado en un modelo de lógica de primer orden.

Una de las tareas de validación que realiza el administrador de políticas consiste en verificar si las políticas siguen el modelo de seguridad de Bell – La Padula. Por tal motivo, en la siguiente sección, se explica qué es un modelo de seguridad y se explica en qué consiste el modelo Bell – La Padula

2.5 MODELOS DE SEGURIDAD.

Un Modelo de Seguridad [22] es una formalización de una política de seguridad. Esta formalización permite probar propiedades para verificar si se está haciendo respetar la política de seguridad. Existen varios modelos de seguridad como son: el modelo comercial (Clark-

[3] Una cláusula unitaria es una cláusula que contiene sólo una literal.

Wilson) [24], el modelo Brewer y Nash (Muralla China) [25] y el modelo militar (Bell-La Padulla) [23].

El modelo de Clark-Wilson propone dos categorías de mecanismos para prevenir la modificación irrecuperable e incorrecta de la información, intencionadamente o no. El primero es el denominado de transacciones correctas y el segundo es denominado separación de obligaciones.

En el modelo de Brewer y Nash (Muralla China) el acceso a los bancos de datos de una compañía es posible si no representa competencia (conflicto de interés) para la compañía que realiza el acceso. Por tal motivo, todas las compañías que no compiten están dentro de la muralla y pueden compartir su información.

El modelo de Bell-La Padula se basa en dos propiedades principales y en el nivel del sujeto para permitir o negar la lectura o escritura de un tipo de información, la cual tiene una clasificación que puede ser ultra secreta, secreta, confidencial o no clasificada.

En comparación con el modelo de Bell-La Padula, en el modelo de Clark-Wilson no se definen niveles de seguridad para la información, sino que define los programas que un usuario puede ejecutar, los cuales, a su vez, manejarán la información.

2.5.1 MODELO DE BELL-LA PADULA.

El modelo de Bell-La Padula es uno de los modelos más conocidos. Está basado en máquinas de estados finitos. El modelo consiste en la definición de un conjunto de variables de estado y funciones de transición y un estado inicial, considerado seguro.

Los componentes definidos por el modelo son:
- Sujetos: las entidades del sistema, incluidos usuarios y procesos.
- Objetos: las estructuras de información, que almacenan la información del sistema.
- Modos de acceso: la forma en que el sujeto interactúa con un objeto.
- Niveles de seguridad: cada objeto tiene una clasificación y cada sujeto un nivel.

Toda información tiene una clase de seguridad y toda persona tiene un nivel de autorización asociado.

Los niveles de seguridad para la información son ultra secreta, secreta, clasificada y no clasificada. Y el orden de importancia va de ultra secreta a no clasificada. Un estado se considera seguro si para cada tripleta (sujeto, objeto, modo de acceso) se cumplen las siguientes propiedades:

- Propiedad de simple seguridad: El nivel del sujeto debe ser igual o menor que la clasificación del objeto para que se le permita leer al objeto. En la siguiente figura se ilustra esta propiedad.

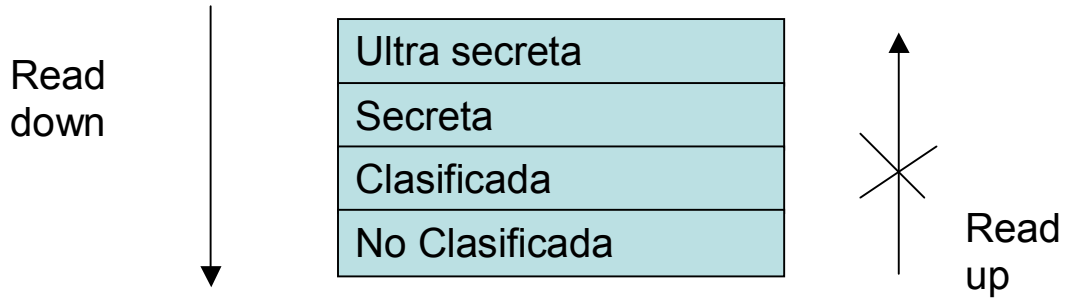

Fig. 2.1. Propiedad de simple seguridad

- Propiedad * (estrella): El nivel del objeto debe ser igual o mayor que la clasificación del objeto para que se le permita al sujeto escribir sobre el objeto. Lo anterior se muestra en la siguiente figura.

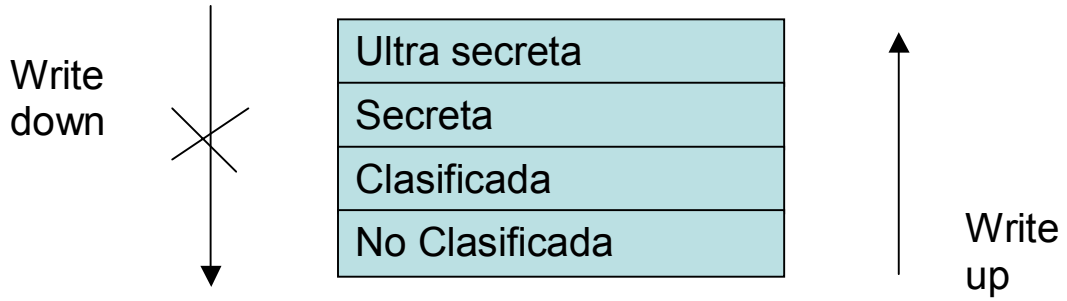

Fig. 2.2. Propiedad estrella

2.6 CONCLUSIONES.

Como se mostró en este capítulo, para poder subsanar las limitantes encontradas en el administrador de políticas organizacionales presentado por [8], nos basamos en el lenguaje

presentado en el trabajo de Halpern y Weissman para formalizar las políticas. Además incluimos a Mace en nuestra herramienta para lograr realizar un análisis completo junto con Otter de las políticas de seguridad.

También el administrador de políticas realizado en esta tesis tiene una nueva capacidad. Esta capacidad consiste en que podemos verificar si las políticas introducidas en la herramienta se ajustan o no al modelo de seguridad de Bell – La Padula.

En los siguientes dos capítulos se explica más a detalle cómo utiliza el administrador de políticas este lenguaje y cómo se complementa el análisis de las políticas al adicionar a Mace.

3 FORMALIZACIÓN DE LAS POLÍTICAS DE SEGURIDAD.

3.1 INTRODUCCIÓN.

En el presente capítulo se explicará cómo el lenguaje de especificación introducido en el capítulo 2 es suficiente para expresar un gran número de políticas. De acuerdo con Halpern y Weissman, para poder analizar las políticas de seguridad, debemos considerar que estas políticas tienen un contexto (ambiente). Por tal motivo, empezaremos en este capítulo por explicar la formalización del ambiente (E), el cual está formado por E_0 (literales fijas) y E_1 (fórmulas con cuantificadores universales) para posteriormente explicar la formalización de las políticas.

Además, como se comentó en capítulos anteriores, en el administrador de políticas se ha implementado el modelo de seguridad de Bell- La Padula. Por tal motivo, se mostrará en este capítulo, cómo el lenguaje de especificación utilizado, también nos permite formalizar las propiedades que rigen dicho modelo y las políticas que se escriban. Todo lo anterior, se explicará utilizando un ejemplo juguete.

3.2 FORMALIZACIÓN DEL AMBIENTE.

El ejemplo juguete consiste en el siguiente escenario: tomamos a la empresa "WWW" dedicada a la comercialización de productos de limpieza, dicha empresa está integrada por 4 departamentos y la dirección general. Los departamentos de esta empresa son finanzas, recursos humanos, sistemas y seguridad. A continuación se presenta el diagrama

organizacional, en el cual se pueden observar los nombres de las personas que integran la empresa y de los puestos que ocupan en la misma.

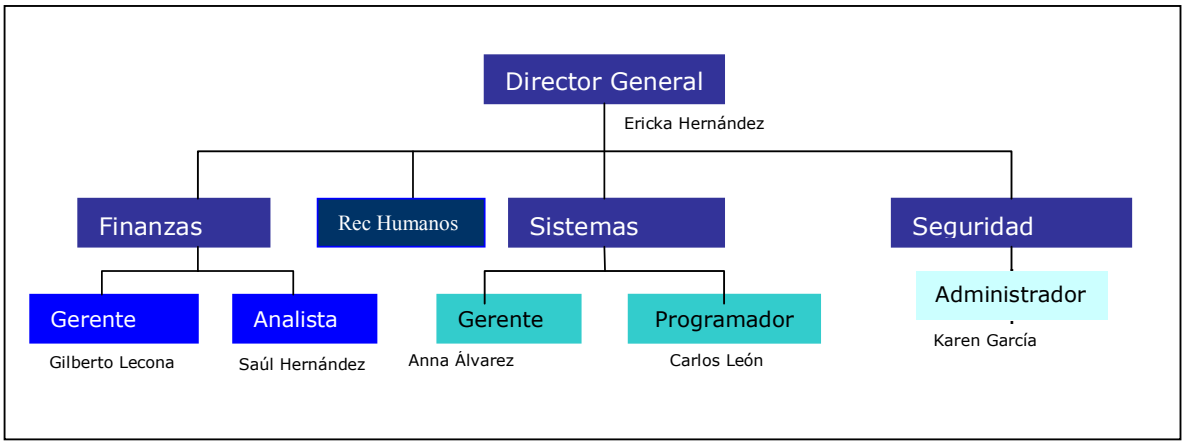

Figura 3.1 Organigrama de la empresa WWW

Este ambiente fue introducido en el administrador de políticas de seguridad, el cual se conecta a una base de datos que se diseñó en SQL Server en donde se almacenan todos los elementos del ambiente. Posteriormente, el administrador de políticas extraerá cada uno de los elementos del ambiente y los formalizará de acuerdo a los parámetros del lenguaje seleccionado al analizar el trabajo de Halpern y Weissman.

En las siguientes secciones se muestra cómo el administrador de políticas de seguridad realizado formaliza cada una de las partes del ambiente (E_0 *y* E_1).

3.2.1 FORMALIZACIÓN DE E_0.

Hemos diseñado al administrador de políticas para que formalice la tabla de departamentos anterior de la siguiente manera:

```
Departamento(Direccion).
Departamento(Finanzas).
Departamento(Seguridad).
Departamento(Sistemas).
Departamento(Recursos_Humanos).
```

En general, cualquier tabla N con m columnas, $A_1, \ldots, A_m$, es formalizada mediante la relación N, de aridad m, $N(A_1, \ldots, A_m)$.

Además de los departamentos, como se puede observar en la figura 3.1 dentro de la empresa existen roles (puestos), por ejemplo existe el rol de Director, Gerente, etc. Lo anterior queda formalizado:

```
Rol(Administrador).
Rol(Analista).
Rol(Director).
....[4]
```

Los roles anteriores están ligados a un departamento y existe una persona dentro de la empresa que ocupa ese puesto, por ejemplo, dentro de la empresa si observamos el organigrama presentado anteriormente, podemos ver que Gilberto Lecona es la persona que ocupa del puesto de Gerente de Finanzas y que Saúl Hernández es el analista de finanzas. Lo anterior al formalizarlo queda:

```
Puesto(Gilberto_lecona,Gerente,Finanzas).
Puesto(Saul_Hernandez,Analista,Finanzas).
Puesto(Esperanza_Garcia,Gerente,Recursos_Humanos).
...
```

En esta empresa también se manejan diferentes tipos de información. En la siguiente tabla, se muestra la información que se maneja, a que departamento pertenece y que clasificación tiene.

Información		
Información	**Departamento**	**Clasificación**
Archivo de passwords	Seguridad	Ultra Secreta
Estados financieros	Finanzas	Ultra Secreta
Lista de precios	Finanzas	No clasificada
Nomina	Recursos Humanos	Secreta

Tabla 3.1 Información manejada por la empresa

La anterior tabla al expresarla en lógica de primer orden, queda de la siguiente forma:

```
Informacion2(Archivo_de_passwords,Seguridad,Confidencial).
Informacion2(Estados_Financieros,Finanzas,Confidencial).
Informacion2(Lista_de_precios,Finanzas,Importante).
Informacion2(Nomina,Recursos_Humanos,Confidencial).
...
```

[4] Estos puntos suspensivos significan que de la misma forma se continuará formalizando todos los registros de la base de datos.

La clasificación sirve para ayudar a proteger a la información de acuerdo a su valor organizacional sin importar del medio en que se almacene o distribuya. En el administrador de políticas de seguridad se pueden introducir los diferentes niveles de clasificación de la información que la empresa u organización decida utilizar, en nuestro caso, se están tomando cuatro niveles que son:

- **Ultra secreta**, en este nivel la información es muy importante para la organización ya que de ser utilizada incorrectamente puede ocasionar graves daños a la organización.
- **Secreta**, la información a este nivel es importante para la información y debe ser protegida.
- **Clasificada**, la información a este nivel es utilizada por los empleados de la empresa.
- **No clasificada**, en el cual la información es de distribución pública a través de los canales autorizados por la empresa.

Estos cuatro niveles quedan formalizados de la siguiente forma:

```
Clasi_inf(Ultra_Secreta).
Clasi_inf(Secreta).
Clasi_inf(Clasificada).
…
```

En la organización la información es almacenada dentro de directorios, los cuales pertenecen a un departamento ya que en ellos se va a guardar la información que genere ese departamento. Además dichos directorios pueden ser compartidos entre varios empleados, ya sea del mismo departamento o de otro. La siguiente tabla muestra los directorios que existen en nuestra empresa y al departamento al que pertenecen.

Directorios de la empresa	
Directorio	**Departamento al que pertenece**
Dirección	Dirección
Dfinanzas	Finanzas
Dseguridad	Seguridad
Dsistemas	Sistemas
DRecursos Humanos	Recursos Humanos

Tabla 3.2 Directorios de la empresa

La anterior tabla queda formalizada de la forma siguiente:

```
Directorio(DDireccion,Direccion).
Directorio(DFinanzas,Finanzas).
Directorio(DSeguridad,Seguridad).
…
```

Como se comentó anteriormente, en estos directorios se almacena la información que cada departamento genera y quiere compartir. La siguiente tabla muestra la información que cada directorio de nuestra empresa ejemplo tiene.

Información almacenada en los directorios	
Directorio	**Departamento al que pertenece**
DFinanzas	Estados financieros y lista de precios.
DSeguridad	Archivo de passwords
DRecursos Humanos	Nomina

Tabla 3.3 Información almacenada en los directorios de la empresa

El administrador de políticas al traducir la tabla anterior utilizando lógica de primer orden, escribe:

```
Estan(Estados_Financieros,DFinanzas).
Estan(Lista_de_precios,DFinanzas).
Estan(Archivo_de_passwords,DSeguridad).
Estan(Nomina,DRecursos_Humanos).
```

Por otro lado es importante escribir en este ambiente los mecanismos de autentificación que los empleados van a utilizar para ingresar al sistema. Los mecanismos de autenticación pueden ser los certificados digitales, las firmas digitales, métodos criptográficos, passwords, reconocimiento de voz o huella, etc. Lo anterior lo formaliza el administrador de políticas de la siguiente forma:

```
Aute(Certificados_Digitales).
Aute(Firmas_Digitales).
Aute(Metodos_Criptograficos).
…
```

Para realizar un análisis del ambiente necesitamos identificar cuando una cláusula es diferente de otra (por ejemplo, que el departamento Dirección es diferente al departamento Sistemas). Lo anterior se conoce como *"Unique names assumption",* este término nos dice que dos objetos con diferente nombre son considerados diferentes.

Utilizando entonces Unique names assumption el administrador de políticas automáticamente expresa que cada elemento del ambiente (departamentos, roles, información, directorios, etc.) es diferente utilizando el símbolo de desigualdad (*!=*). Por ejemplo, el administrador de políticas escribirá los diferentes departamentos de la siguiente forma:

```
Departamento(x) != Departamento(y).
```

Donde *x* representa un departamento diferente de *y,* por ejemplo Departamento(Sistemas) != Departamento(Finanzas).

3.2.2 FORMALIZACIÓN DE E_1.

Como se mencionó en el capítulo 2, el ambiente E_1 está compuesto de fórmulas con cuantificadores universales. A continuación se describe qué tipo de fórmulas universales se escriben en el ambiente, cómo el administrador de políticas realiza esta formalización y por qué es importante considerarlas dentro del ambiente.

Para poder manejar correctamente a Unique names assumption es necesario definir los axiomas de igualdad (reflexividad, simetría y transitividad). El administrador de políticas los escribe utilizando las siguientes fórmulas:

```
%Axiomas de igualdad
all x (x = x). % Reflexividad
all x y (x=y -> y=x). % Simetría
all x y z (x=y & y=z -> x=z). % Transitividad
```

Por otro lado, es importante indicar mediante una cerradura que cada empleado pertenece a un departamento y que por lo tanto, dicho empleado pertenece a la empresa. Para lograr lo anterior, el administrador de políticas escribe la siguiente fórmula partiendo de la relación anteriormente escrita para cada puesto de la empresa, la cual es *Puesto(x,y,z)*. Esta cerradura queda de la siguiente forma:

```
all x y z (Puesto(x,y,z) -> (Empleado(x,z) & Empleado1(x))).
```

Así como se tuvo que crear una cerradura para los empleados, de igual forma, es necesario que el administrador de políticas escriba una cerradura sobre la información. Esta cerradura indica que la información pertenece a un departamento y por lo tanto es información de la

empresa. En este caso partimos de la relación *Informacion2(x,y,z)*, quedando la cerradura de la siguiente forma:

```
all x y z (Informacion2(x,y,z) -> Informacion1(x,y) & Informacion(x)).
```

Dentro del ambiente, el administrador de políticas define los modos de acceso sobre los archivos, esto es importante ya que más adelante serán utilizados para escribir las políticas de acceso de control sobre archivos. Los principales modos de accesos sobre archivos que tenemos son: leer, escribir, ejecutar e imprimir. Sin embargo, tenemos que considerar las posibles combinaciones que se tengan. Se crearon las siguientes abreviaturas para expresar cada uno de los modos de acceso posibles. En la siguiente tabla se presentan todos los diferentes modos de acceso combinados y sus formalizaciones:

Modos de acceso de archivos	
Modo de acceso	**Formalización**
Sólo Lectura (R)	`all x y (Puede(x,R(y)) -> Puede(x,Leer(y)) & - (Puede(x,Escribir(y)) |Puede(x,Imprimir(y)) | Puede(x,Ejecutar(y))))).`
Leer y escribir. (RW)	`all x y (Puede(x,RW(y)) -> Puede(x,Leer(y)) & Puede(x,Escribir(y)) & -(Puede(x,Imprimir(y)) |Puede(x,Ejecutar(y))))).`
Leer y Ejecutar. (RE)	`all x y (Puede(x,RE(y)) -> Puede(x,Leer(y)) & Puede(x,Ejecutar(y)) & -(Puede(x,Imprimir(y)) | Puede(x,Escribir(y))))).`
Leer e Imprimir.(RI)	`all x y (Puede(x,RI(y)) -> Puede(x,Leer(y)) & Puede(x,Imprimir(y)) & -(Puede(x,Ejecutar(y)) | Puede(x,Escribir(y))))).`
Leer, escribir, ejecutar. (RWE)	`all x y (Puede(x,RWE(y)) -> Puede(x,Leer(y)) & Puede(x,Escribir(y)) & Puede(x,Ejecutar(y)) &- Puede(x,Imprimir(y))).`
Leer, Ejecutar e imprimir. (REI)	`all x y (Puede(x,REI(y)) -> Puede(x,Ejecutar(y)) & Puede(x,Imprimir(y)) & Puede(x,Leer(y)) & - Puede(x,Escribir(y)))`
Leer, escribir, imprimir. (RWI)	`all x y (Puede(x,RWI(y)) -> Puede(x,Leer(y)) & Puede(x,Escribir(y)) & Puede(x,Imprimir(y)) & -Puede(x,Ejecutar(y))).`
Leer, escribir, ejecutar e imprimir. (RWEI)	`all x y (Puede(x,RWEI(y)) -> Puede(x,Leer(y)) & Puede(x,Escribir(y)) & Puede(x,Ejecutar(y)) & Puede(x,Imprimir(y))).`

Modos de acceso de archivos	
Modo de acceso	**Formalización**
No puede leer ni escribir ni ejecutar ni imprimir (NoRWEI)	`all x y (Puede(x,NoRWEI(y)) -> -(Puede(x,Leer(y)) | Puede(x,Escribir(y)) | Puede(x,Imprimir(y)) |Puede(x,Ejecutar(y)))).`

Tabla 3.4 Modos de acceso sobre archivos y su formalización en lógica de primer orden

También, nuestro ambiente debe contener los modos de acceso sobre directorios. Estos modos básicos son los siguientes: leer, escribir y compartir. De igual forma que en el caso de los modos de acceso sobre archivos debemos considerar sus posibles combinaciones. En la siguiente tabla se muestran cada una de las abreviaturas para los diferentes modos de acceso sobre directorios combinados y su formalización:

Modos de acceso sobre directorios	
Modo de acceso	**Formalización**
Leer directorio. (Rd)	`all x y (Puede(x,Rd(y)) -> Puede(x,LeerD(y)) & -(Puede(x,EscribirD(y)) | Puede(x,CompartirD(y)))).`
Leer y escribir en el directorio. (RWd)	`all x y (Puede(x,RWd(y)) -> Puede(x,LeerD(y)) & Puede(x,EscribirD(y)) &-Puede(x,CompartirD(y))).`
Leer y compartir el directorio. (RSd)	`all x y (Puede(x,RSd(y)) -> Puede(x,LeerD(y)) & Puede(x,CompartirD(y)) & -Puede(x,EscribirD(y))).`
Leer, escribir y compartir en el directorio. (RWSd)	`all x y (Puede(x,RWSd(y)) -> Puede(x,LeerD(y)) & Puede(x,EscribirD(y)) & Puede(x,CompartirD(y))).`
No puede leer, escribir ni compartir en el directorio. (NoRWSd)	`all x y (Puede(x,NoRWSd(y)) -> -(Puede(x,LeerD(y)) | Puede(x,EscribirD(y)) | Puede(x,CompartirD(y)))).`

Tabla 3.5 Modos de acceso sobre directorios y su formalización en lógica de primer orden

Es importante mencionar que el último modo de acceso de las tablas 3.4 y 3.5, visto desde el punto de vista lógico no tiene sentido, sin embargo es necesario crearlo debido a que al momento de formalizar las políticas de seguridad debemos indicar que existe un modo de acceso el cual no realiza ninguna operación básica (leer, escribir, ejecutar e imprimir para el caso de archivos o leer, escribir y compartir en el caso de directorios). Lo anterior nos sirve de cerradura al momento de escribir las políticas de control de acceso sobre archivos y directorios.

Hasta este punto, se han formalizado todos los elementos que componen el ambiente, es decir todos los elementos básicos que son necesarios para la empresa. Dentro de la interfaz, el administrador podrá generar el ambiente al seleccionar la opción "Generar ambiente" dentro del menú "Formalización". En la siguiente sección se explica cómo se formalizan las políticas de seguridad.

3.3 FORMALIZACIÓN DE LAS POLÍTICAS DE SEGURIDAD.

En el administrador de políticas de seguridad se pueden escribir las políticas de seguridad mediante dos formas: i) a través de tablas de control de acceso y/o ii) insertando la política directamente utilizando la interfaz gráfica. En las siguientes secciones se muestra cómo se formalizan cada una de estas políticas.

3.3.1 POLÍTICAS ESCRITAS A TRAVÉS DE TABLAS DE CONTROL DE ACCESO.

En el escenario juguete que estamos manejando, los permisos sobre la información están definidos en las siguientes listas de acceso:

Estados financieros	
Puesto	**Permisos**
Director	R,-- ,-- ,I
Administrador de seguridad	--,-- ,-- ,--
Gerente de finanzas	R,W,E,I
Analista de finanzas	R,-- ,-- ,--
Gerente de sistemas	--,-- ,-- ,--
Programador de sistemas	--,-- ,-- ,--
Gerente de Recursos H	--,-- ,-- ,--

Archivo de passwords	
Puesto	**Permisos**
Director	--,-- ,-- ,--
Administrador de seguridad	R,W,E,I
Gerente de finanzas	--,-- ,-- ,--
Analista de finanzas	--,-- ,-- ,--
Gerente de sistemas	R,---,--,--
Programador de sistemas	--,-- ,-- ,--
Gerente de Recursos H	--,-- ,-- ,--

Lista de precios	
Puesto	**Permisos**
Director	R,-- ,-- ,I
Administrador de seguridad	--,-- ,-- ,--
Gerente de finanzas	R,W,E,I
Analista de finanzas	R,W,--,--
Gerente de sistemas	--,-- ,-- ,--
Programador de sistemas	--,-- ,-- ,--
Gerente de Recursos H	--,-- ,-- ,--

Nomina	
Puesto	**Permisos**
Director	R,-- ,-- ,I
Administrador de seguridad	--,-- ,-- ,--
Gerente de finanzas	--,-- ,-- ,--
Analista de finanzas	--,-- ,-- ,--
Gerente de sistemas	--,-- ,-- ,--
Programador de sistemas	--,-- ,-- ,--
Gerente de Recursos H	R,W,E,I

Figura 3.2 Listas de control sobre archivos

Las tablas anteriores fueron capturadas en el administrador de políticas de seguridad, por lo que dichas tablas quedaron almacenadas en la tabla "control" dentro de la base de datos. Para explicar cómo se formalizan estas políticas tomaremos como ejemplo la lista de control de acceso de los estados financieros. De acuerdo a lo anterior, se tienen las siguientes políticas:

1. El Analista de Finanzas puede leer los estados financieros.
2. El Director de Dirección puede leer e imprimir los estados financieros.
3. El Gerente de Finanzas puede leer, escribir, ejecutar e imprimir los estados financieros.

El administrador de políticas traduce estas políticas a lógica de primer orden utilizando el lenguaje seleccionado del trabajo de Halpern y Weissman, explicados en el capítulo anterior. Por tal motivo, las políticas son formalizadas de la siguiente forma:

$$\forall_{x} \ldots\ldots x_{m} ((C \rightarrow (\neg)Puede(t,t')) \ ,$$

donde C representa las literales fijas que no pueden contener el predicado *Puede(t,t')*, *t* es un término que representa al sujeto y *t'* un término que representa la acción que realiza.

Siguiendo este formato, las políticas anteriores quedan expresadas automáticamente por el administrador de políticas de la siguiente forma:

```
all x (Puesto(x,Analista,Finanzas) -> Puede(x,R(Estados_Financieros))).
all x (Puesto(x,Director,Direccion) -> Puede(x,RI(Estados_Financieros))).
all x (Puesto(x,Gerente,Finanzas) -> Puede(x,RWEI(Estados_Financieros))).
```

Como se puede observar, para formalizar estas políticas se aplican los modos de acceso descritos en la tabla 3.4.

3.3.2 POLÍTICAS ESCRITAS A TRAVÉS DE LA INTERFAZ GRÁFICA.

Como se comentó anteriormente, en la herramienta se puede escribir una política a través de la interfaz gráfica. Esto se hace siguiendo una secuencia de ventanas en las cuales se le va indicando al administrador que seleccione el sujeto, la acción y el complemento de la política. Por ejemplo, podemos escribir con la interfaz gráfica de la herramienta una política que diga

que "*el gerente de sistemas puede modificar el archivo de passwords"*. Esta política es almacenada en la base de datos para que posteriormente el administrador de políticas la formalice.

La formalización que sigue el administrador de políticas es de acuerdo a los lineamientos del lenguaje $\mathcal{L}_7$ definido el trabajo de Halpern y Weissman [14] por lo que sigue el mismo formato mostrado en la sección anterior. Sin embargo, debido a que en este tipo de políticas existen acciones equivalentes se han creado dos tablas (una a nivel archivo y la otra a nivel directorio) para manejar sinónimos. Por ejemplo, si el administrador de seguridad captura una política y selecciona la acción "modificar", entonces el administrador de políticas al momento de formalizarla buscará en las tablas de sinónimos la acción correspondiente, que en este caso es escribir.

El objetivo que se persigue con este cambio es unificar descripciones y evitar redundancias en las acciones de las políticas de seguridad. A continuación se muestran en la siguiente tabla los diferentes verbos que se pueden utilizar para escribir una política de control de acceso sobre archivos y su sinónimo correspondiente:

Acciones a nivel archivo	
Verbo de la política	**Sustitución en la formalización**
Escribir, agregar, insertar, modificar o cambiar	Escribir
Salvar, almacenar, guardar, respaldar o realizar backups	Escribir
Leer	Leer
Ejecutar o cargar	Ejecutar
Imprimir	Imprimir

Tabla 3.6 Tabla de acciones a nivel archivo

La siguiente tabla nos muestra los diferentes verbos que se utilizan para escribir políticas a nivel directorio:

Acciones a nivel directorio	
Verbo de la política	**Sustitución en la formalización**
Crear, Generar, Borrar, Eliminar o Quitar	EscribirD
Copiar, Duplicar, Cortar, Mover, Renombrar o Cambiar de nombre	EscribirD
Leer Directorio, Navegar	LeerD
Compartir	CompartirD

Tabla 3.7 Tabla de acciones a nivel directorio

Utilizando las tablas anteriores, la herramienta formaliza la política "*el gerente de sistemas puede modificar el archivo de passwords"* de la siguiente forma:

```
all x (Puesto(x,Gerente,Sistemas) -> -
Puede(x,Escribir(Archivo_de_passwords))).
```

Una vez expresadas estas políticas, el administrador de políticas debe crear una cerradura. Esta cerradura nos sirve para indicar que todo lo que no está expresado en dicha política está prohibido. Es decir, se necesita poner una cerradura, la cual indique que todos los puestos que no estén especificados en las políticas anteriores no tendrán ningún tipo de permisos.

Como se mencionó anteriormente, el administrador de seguridad se está basando en el lenguaje $\mathcal{L}_7$ descrito en el trabajo de Halpern y Weissman. Sin embargo, la formalización de esta cerradura no cumple con la condición 2 de este lenguaje. Recordando, esta condición nos dice que n+o podemos tener desigualdades en los antecedentes de las políticas.

En la siguiente sección se explica cómo la herramienta crea esta cerradura.

3.4 CERRADURA DE LAS POLÍTICAS DE SEGURIDAD.

Para explicar de una mejor forma cómo el administrador de políticas tiene que calcular la cerradura, tomaremos como ejemplo las políticas que se escribieron en la sección 3.3 sobre los estados financieros (El analista de finanzas puede leer los estados financieros, el director de dirección puede leer e imprimir los estados financieros y el gerente de finanzas puede leer, escribir, ejecutar e imprimir los estados financieros).

Entonces tomando estas políticas, la herramienta calcularía una cerradura que nos indicará que nadie que no sea el analista de finanzas o el gerente de finanzas o el director de dirección podrá tener algún tipo de permiso sobre este archivo. Dicha cerradura, queda expresada de la siguiente forma:

```
all x y z (Puesto(x,y,z) & z != Finanzas & z != Direccion
 -> Puede(x,NoRWEI(Estados_Financieros))).
```

Para calcular la anterior cerradura, el administrador de políticas utiliza un algoritmo computacional, el cual es explicado en la siguiente sección.

3.4.1 ALGORITMO PARA CALCULAR LA CERRADURA.

El algoritmo que utiliza el administrador de políticas para calcular la cerradura consiste en tres módulos principales. Los dos primeros módulos trabajan de forma independiente y su objetivo es extraer información de la base de datos. Esta información va a ser utilizada por el último módulo, que es en sí en donde se calcula la cerradura.

Debido a que las políticas de seguridad son escritas en base al puesto o rol que ocupa cierto sujeto de la organización, este primer módulo tiene el objetivo de que el administrador de políticas sepa cuantos puestos hay por cada departamento. Por lo tanto, el primer módulo de este algoritmo, extrae de la base de datos el número de puestos por departamento.

Estos datos (departamento y número de puestos) son almacenados en una matriz. Esta matriz consiste en dos columnas. En la primera guarda el nombre del departamento y en la segunda guarda la cantidad de puestos que existen en ese departamento. Un ejemplo de esta matriz es el siguiente:

Departamento	**No de Puestos**
Dirección	1
Finanzas	2
Sistemas	2

Figura 3.3 Matriz de departamentos y su número de puestos

El pseudo-código de este módulo es el siguiente:

```
String querye1 = "Select * from Departamento order by
                  Cve_Departamento";
          clave = resultSet.getInt("Cve_Depto");

int [] totaldepto;    totaldepto = new int[clave.size()];

           for(int i = 0; i< clave.size();i++){

String querye2 = "Select count(Nombre_Sujeto) as Cantidad from sujeto
      where Cve_Departamento =" + clave.elementAt(i);

          cantidad = resultSet.getInt("Cantidad");

                totaldepto[i] = cantidad;

                          }
```

Figura 3.3 pseudo-código del primer módulo

El siguiente módulo busca saber cuántos empleados por cada departamento de la empresa están involucrados en cada tipo de política. Para lograr esto, primeramente, extrae las políticas de control de acceso sobre la información de la base de datos y las ordena por el tipo de información (archivo de passwords, estados financieros, etc.).

Una vez, realizado lo anterior, entonces se crea una matriz en la cual se almacenan el número de empleados (puestos) por departamento estén involucrados en las políticas de control de acceso sobre este archivo en particular.

Por ejemplo si tomamos las políticas escritas sobre los estados financieros de esta empresa, la matriz de este modulo quedaría:

Departamento	**No de Puestos**	**Puestos**
Dirección	1	Director
Finanzas	2	Gerente y analista

Figura 3.4 Matriz de puestos involucrados por política de un documento en especial

El pseudo-código de este módulo es el siguiente:

```
String query3 = "Select * from PoliticasC  order by Cve_Info";
info = resultSet.getInt("Cve_Info");

int [] Polidepto;    Polidepto = new int[clave.size()];

for (int j = 0; j < politica.size(); j++) {
depar =  depar - 1

Politotal[depar]  = Politotal [depar]  + 1;

}
```

Figura 3.5 pseudo-código del segundo módulo

El último módulo compara el total de empleados (puestos) del departamento involucrado en el conjunto de políticas contra el número de empleados (puestos) que tienen permisos sobre el archivo. El resultado es la cerradura.

Como se tiene un conjunto de políticas por tipo de archivo, cuando este módulo realiza las comparaciones, la cerradura resultante puede caer en cualquiera de estos casos:

1.- Todos los empleados de un departamento en particular intervienen en las políticas. Donde la cerradura queda *z != Departamento.*

2.- No todos los empleados de un departamento en particular intervienen en las políticas. En este caso la cerradura es *y != puesto | z != Departamento*.

3.- La combinación de ambas. Es decir, cuando se tiene que en el conjunto de políticas de un archivo en especial se involucra a todo un departamento y además a una persona en especial de otro departamento.

El pseudo-código de este módulo es el siguiente:

```
for(int n = 0; n<politica.size();n++){

    if(totaldepto[n] = Politotal [n])
cerradura = " & z != " +  Departamento.elementAt(n));

                    else
 cerradura = " & ( y != " + Puesto(n)+ " |  z != " +
          Departamento.elementAt(n) + ")");


cerraduraF =  "all x y z (Puesto(x,y,z) " + cerradura  ->
          Puede(x,NoRWEI(" + informacion))."
                    }
```

Figura 3.6 pseudo-código del tercer módulo

Para entender mejor este algoritmo, analicemos la cerradura que calcularía la herramienta para el conjunto de políticas escritas para los estados financieros. En este caso se tienen 3 políticas en donde interviene el director de dirección, el gerente de finanzas y el analista de finanzas. En la base de datos, el departamento de dirección tiene sólo un empleado (el director) y el departamento de finanzas tiene dos empleados (Gerente y analista) por lo que se puede observar que todos los empleados de cada uno de estos departamentos están involucrados en este conjunto de políticas. La cerradura calculada corresponde al caso 1, quedando expresada así:

```
all x y z (Puesto(x,y,z) & z != Finanzas & z != Direccion
 -> Puede(x,NoRWEI(Estados_Financieros))).
```

Si eliminamos la política que involucra al director de dirección y la política que involucra al analista de finanzas de tal forma que el conjunto de políticas sobre el control de acceso de los estados financieros sólo involucra al gerente de finanzas, entonces la cerradura correspondería al caso 2, quedando expresada de la siguiente forma:

```
all x y z (Puesto(x,y,z) & y != Gerente & z != Finanzas
 -> Puede(x,NoRWEI(Estados_Financieros))).
```

Para que la cerradura corresponda al caso 3, tendríamos que dejar las políticas referentes al director de dirección y al gerente de finazas. Entonces, la cerradura calculada tiene que expresar que cualquiera que no sea del departamento de dirección o el gerente de finanzas no puede realizar ninguna acción sobre los estados financieros.

```
all x y z (Puesto(x,y,z) & z != Direccion & y != Gerente & z != Finanzas
 -> Puede(x,NoRWEI(Estados_Financieros))).
```

Una vez mostrado el algoritmo que el administrador de políticas utiliza para calcular la cerradura, se mostrará en la siguiente sección su complejidad computacional.

3.4.2 COMPLEJIDAD COMPUTACIONAL DEL ALGORITMO.

La complejidad computacional [19] que tiene el algoritmo que genera la cerradura se calcula de la siguiente forma:

Complejidad Computacional		
Instrucción	**Costo**	**Número de veces**
N = 0	c1	1
N<política.size()	c2n	N
N++	c3n	N
Instrucciones internas	c4n	N

Tabla 3.8 Tabla de complejidad computacional

Para una mayor simplicidad, asumiremos que todas las instrucciones tienen un costo de 1. Entonces, sumando las operaciones tenemos:

$$T(n) = c1 + c2n + c3n + c4n = c1 + n(c2 + c3 + c4) = 1 + n(3).$$

Como se puede observar, la complejidad computacional entonces queda $T(n) = 3n + 1$.

3.5 FORMALIZACIÓN DEL MODELO BELL-LA PADULA.

El objetivo de implementar el modelo de Bell-La Padula es demostrar como las políticas introducidas en nuestro administrador de políticas de seguridad pueden seguir fielmente las reglas que rigen este modelo.

De acuerdo a lo anterior, en la siguiente sección sólo se explican las cláusulas y fórmulas lógicas que son necesarias escribir en el ambiente para que las políticas estén regidas por las dos reglas principales (simple seguridad y la propiedad * (estrella)) de este modelo.

3.5.1 FORMALIZACIÓN DEL AMBIENTE DEL MODELO BELL-LA PADULA.

Para cumplir las reglas de este modelo, es importante especificar que cada clasificación de la información tiene diferentes niveles. Lo anterior significa que el orden de importancia va de ultra secreta a no clasificada. Para lograr lo anterior, utilizamos la técnica de Unique names assumption para indicar que cada elemento es diferente entre sí y entonces usamos operadores de igualdad (igualdad =, menor que <, mayor que >) para indicar los diferentes niveles.

Lo anterior lo formaliza la herramienta de forma automática, quedando expresado de la siguiente forma:

```
Ultra_Secreta > Secreta.
Ultra_Secreta > Clasificada.
Ultra_Secreta > No_Clasificada.
…
No_Clasificada < Clasificada.
No_Clasificada < Secreta.
No_Clasificada < Ultra_Secreta.
...
```

Debido a que estamos formalizando un ambiente militar es importante indicar mediante una cerradura que la persona "*x*" que ocupa un puesto "*y* "es un militar. Esto es con el objetivo de poder realizar más adelante las validaciones con Otter/Mace.

```
all x y (Puesto(x,y) -> (Militar(x))).
```

Una de las características principales de este modelo es que cada sujeto tiene un nivel de autorización asociado a cada rango militar. Los niveles de autorización que cada sujeto puede tener son ultra secreta, secreta, clasificada y no clasificada.

Para explicar mejor lo anterior, supongamos que por ejemplo, en la herramienta se ha capturado que el Teniente general y el Coronel tienen un nivel de autorización de ultra secreta

y que el Capitán y el Teniente primero tienen un nivel de autorización secreta. Entonces el administrador de políticas de seguridad formaliza lo anterior de la siguiente forma:

```
all x (Puesto(x,Teniente_General) -> Permisos(x,Ultra_Secreta)).
all x (Puesto(x,Coronel) -> Permisos(x,Ultra_Secreta)).
all x (Puesto(x,Capitan) -> Permisos(x,Secreta)).
all x (Puesto(x,Teniente_primero) -> Permisos(x,Secreta)).
```

El siguiente paso es formalizar las propiedades que rigen la lectura y escritura de la información dentro de este modelo. La primera es la propiedad de simple seguridad la cual dice que *"El nivel del sujeto debe ser, por lo menos, igual o menor que la clasificación de la información para que se le permita leer la información."* Lo anterior lo formaliza la herramienta así:

```
all x y w z (Permisos(x,w) & Informacion2(y,z) & (w > z | w = z ) ->
Puede(x,Leer(y))).

all x y w z (Permisos(x,w) & Informacion2(y,z) & (w < z ) -> -
Puede(x,Leer(y))).
```

La segunda es la propiedad * (estrella) que dice nos *"El nivel de la información debe ser, igual o mayor que la clasificación de la misma para que se le permita al sujeto escribir sobre la información"*. La formalización de esta propiedad queda:

```
all x y w z (Permisos(x,w) & Informacion2(y,z) & (w < z | w = z ) ->
Puede(x,Escribir(y))).

all x y w z (Permisos(x,w) & Informacion2(y,z) & (w > z ) -> -
Puede(x,Escribir(y))).
```

Estas dos reglas son muy importantes debido a que definen el comportamiento del modelo. Por lo tanto, las políticas que se escriban deberán estar en concordancia con estas dos propiedades para poder ser implementadas en la organización.

Es importante mencionar que en este modelo sólo existen dos modos de acceso que son leer y escribir.

3.6 CONCLUSIONES.

Las políticas de seguridad que pueden ser introducidas en la herramienta deben ser políticas referentes al control acceso y flujo de la información. En este capítulo se mostró cómo el

administrador de políticas traduce estas políticas a lógica de primer orden utilizando los parámetros del lenguaje definido del trabajo de Halpern y Weissman.

Una vez realizada esta formalización, se puede realizar un razonamiento sobre las políticas. En el siguiente capítulo se muestra cómo se realizan las validaciones utilizando Otter/Mace.

4 VALIDACIÓN DE LAS POLÍTICAS DE SEGURIDAD UTILIZANDO OTTER/MACE.

4.1 INTRODUCCIÓN.

Gracias a que el administrador de políticas utiliza el lenguaje de especificación explicado anteriormente para la formalización de las políticas, ahora podemos realizar un análisis más extenso de las mismas. El objetivo del presente capítulo es explicar cómo se realizan estas validaciones utilizando Otter/Mace.

Para una mejor explicación, dividiremos las validaciones de políticas en dos partes: validaciones del ambiente y validaciones de las políticas. En lo que respecta a las validaciones del ambiente, se explicará cómo el administrador de políticas valida el ambiente mediante la realización de preguntas interesantes utilizando Otter y/o construyendo un modelo con Mace.

Por otro lado, en lo que se refiere a las políticas, el administrador de políticas puede realizar las siguientes validaciones utilizando Otter/ Mace: i) verificar qué sujeto(s) tiene(n) ciertos permisos sobre un objeto en especial, ii) verificar si un sujeto puede o no hacer una acción *x*, iii) verificar si las políticas escritas son consistentes y iv) verificar si las políticas escritas satisfacen el modelo de Bell – La Padula.

4.2 VALIDACIONES DEL AMBIENTE.

Uno de los objetivos del administrador de políticas es verificar que las políticas de seguridad estén en concordancia con su ambiente. Por lo tanto, el realizar pruebas sobre el ambiente nos

permite verificar que éste no contenga ningún elemento que genere una contradicción en el contexto.

Para que la herramienta pueda verificar lo anterior, utiliza la interfaz gráfica, que a través de ventanas dirigidas formular la validación (pregunta). Esta pregunta es formalizada por la herramienta y genera un archivo de entrada.

Este archivo de entrada que Otter/Mace utiliza(n) está compuesto de E_0 (literales fijas) y E_1 (fórmulas con cuantificadores universales). En las siguientes secciones se muestra cómo se utiliza Otter/Mace para lograr este objetivo.

4.2.1 VALIDACIÓN DEL AMBIENTE UTILIZANDO OTTER.

Mediante la herramienta se pueden probar dos características importantes del ambiente que son a) los empleados y b) la información del sistema. Lo anterior, lo realiza la herramienta mediante la formulación de preguntas que Otter deberá contestar.

Para explicar mejor cómo se hacen las validaciones sobre el ambiente usando Otter, usaremos el siguiente ejemplo. Supongamos que existe un error en la captura de los empleados de tal forma que se introduce en la herramienta a un mismo empleado en dos departamentos. Como sabemos, esto no es posible ya que dentro de una organización no podemos tener físicamente a dos empleados laborando en dos departamentos diferentes. Si deseamos verificar lo anterior, se deberá formular mediante la interfaz gráfica esta una pregunta para Otter. Para realizar esto, debemos entrar al submenú “Probar ambiente” en el menú “Formalizar” y entonces la interfaz nos mostrará una pantalla en la cual iremos construyendo nuestra pregunta.

Una vez construida nuestra validación, la herramienta escribirá la siguiente fórmula lógica en el archivo entrada.

```
all x y z (Empleado(x,y) & Empleado(x,z) -> y=z).
```

Inmediatamente, el administrador de políticas mandará llamar a Otter para que nos dé un respuesta. Si Otter se detiene y regresa como resultado la cláusula vacía se deberá revisar este resultado para detectar que el empleado está en dos departamentos a la vez y corregir este error. En caso de que Otter se detenga y nos dé como resultado la lista de soporte (sos) vacía

entonces el administrador del sistema podrá estar seguro de que no existe un empleado registrado en dos departamentos.

Otro tipo de preguntas que podemos realizar sobre los empleados son por ejemplo, ¿Quiénes son los empleados? ¿Cuáles son los departamentos de la empresa? ¿Qué roles existen en la empresa? ¿Quiénes trabajan en el departamento *x*? ¿En cuál departamento trabaja cierto empleado? Y por último ¿Quién/Quienes tienen cierto puesto?

En lo que se refiere a la información, se puede verificar si la información pertenece a un departamento o verificar que toda la información introducida tenga una clasificación. También se pueden cuestionar qué tipos y/o clasificaciones de la información se tienen en el sistema, si *x* información pertenece a algún departamento o tiene cierta clasificación.

Un usuario puede conocer exactamente cuál es el espectro de preguntas que puede formular al entrar a la ventana de "Probar empleados" o "Probar información" en el menú "Formalización" en la interfaz gráfica.

Otra forma en que la herramienta verifica que los elementos que componen el ambiente sean consistentes es ejecutando a Mace. Por tal motivo, en la siguiente sección se explica cómo se realiza esta validación con Mace.

4.2.2 VALIDACIÓN DEL AMBIENTE UTILIZANDO MACE.

Cuando la herramienta ejecuta a Mace, éste construye un modelo si no existen contradicciones en la teoría evaluada, en este caso, si el ambiente es consistente. Si el ambiente tiene alguna contradicción, Mace no podrá construir un modelo por lo que mandará el siguiente mensaje:

The search is complete. No Models were found

Para ejemplificar lo anterior, se verificó la consistencia del ambiente del ejemplo juguete mostrado en el capítulo anterior. En este caso, la herramienta teniendo al ambiente formalizado manda ejecutar a Mace. Mace al no encontrar ninguna contradicción en el ambiente construye el siguiente modelo:

```
====================== Model #1 at 0.02 seconds:

 Direccion: 0 Finanzas: 1 Seguridad: 2 Sistemas: 3 Recursos_Humanos: 4

 Departamento:
    0 1 2 3 4 5 6
  -----------------
    T T T T T F F


 Administrador: 0 Analista: 1 Director: 2 Gerente: 3 Programador: 4


 Rol :
    0 1 2 3 4 5 6
  -----------------
    T T T T T F F


Anna_Alvarez: 0 Carlos_Leon: 1 Ericka_Hernandez: 2 Esperanza_Garcia: 3
Gilberto_lecona: 4 Karen_Garcia: 5 Saul_Hernandez: 6

 Empleado (x,y) : (Nombre,Departamento)
      | 0 1 2 3 4 5 6
    --+--------------
    0 | F F F T F F F   (Anna Alvarez, Sistemas)
    1 | F F F T F F F   (Carlos Leon, Sistemas)
    2 | T F F F F F F   (Ericka Hernandez, Dirección)
    3 | F F F F T F F   (Esperanza Garcia, Recursos Humanos)
    4 | F T F F F F F   (Gilberto Lecona, Finanzas)
    5 | F F T F F F F   (Karen Garcia, Seguridad)
    6 | F T F F F F F   (Saúl Hernandez, Finanzas)

 Empleado1 :
        0 1 2 3 4 5 6
    -----------------
        T T T T T T T

Dimension 3 table for Puesto not printed

 Confidencial: 0 General: 1 Importante: 2

 Clasi_inf :
        0 1 2 3 4 5 6
    -----------------
        T T T F F F F


 Archivo_de_passwords: 0 Estados_Financieros: 1 Lista_de_precios: 2 Nomina:
3


 Informacion :
        0 1 2 3 4 5 6
    -----------------
        T T T T F F F

Informacion1(x,y) : (Información, Departamento)
      | 0 1 2 3 4 5 6
    --+--------------
    0 | F F T F F F F   (Archivo de passwords, seguridad)
    1 | F T F F F F F   (Estados financieros, finanzas)
    2 | F T F F F F F   (Lista de precios, finanzas)
```

```
   3 | F F F F T F F    (Nomina, Recursos humanos)
   4 | F F F F F F F
   5 | F F F F F F F
   6 | F F F F F F F

 Clas_inf1 (x,y) (información, clasificación)

     | 0 1 2 3 4 5 6
   --+--------------
   0 | T F F F F F F   (Archivo de passwords, Confidencial)
   1 | T F F F F F F   (Estados financieros, Confidencial)
   2 | F F T F F F F   (Lista de precios, Importante)
   3 | T F F F F F F   (Nomina, Confidencial)
   4 | F F F F F F F
   5 | F F F F F F F
   6 | F F F F F F F

Dimension 3 table for Informacion2 not printed


DDireccion: 0  DFinanzas: 1  DSeguridad: 2 DSistemas: 3 DRecursos_Humanos:
4


 Directorio (Directorio, Departamento)
     | 0 1 2 3 4 5 6
   --+--------------
   0 | T F F F F F F  DDirección,Dirección
   1 | F T F F F F F  DFinanzas, Finanza
   2 | F F T F F F F  DSeguridad, Seguridad
   3 | F F F T F F F  DSistemas, Sistemas
   4 | F F F F T F F  DRecursos_Humanos, Recursos Humanos
   5 | F F F F F F F
   6 | F F F F F F F


 Estan(Archivo, Directorio)

     | 0 1 2 3 4 5 6
   --+--------------
   0 | F F T F F F F  Archivos de passwords, Seguridad
   1 | F T F F F F F   Estados financieros, Finanzas
   2 | F T F F F F F   Lista de precios, Finanzas
   3 | F F F F T F F   Nomina, Recursos Humanos
   4 | F F F F F F F
   5 | F F F F F F F
   6 | F F F F F F F

Certificados_Digitales: 0 Firmas_Digitales: 1 Metodos_Criptograficos: 2
Passwords: 3
Reconocimiento_de_huella: 4 Reconocimiento_de_voz: 5


 Aute :
       0 1 2 3 4 5 6
   ----------------
       T T T T T T F
end_of_model
```

Analizando el modelo anterior, podemos observar que Mace asigna un valor a cada uno de los elementos del ambiente. Dicho valor va de *0* a *n-1* siendo *n* el número de elementos. Por

ejemplo, en el ambiente juguete se tienen 5 departamentos, entonces Mace le asigna el valor de cero al primer departamento, el valor de uno al segundo departamento y así sucesivamente.

Posteriormente a esto, Mace construye un modelo de las relaciones lógicas existentes en las fórmulas con cuantificadores universales. Por ejemplo, como vimos en el capítulo anterior, dentro del ambiente la herramienta escribe una cerradura para indicar que los empleados de un determinado departamento son empleados de la empresa. La cerradura que se utiliza es: *all x y z (Puesto(x,y,z) -> (Empleado(x,z) & Empleado1(x)))* en la cual la relación Empleado(x,y) nos indica que *x* es empleado del departamento *y*.

Mace construirá una tabla de verdad para representar esta relación, en la cual, los valores de los empleados serán los valores verticales de la tabla y los valores de los departamentos estarán localizados en forma horizontal. La intersección de ambos valores será verdadera si el empleado existe en dicho departamento (por ejemplo, Esperanza García labora en el departamento de recursos humanos), en caso contrario será falsa.

4.3 VALIDACIONES DE LAS POLÍTICAS CON OTTER/MACE.

Como se comentó al inicio de este capítulo, el administrador de políticas realiza cuatro validaciones de las políticas utilizando Otter/ Mace. Todas estas validaciones parten de que la herramienta genera un archivo de entrada, el cual contiene al ambiente (E_0 y E_1) y al conjunto de políticas. Ambos elementos están formalizados siguiendo los lineamientos de [14]. En las siguientes secciones se describe cómo el administrador de políticas puede realizar cada una de estas validaciones.

4.3.1 VERIFICAR QUÉ SUJETO(S) TIENE(N) CIERTOS PERMISOS SOBRE UN OBJETO EN ESPECIAL.

Esta validación nos permite saber en forma rápida los permisos que tiene un sujeto de acuerdo a las listas de control de acceso introducidas en el administrador de políticas. Es decir, para determinar el perfil de un individuo *S*, la herramienta le da a Otter la conjetura $E_0 \wedge E_1 \wedge P_1 \wedge$

$\ldots \wedge P_n \wedge (\neg permitted(S, X) \vee \$ANS(X))$. Otter evaluará lo anterior y nos dará como respuesta todas las pruebas posibles que encuentre.

Para que Otter puede buscar todas las respuestas, se introduce el predicado *$ANS*. Utilizando este predicado, Otter considera el conjunto de las cláusulas correspondientes de $S \cup \{ \forall x_n [F(x_1 \ldots\ldots x_n)[F(x_1 \ldots\ldots x_n) \rightarrow \$ANS(x_1 \ldots\ldots x_n)]\}$ donde S es un conjunto de fórmulas y $x_1 \ldots x_n$ son variables libres de la fórmula a evaluar. Teniendo lo anterior, Otter aplicará el procedimiento de resolución hasta encontrar una cláusula cuyo único literal contenga el predicado *$ANS*. Los términos que aparecen en dicho literal forman una respuesta a la cuestión planteada.

Para ejemplificar lo anterior y usando el ejemplo juguete introducido en el capítulo anterior, podemos preguntarle a Otter quién tiene permisos de lectura sobre los estados financieros. Lo anterior, quedaría de la siguiente forma:

```
Puede(x,Leer(Estados_financieros)) | $ANS(x).
```

Si existe alguna política en la cual se dice que cierta persona tiene permisos sobre los estados financieros, entonces Otter nos responderá dándonos el ó los nombres de los sujetos que pueden leer este archivo. En caso contrario, nos dirá que la lista de soporte (sos) está vacía.

4.3.2 VERIFICAR SI UN SUJETO PUEDE REALIZAR UNA ACCIÓN X.

Otra validación que podemos hacer utilizando Otter/Mace es verificar si cierto sujeto puede realizar una acción *x*. Para realizar esta validación la herramienta le da a Otter/Mace un archivo de entrada el cual contiene $E_0 \wedge E_1 \wedge P_1 \wedge \ldots \wedge P_n \wedge \neg puede(S, A)$, siendo S un individuo y A la acción que realiza dicho individuo sobre un objeto en especial. Si dicha conjetura es un teorema, Otter deducirá la cláusula vacía. En caso contrario, Mace construirá un contraejemplo.

Además, de los resultados anteriores, debemos considerar que existe la posibilidad de que Otter no cuente con lo elementos necesarios para darnos una respuesta, por lo que en el mejor de los casos nos responderá con la lista de soporte (sos) vacía.

De acuerdo a lo anterior, si deseamos validar si un sujeto tiene los permisos para realizar una acción *x* la herramienta negará el teorema que queremos validar, es decir, la herramienta escribe las cláusulas para validar este punto de la siguiente forma:

- ((-) Puede (x,Accion(y))).

Por ejemplo, si se desea validar si Carlos León (programador de sistemas) tiene permisos para leer los estados financieros de la empresa, entonces, el administrador de políticas escribirá la siguiente cláusula:

- (Puede (Carlos_Leon,Leer(Estados_Financieros))).

Una vez escrita la cláusula anterior, la herramienta mandará ejecutar a Otter/Mace para evaluar esta cláusula. Si esta cláusula no se contradice con las políticas de control de acceso que ya se escribieron sobre los estados financieros en el administrador de políticas, Otter se detiene y nos dice que su lista de soporte (sos) esta vacía. En caso contrario, Otter se detiene y nos indica que encontró la cláusula vacía.

Si Otter no nos da ninguna de estas dos repuestas en un tiempo razonable, podemos utilizar a Mace para validar lo anterior. Cómo ya se mencionó anteriormente Mace construirá un modelo, siempre y cuando la validación que estemos haciendo no se contradiga con listas de control de acceso introducida en el administrador de políticas.

4.4.3 VERIFICAR CONSISTENCIA DE POLÍTICAS.

Otro aspecto importante que podemos verificar con Otter/Mace es la consistencia de un conjunto de políticas. La herramienta le dará a Otter y a Mace como entrada una conjetura $E_0 \wedge E_1 \wedge P_1 \wedge \ldots \wedge P_n$. Para evaluar dicha conjetura el administrador de políticas mandará llamar a Otter o Mace. En el caso de Otter obtenemos los siguientes resultados:

a) Se detiene en el momento en que encuentra la cláusula vacía, resultado que nos lleva a la conclusión de que hay inconsistencias en el conjunto de políticas de seguridad evaluadas por lo que éstas deben ser modificadas o eliminadas según se requiera antes de implementarlas en la organización.

b) Se detiene y nos indica que la lista de soporte (sos) esta vacía, es decir, Otter ya no encontró ninguna forma de inferir nuevas cláusulas. Por lo tanto, este resultado nos indica que el conjunto de políticas evaluado es consistente y pueden ser implantadas en la organización.

c) No se detiene en un tiempo razonable debido a que no tiene los elementos necesarios para dar una respuesta.

El tercer resultado (c) se vuelve un grave problema para el administrador de seguridad ya que no se sabe si las políticas analizadas van a cumplir con su objetivo de proteger a la información, por lo tanto, tampoco sabemos si se pueden implementar en la organización o no.

Para subsanar este problema, podemos utilizar a Mace (constructor de modelos). Mace al evaluar a las políticas nos arrojará dos respuestas:

a) Construye un modelo si el conjunto de políticas es consistente. Este resultado, nos dice que nuestras políticas pueden ser implementadas en la organización.

b) No construye un modelo, al encontrar contradicciones en las políticas.

Para entender mejor este punto, a continuación se muestra como Mace construye un modelo cuando las políticas son consistentes. Para ilustrar lo anterior, se escribieron las siguientes políticas en el administrador de políticas realizado:

1.- El Gerente de sistemas puede leer el archivo de passwords.
2.- El director de dirección puede leer e imprimir los estados financieros.
3.- El Gerente de recursos humanos puede leer, escribir e imprimir la nomina
4.- El Gerente de finanzas puede leer, escribir e imprimir estados financieros.
5.- El administrador de seguridad puede leer, escribir e imprimir el archivo de passwords.
6.- El analista de finanzas puede leer los estados financieros y puede leer y escribir la lista de precios.

Una vez capturadas estas políticas, el administrador de políticas tiene que generar un archivo de entrada para Mace, en el cual formaliza las políticas anteriores y además agrega una lista

de cláusulas usables utilizando la técnica de Unique names assumption para cada modo de acceso que se utilizan en la escritura de las políticas. A continuación se muestra un ejemplo de cómo queda dicha sección en el archivo de entrada:

```
list(usable).

RWEI(y) != R(y).
RWEI(y) != RI(y).
RWEI(y) != RW(y).
RWEI(y) != RWI(y).
R(y) != RI(y).
….
End_of_list.
```

Al no tener contradicciones en las políticas anteriores, Mace construye el siguiente modelo:

```
====================== Model #1 at 0.02 seconds:


 RWI :
        0 1 2 3 4 5 6
    ----------------
        0 0 0 0 0 0 0
R :
        0 1 2 3 4 5 6
    ----------------
        1 1 1 1 1 1 1

RI :
        0 1 2 3 4 5 6
    ----------------
        2 2 2 2 2 2 2

RW :
        0 1 2 3 4 5 6
    ----------------
        3 3 3 3 3 3 3

 RWEI :
        0 1 2 3 4 5 6
    ----------------
        4 4 4 4 4 4 4


Puede (x,y)  (nombre,permisos)

      | 0 1 2 3 4 5 6
    --+--------------
    0 | F T F F F F F   Política 1
    1 | F F F F F F F
    2 | F F T F F F F   Política 2
    3 | T F F F F F F  Política 3
    4 | T F F F F F F   Política 4
    5 | T F F F F F F   Política 5
    6 | F T F T F F F   Política 6

 end_of_model
```

Como se explicó anteriormente, Mace para poder escribir un modelo le asigna un valor a cada elemento que va de *0* a *n-1* siendo *n* el número de elementos. Por ejemplo, en el caso de las acciones, RWI (leer, escribir e imprimir un archivo) se les asigna el valor de 0, la acción R (leer) se le asigna el valor de 1, y así sucesivamente.

Una vez que Mace asigna estos valores, construye una definición por enumeración explícita de la relación "*Puede(x,y)"* que es en sí la política. Si la intersección de "*x"* y "*y"* es 1, entonces *(x,y)* es parte de *R*; en caso contrario *(x, y)* no es parte de *R*.

Para ilustrar mejor lo anterior, tomemos la primera política que dice "El Gerente de sistemas puede leer el archivo de passwords". En la herramienta la persona que ocupa el puesto de gerente de sistemas es Anna Álvarez, por lo que al formalizar esta política, tenemos:

Puede (Anna_Alvarez, R(Archivo_de_passwords)

Es decir, dicha política es de la forma *Puede(x,R(y))*. De acuerdo con los valores anteriormente asignados por Mace, el primer componente de esta relación (Anna Álvarez) vale 0 y el segundo componente (R(y)) vale 1. Por lo tanto, en la tabla, la intercepción de 0,1 corresponde a esta política. Y si observamos el modelo que Mace construyó veremos que a dicha intercepción le asignó un valor de verdadero, lo que nos indica que la política es consistente.

Es importante mencionar que para que Mace construya un modelo de las políticas no es necesario que el archivo de entrada contenga la cerradura que utiliza Otter (descrita en el capítulo anterior) para indicarle que nadie más que las personas involucradas en la política tienen acceso a este archivo en particular.

4.3.4 VERIFICAR SI LAS POLÍTICAS SIGUEN EL MODELO DE BELL- LA PADULA.

Por último, al implementar el modelo de Bell-La Padula en la herramienta, podemos verificar con Otter/Mace si las políticas escritas satisfacen las dos principales propiedades de este modelo. Recordemos, que la primera propiedad de Bell – La Padula nos dice que "El nivel del

sujeto debe ser, por lo menos, igual o menor que la clasificación del objeto para que se le permita leer al objeto". Esta propiedad es expresada en lógica de primer orden de la siguiente forma $E_0 \wedge E_1 \wedge P_1 \wedge \ldots \wedge P_n \wedge permisos(S) \leq nivel(O) \wedge \neg Puede(S, A)$.

Por otro lado, la segunda propiedad de este modelo nos dice que "El nivel del objeto debe ser, igual o mayor que la clasificación del objeto para que se le permita al sujeto escribir sobre el objeto". Dicha propiedad es expresada en lógica de primer orden de la siguiente forma $E_0 \wedge E_1 \wedge P_1 \wedge \ldots \wedge P_n \wedge Puede(S, A) \wedge \neg(permisos(S) \leq nivel(O))$. En ambas propiedades S es el sujeto y A es la acción que recae sobre un objeto *(O)* en especial.

La validación del modelo Bell-La Padula se realiza de igual forma que la validación de consistencia en las políticas de seguridad mostrada en la sección anterior. Esto es, la herramienta formaliza las políticas siguiendo los parámetros del lenguaje $\mathcal{L}_7$ definido en el trabajo de Halpern y Weissman y posteriormente manda llamar a Otter/Mace para realizar las validaciones.

Los resultados que Otter regresará a la herramienta son los siguientes: i) La cláusula vacía, lo que nos indica, que existe una política que no está siguiendo fielmente alguna de las dos propiedades principales del modelo. Esta política puede ser identificada fácilmente al revisar la prueba que Otter genera. ii) la lista de soporte (sos) esta vacía, lo cual nos indica que todas las políticas evaluadas satisfacen este modelo y iii) ninguna respuesta. Otter no tiene los elementos necesarios para decirnos si el conjunto de políticas escritas cumple o no con el modelo.

Si Otter nos arroja el último resultado, las políticas deberán ser verificadas utilizando a Mace. Mace construirá un modelo parecido al presentado en la sección anterior, sólo si las políticas cumplen con las reglas del modelo Bell- La Padula.

4.4 CONCLUSIONES.

Gracias a que el administrador de políticas realiza una formalización de las políticas siguiendo los parámetros del lenguaje seleccionado del trabajo de Halpern y Weissman podemos validar varios aspectos importantes de las políticas y su ambiente. Como se mostró en este capítulo,

el realizar estas validaciones con Otter/Mace nos permite tener un análisis más completo de las políticas.

El objetivo de que el administrador de políticas pueda realizar las validaciones mostradas en este capítulo utilizando Otter/Mace es permitir que el administrador de seguridad tenga los elementos necesarios para decidir si las políticas deben ser implementadas en la organización o no.

Para lograr que el administrador de políticas realice las validaciones mencionadas y nos ayude en la administración de las políticas se tuvieron que realizar varios cambios en la interfaz gráfica del administrador de políticas. Por tal motivo, en el siguiente capítulo se describen más a detalle todas las mejoras y capacidades adicionales que se tuvieron que hacer en la interfaz gráfica.

5 NUEVA INTERFAZ GRÁFICA PARA LA HERRAMIENTA DE SEGURIDAD.

5.1 INTRODUCCIÓN.

Partiendo de las observaciones y comentarios que se le hicieron al trabajo presentado por M en C Karen García Gamboa presentado en el capítulo 2, se decidió subsanar las limitantes que presentó la interfaz gráfica y además se le adicionaron una serie de capacidades con las cuales, es más fácil la validación y administración de las políticas de seguridad. Este capítulo tiene como objetivo mostrar cómo se resolvieron cada una de las limitaciones y explicar en qué consiste cada una de estas capacidades adicionales.

Primero, se explicará cómo se cambio el manejador de bases de datos de la interfaz para poder probar el rendimiento de dicha herramienta con bases de datos de organizaciones medianas a grandes[5].

En lo que se refiere a la captura de políticas, la nueva interfaz permite capturar las políticas a través de listas de control de acceso o mediante una navegación dirigida por medio de ventanas. Además, en este capítulo se muestra cómo se capturan las políticas del modelo Bell-La Padula en la interfaz.

Otra capacidad con la cuenta la interfaz gráfica consiste en la adición de un módulo que nos permita tener la capacidad de edición y revisión de las políticas de seguridad y de cada uno de los elementos que componen el ambiente.

[5] En base al volumen (espacio en disco) que ocupa una base de datos, se considera pequeña, mediana o grande. No existe un estándar a este respecto. Sin embargo en base a mi experiencia, yo considero que una base de datos pequeña es aquella que ocupa que tiene un volumen menor de un 1 Gb, media de entre un 1Gb y 5 Gb y grande mayor de 5 Gb.

Igualmente, se explicará cómo a través de la interfaz podemos realizar generalizaciones de políticas, esto es cambiar una política particular a una política general según los requerimientos que se vayan dando dentro de la organización.

Las políticas son formalizadas por la herramienta de acuerdo a los parámetros del lenguaje seleccionado del trabajo de Halpern y Weissman. Posteriormente, para tener un análisis más completo de las políticas se han adicionado una serie de módulos en la interfaz que nos permitirán no sólo utilizar a Otter (cómo lo venía haciendo el administrador de políticas organizaciones de [8]) sino también a Mace.

5.2 BASE DE DATOS.

Una sugerencia que se le hizo a la herramienta presentada por M en C Karen García Gamboa fue que debería probarse su rendimiento y conocer su flexibilidad al capturar políticas con una base de datos grande.

El administrador de políticas de seguridad organizacionales presentado por Karen García Gamboa [8] maneja una base de datos relacional creada en *Access de Microsoft*. Esto es una limitante para poder probar el rendimiento de la herramienta con una base de datos grande ya que Access de Microsoft [20] es un manejador de bases de datos para escritorio, que maneja bases de datos de tamaño pequeño a mediano.

Para resolver este obstáculo se decidió cambiar el manejador de la base de datos de Access a SQL Server. Este manejador nos permite soportar bases de datos muy grandes y con numerosas transacciones. Además dicha base de datos se puede manejar desde un servidor en forma distribuida, y no en forma local como se hace con el administrador de políticas organizacionales presentado por Karen García Gamboa [8].

La base datos tiene un papel muy importante en la herramienta de seguridad. Sus principales funciones son almacenar cada uno de los elementos del ambiente y tomar dichos elementos para escribir las políticas de seguridad.

Las principales tablas que utiliza nuestro administrador de políticas son las siguientes:

- Tabla Sujeto: Esta tabla permite llevar un registro de los empleados de la empresa. Por tal motivo en esta tabla se introduce el nombre del sujeto, su puesto y al departamento al que pertenece.

- Tabla Objeto: Esta tabla es muy importante para crear las políticas ya que administra la información de la empresa. Cada registro contiene el tipo de información al departamento que pertenece y la clasificación que tiene.

- Tabla Verbos: En esta tabla se guardan las acciones que puede realizar un sujeto sobre un objeto. Además lleva la relación entre acciones equivalentes (sinónimos).

Las tablas *Departamento, Puesto1, Importanciainf y MecSeg* 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 y los mecanismos de acceso a la información.

En la siguiente figura se muestran el modelo entidad-relación de la base de datos que maneja el administrador de políticas desarrollado en esta tesis.

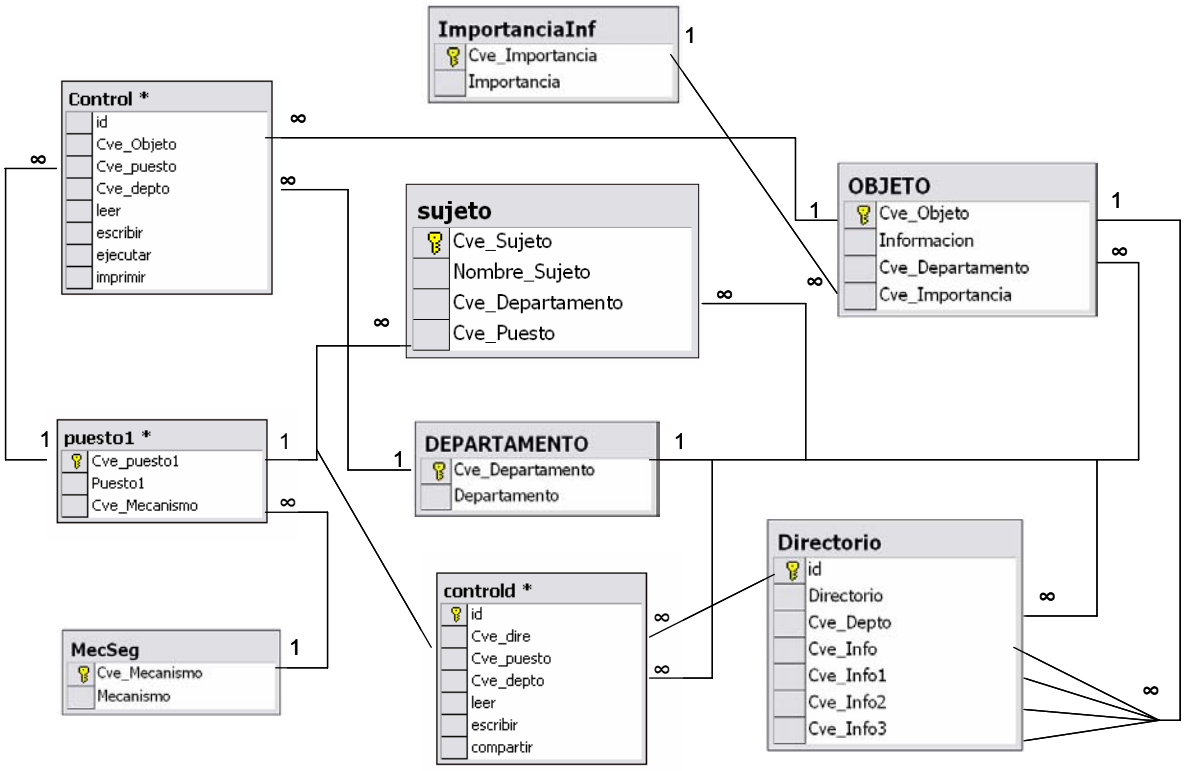

Figura 5.1 Tablas principales de la base de datos y sus relaciones

En las siguientes secciones de este capítulo se muestran los cambios y las mejoras realizadas a la interfaz para capturar las políticas de seguridad.

5.3 CAPTURA DE LAS POLÍTICAS DE SEGURIDAD.

Una forma sencilla de capturar las políticas sería mediante 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. Como consecuencia resulta sumamente difícil manejar un proceso semejante ya que podrían encontrarse interpretaciones erróneas de lo que realmente el usuario intentó escribir.

Por tal motivo, el administrador de políticas organizaciones presentado por M en C Karen García Gamboa realiza la captura de las políticas mediante ventanas dirigidas en las que el usuario va seleccionado el sujeto, acción y objeto de la política. Sin embargo, antes de

seleccionar cada uno de estos elementos el usuario debía dibujar un nodo grafico representativo de cada elemento.

Las limitantes que se encontraron al evaluar la forma en que se capturan las políticas el administrador de políticas organizacionales de [8] fueron 2. Primero que dicha tarea era tediosa ya que al capturar la acción de una política en el segundo nivel de sub-ventanas el usuario se confunde. Esta confusión se debe a que el usuario no ve claramente la opción que debe seleccionar. Y segundo, que el dibujar un grafo antes de seleccionar cada elemento de la política también genera confusión y por lo tanto es innecesario.

Nuestro administrador de políticas resuelve estas dos limitante. La primera restricción se solucionó rediseñando cada una de las ventanas que aparecen en la selección de cada elemento de la política. Este diseño consistió en crear ventanas sencillas y fáciles de entender y usar por los usuarios de tal forma que no le generen confusión sobre la opción que debe escoger en ese momento.

La segunda restricción se subsanó eliminando que el usuario tenga que dibujar un nodo gráfico para capturar cada elemento de la política. Sino que más bien, el usuario genera la política seleccionando cada elemento mediante sub-ventanas que lo van dirigiendo y limitando en el proceso de escritura y al final la interfaz dibujará automáticamente este grafo. Lo anterior permitirá al administrador del sistema verificar si la política escrita es correcta.

Además se le adicionó a nuestra herramienta de seguridad la capacidad de escribir las políticas mediante una lista de control de acceso. Para lograr lo anterior, la interfaz cuenta con una ventana en la cual el usuario va seleccionando los permisos que tiene cada usuario sobre un objeto en especial.

Otra capacidad adicional con la que cuenta la interfaz es poder capturar políticas que se rijan por el modelo Bell – La Padula.

Resumiendo, nuestro administrador de políticas tiene dos formas de capturar políticas de seguridad referentes a la información: mediante una lista de control de accesos y/o mediante ventanas dirigidas. Por otro lado, el administrador de políticas puede capturar políticas que cumplan con el modelo Bell- La Padula. Por lo tal motivo, en las siguientes secciones se describe cada una de estas formas.

5.3.1 CAPTURA DE POLÍTICAS A TRAVÉS DE LISTAS DE CONTROL DE ACCESO.

Como se mencionó anteriormente, se le agregó al administrador de políticas realizado en esta tesis, la opción de poder generar políticas mediante listas de control de acceso. Esto se debe a que las listas de control de acceso es una técnica en donde se crea una lista por cada objeto (información) colocando los permisos que cada sujeto tiene sobre el. Esta técnica permite ver con mayor facilidad quien tiene acceso a una información específica.

Nuestra herramienta de seguridad maneja dos niveles en los permisos. El primer nivel consiste en los permisos que recaen directamente sobre el archivo y el segundo nivel consiste en los permisos que recaen sobre el directorio que contiene los archivos.

Para crear políticas de seguridad utilizando esta técnica, el administrador del sistema deberá seleccionar en la interfaz el menú de *"Políticas"* y en el submenú de políticas de control de acceso la opción *"Archivos"* para seleccionar los permisos sobre un archivo en especial o la opción *"Directorios"* para seleccionar los permisos sobre un directorio en donde se encuentre estos archivos.

Si se selecciona la opción de archivos, la pantalla que se presentará será la siguiente:

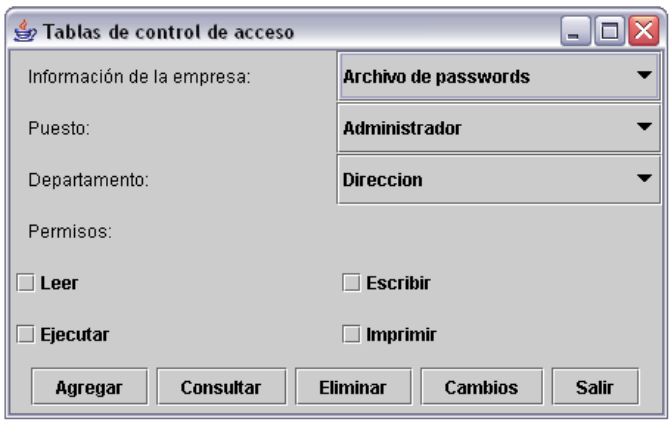

Figura 5.2 Pantalla para dar de alta un política de control de acceso de la información

Como se puede observar la lista de control de acceso se crea seleccionando primeramente el tipo de información. Una vez seleccionada, se escoge el puesto y el departamento de la persona que tiene permisos sobre esta información y para terminar se seleccionan los permisos que dicho sujeto tendrá sobre esta información.

Los modos de acceso (permisos) sobre los archivos que maneja el administrador de políticas son los siguientes:

- Lectura: el usuario únicamente puede leer o visualizar la información contenida en un archivo.
- Escritura: en este tipo de acceso el usuario tiene permitido agregar datos, modificar y salvar la información.
- Ejecución: este acceso otorga al usuario el privilegio de ejecutar programas.
- Impresión: imprimir la información.

Estos modos de acceso pueden seleccionarse en forma simple o combinarse. Se realizó la tarea de revisar cada una de las posibles combinaciones que se puede obtener teniendo estos 4 modos de acceso. De tal forma que el administrador de políticas maneja las siguientes combinaciones:

Acciones sobre archivos.	
Combinación	**Descripción**
Leer y escribir.	Permite al usuario visualizar el archivo y realizar modificaciones en el mismo y posteriormente salvarlo.
Leer y ejecutar.	Si el archivo es ejecutable, se podrá ejecutar y visualizar su código.
Leer e imprimir.	Permite al usuario visualizar un archivo y si lo desea imprimirlo.
Leer, escribir y ejecutar.	Si el archivo es ejecutable, se podrá ejecutar y visualizar su código y si el usuario lo desea modificarlo.
Leer, ejecutar e imprimir.	Si el archivo es ejecutable, se podrá ejecutar y visualizar su código y si el usuario lo desea imprimirlo.
Leer, escribir e imprimir.	Permite al usuario visualizar el contenido del archivo, realizar modificaciones en el mismo y si lo desea imprimirlo.
Leer, escribir, ejecutar e imprimir.	Permite al usuario realizar todas las operaciones sobre el archivo.

Tabla 5.1 Tabla de las combinaciones permitidas sobre archivos.

Por otro lado, si se selecciona la opción de directorios, la pantalla que mostrará la interfaz será muy parecida a la pantalla para dar de alta una política de control de acceso de archivos (figura 5.2).

Los modos de acceso sobre directorios que se están considerando son siguientes:

- Leer: permite al usuario visualizar el contenido del directorio.
- Escribir: permite al usuario realizar las siguientes operaciones: crear un archivo, borrarlo, copiarlo, moverlo y renombrarlo.
- Compartir: permite que el usuario pueda compartir a otros usuarios la información contenida en su directorio.

De igual forma que los permisos sobre archivos, los modos de acceso sobre directorios se pueden combinar. A continuación se presenta la siguiente tabla en la cual se pueden observar las combinaciones posibles:

Acciones sobre directorios.	
Combinación	**Descripción**
Leer y escribir.	El usuario puede visualizar el contenido del directorio y modificar los archivos del directorio si lo desea.
Leer y compartir.	El usuario puede visualizar el contenido del directorio y compartirlo con otros usuarios.
Leer, escribir y compartir.	El usuario puede visualizar el contenido del directorio, compartir el directorio con otros usuarios y modificarlo si lo desea.

Tabla 5.2 Tabla de las combinaciones permitidas sobre directorios.

Una vez que las políticas de control de acceso sobre archivos y/o directorios son seleccionadas, éstas se almacenan en la tabla “Control” y “Controld” en la base de datos respectivamente.

5.3.2 CAPTURA DE POLÍTICAS A TRAVÉS DE LA INTERFAZ.

Para escribir las políticas a través de sub-ventanas que la interfaz va presentado se siguió un esquema el cual utiliza las reglas gramaticales de nuestro idioma. Dichas reglas consisten en que para crear una oración debemos tener un sujeto + verbo (acción) + objeto.

En nuestro caso, el sujeto será los usuarios o los miembros de la organización. Para seleccionar al sujeto de la política el administrador del sistema debe seleccionar en la interfaz la opción de *"Políticas sobre la información"* en el submenú *"Generar"* del menú *"Políticas"*. Una vez selecciona esta opción se presentará la siguiente pantalla:

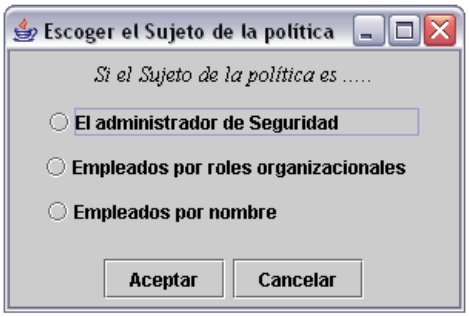

Figura 5.3 Pantalla para seleccionar el Sujeto de la política

Como se puede observar, a través de esta ventana se tienen tres opciones para escoger al sujeto de la política. La primera opción es cuando se va escribir una política sobre el administrador de seguridad.

Si se selecciona la segunda opción, se podrá seleccionar al sujeto por el puesto que tenga en la empresa o por departamento al que pertenezca. Sin embargo, esta opción no sólo nos permite escribir políticas sobre un sujeto en particular sino que también nos permite escribir políticas generales sobre un puesto (por ejemplo, todos los gerentes de la empresa) o por departamento (por ejemplo, todos los empleados del departamento *x*).

Y por último si se selecciona la tercera opción, ésta nos permite seleccionar el sujeto por su nombre. A continuación se muestra la pantalla que se abre al seleccionar esta última opción.

Figura 5.4 Pantalla para seleccionar el Sujeto de la política por nombre

Esta última opción ya existía en el administrador de políticas organizacionales presentado por [8]. El objetivo de continuar con esta opción es que la selección del sujeto sea más sencilla y flexible. Sin embargo, en nuestro administrador de políticas además, se le adicionó la capacidad de que NO se genere la política teniendo como sujeto el nombre de una persona. Lo anterior se debe a que las políticas se escriben en base al puesto o rol que dicho sujeto tiene en la empresa.

Para explicar mejor lo anterior, supongamos que el usuario selecciona el sujeto de la política a través de esta opción (por nombre) y que por ejemplo, escoge al empleado "Anna Álvarez". Entonces, la herramienta de seguridad se conectará a la tabla "sujeto" de la base de datos y buscará el puesto y el departamento que tiene el empleado en cuestión (en este caso Anna Álvarez es gerente de sistemas). Una vez, que la herramienta conoce el puesto y el departamento de dicho empleado, tomará estos datos como el sujeto y no su nombre. Por lo tanto, el sujeto de la política, en nuestro caso será "El gerente de sistemas" y no "Anna Álvarez".

Una vez seleccionado el sujeto, aparecerá una pantalla donde se podrá escoger las acciones que el sujeto realizará sobre el objeto (información). En este caso, la herramienta de seguridad se conectará a la tabla "Verbos" de la base de datos y desplegará los siguientes verbos:

Acciones sobre la información	
Clave	Verbo
1	Acceder
2	Agregar
3	Almacenar
4	Asignar
5	Autentificar
6	Backup´s
7	Borrar
8	Cambiar
9	Cambiar Nombre
10	Cargar
11	Compartir
12	Copiar
13	Cortar
14	Crear
15	Dar
16	Designar
17	Duplicar
18	Ejecutar
19	Eliminar
20	Escribir
21	Generar
22	Guardar
23	Imprimir
24	Ingresar
25	Insertar
26	Leer
27	Modificar
28	Mover
29	Proteger
30	Quitar
31	Renombar
32	Resguardar
33	Respaldar
34	Salvar

Tabla 5.3 Tabla de las acciones sobre la información.

Una vez seleccionada la acción, la herramienta de seguridad le presentará al usuario una pantalla, en la cual el usuario podrá seleccionar el tipo de objeto de la política. Nos referimos al objeto de la política como la estructura de datos que está almacenada en el sistema, esto es la información. La pantalla que se despliega es la siguiente:

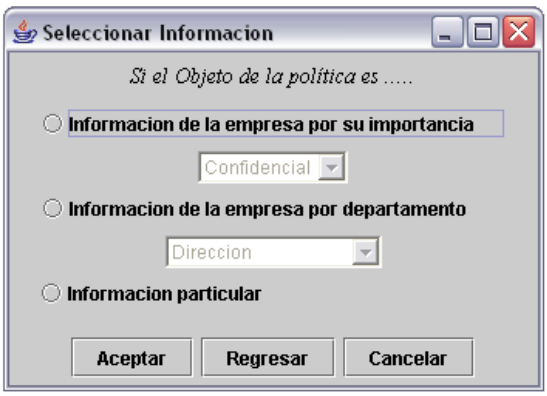

Figura 5.5 Pantalla para seleccionar la información

Como se puede observar en la figura 5.5, el objeto de la política podrá ser: a) la información de la empresa por su importancia (confidencial, general, importante) o b) información de la empresa por departamento o c) una información en particular (estados financieros, nomina, etc.).

Una vez escrita la política la interfaz mostrará la siguiente pantalla, en la cual el administrador del sistema podrá confirmar la política que acaba de generarse (con el botón *"Aceptar"*).

Figura 5.6 Pantalla final de la política

Además, en esta pantalla se dibuja un grafo en el cual se muestra el sujeto, acción y objeto de la política. En este caso, la acción de la política esta representada por una línea que une al sujeto con el objeto y esta línea podrá ser de color verde si la política es afirmativa o podrá ser roja si la política es negativa. De esta forma el administrador de la herramienta podrá identificar visualmente si la política es afirmativa o negativa.

Si se selecciona el botón *Verificar* la herramienta formalizar la política y ejecutará a Otter/Mace. Esta función tiene como objetivo que ratificar la concordancia de la política que se acaba de crear.

5.3.3 CAPTURA DE LAS POLÍTICAS BELL- LA PADULA.

La captura de las políticas que se rigen por el modelo de Bell – La Padula se realiza de manera muy sencilla. El usuario del sistema deberá seleccionar en el menú "*Políticas*" el submenú "*Bell- La Padula*", en la cual aparecerá una sola ventana.

En dicha ventana, se podrá seleccionar el sujeto de la política, la acción que puede o no realiza sobre el objeto (recordemos que este modelo sólo tiene dos modos de acceso: leer y escribir) y por último se seleccionará el objeto (información) donde recaerá dicha acción. A continuación se muestra esta ventana.

Figura 5.7 Pantalla para crear una política Bell-La Padula

5.4 EDICIÓN Y REVISIÓN.

La edición y revisión de las políticas introducidas en la herramienta así como de los elementos que componen su ambiente juega un papel muy importante en la administración de las políticas de seguridad. Debido a que las organizaciones están en constante cambio es necesario que la herramienta cuente con un módulo dentro de la interfaz que le permita al administrador del sistema de una forma muy sencilla y funcional dar de alta, consultar, borrar o modificar un elemento de la organización o una políticas de seguridad.

Otra limitante que encontramos en el administrador de políticas organizacionales presentado por M en C Karen García Gamboa fue que la consulta, modificación y eliminación de política así como de los elementos necesarios para construirla (departamentos, empleados, etc.) resulta un trabajo difícil para el usuario ya que dichos operaciones se encontraban en menús separados en la interfaz.

Nuestro administrador de políticas resuelve esta limitante al permitir al usuario realizar estas operaciones en la interfaz a través de una misma ventana, sin necesidad de navegar en diferentes menús.

Para facilitar nuestra explicación se ha dividido esta sección en subsecciones. Primeramente se mostrará cómo se realizan todas estas operaciones con el contexto de las políticas (ambiente) y posteriormente con las políticas de seguridad.

5.4.1 EDICIÓN Y REVISIÓN DE LOS ELEMENTOS QUE COMPONEN EL AMBIENTE.

Dentro de la interfaz gráfica se ha creado el módulo "*Catálogos"* el cual permitirá al administrador del sistema generar, consultar, eliminar y modificar cualquiera de los elementos que componen a la empresa u organización (es decir, el ambiente) como son: los empleados, los departamentos, puestos, etc.

Por ejemplo, si se desea editar o revisar los empleados de la empresa, la pantalla que el administrador de políticas mostrará será la que se muestra a continuación. Como se podrá observar en esta figura, en este mismo módulo se tiene la opción de realizar todas las operaciones (agregar un registro, consultar, eliminar ó cambiar los registros ya existentes) necesarias para la administración de los recursos.

Figura 5.8 Pantalla para el catálogo de empleados

Para la edición y revisión de cada uno de los elementos (departamentos, puestos, clasificación de la información, tipo de información, etc.) que conforman el ambiente, se obtendrá una pantalla similar a la anterior.

5.4.2 EDICIÓN Y REVISIÓN DE LAS POLÍTICAS DE SEGURIDAD.

Debido a que la generación de políticas mediante las listas de control de acceso es una capacidad nueva en la interfaz de nuestra herramienta, también se le adicionó a la interfaz la capacidad de editarlas y revisarlas dentro de la misma pantalla (ver figura 5.2)

En lo que se refiere a la consulta, la interfaz presentará una lista de control de acceso, por ejemplo, si se desea consultar quienes y que tipo de permisos tienen sobre los estados financieros, la pantalla que se desplegará en la interfaz será la siguiente:

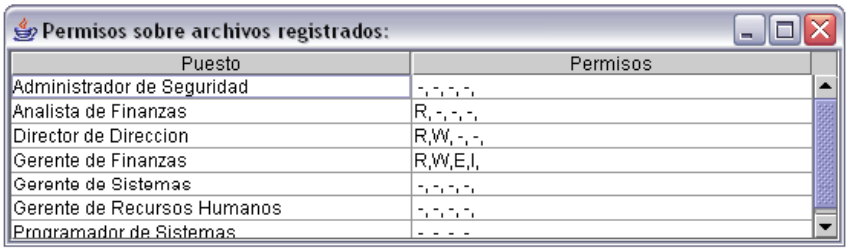

Figura 5.9 Lista de control de acceso sobre los estados financieros de la empresa

Como se puede observar en la figura 5.9, se despliegan todos los puestos existentes en la empresa y sus permisos en relación con este tipo de información y algunos de éstos no tienen permisos, sin embargo el administrador del sistema únicamente introducirá en el administrador de políticas los puestos que SI tengan permisos y la herramienta automáticamente pondrá a los restantes sin permisos.

De igual forma que la realización de consultas, se podrá seleccionar las operaciones de eliminar alguna política ya existente o modificarlas. Todo lo anterior se puede realizar en la misma ventana (figura 5.2) y el administrador de políticas se conectará a la tabla Control de la base de datos para realizar los comandos correspondientes.

Por otro lado, para realizar alguna operación de edición y revisión de políticas escritas mediante la interfaz y las políticas que siguen el modelo de Bell- La Padula se deberá entrar al menú "*Políticas*", en el cual se podrá seleccionar la opción "*Consultar*", "*Eliminar*" o "*Modificar*" según sea el caso.

5.4.5 GENERALIZACIÓN DE LAS POLÍTICAS.

Debido a que las organizaciones o empresas son entes dinámicos, es decir, están en constante cambio, el administrador de políticas realizado en esta tesis permite que se pueda generalizar una política. Dicha generalización consiste en cambiar una política que originalmente está escrita para un sujeto en especial (por ejemplo, para el gerente de sistemas) a una política general (por ejemplo, todos los gerentes) a través la interfaz gráfica. Esta opción es una capacidad nueva que se le adicionó a nuestra interfaz

Para clarificar este punto, supongamos que originalmente se creó una política que dice que "El gerente de sistemas puede salvar toda la información confidencial". Sin embargo, después de un tiempo y por convenir a los intereses de la organización, dicha política debe ser remplazada por "todos los gerentes puedan salvar toda la información confidencial".

En sí, esta nueva capacidad nos permite generalizar el sujeto de una política. El sujeto puede ser generalizado por puesto (por ejemplo, "el gerente *x"* a "todos los gerentes de la empresa") o por departamento (por ejemplo, "el empleado *x*" a "todos los empleados del departamento *x* ").

En la interfaz, podemos generalizar las políticas de las listas de control de acceso para archivos y directorios. Y también las políticas particulares (aquellas que involucran a un sujeto).

Para realizar lo anterior, se deberá entrar al submenú "*Conversión*" del menú "*Políticas*". En este submenú deberá de seleccionar que tipo de política se desea generalizar. A continuación se muestra la pantalla que aparecerá en la interfaz.

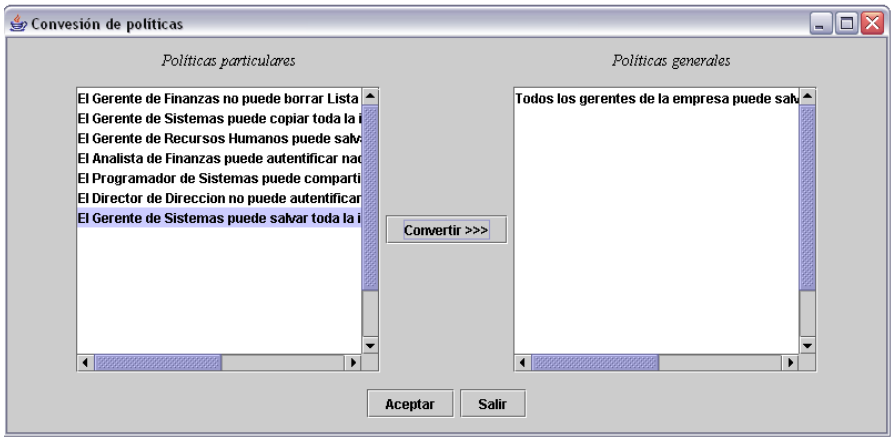

Figura 5.10 Pantalla de conversión de una política particular a una general

Como se puede observar en la figura 5.10, en la ventana izquierda se tienen las políticas particulares que se han introducido al sistema, una vez seleccionada la política que se desea cambiar, se debe seleccionar el botón "*Convertir*" y entonces aparecerá del lado derecho la política ya convertida a general. Sin embargo, dicha conversión únicamente se llevará a cabo hasta que se seleccione el botón "*Aceptar*". En ese momento la política será borrada de la base de datos de las políticas particulares y será adiciona a la base de datos de la políticas generales.

5.5 VALIDACIÓN DE LAS POLÍTICAS DE SEGURIDAD.

Una característica importante de la herramienta de seguridad es poder validar las políticas. En el capítulo anterior, se dividieron estas validaciones en dos partes: ambiente y políticas. De igual forma esta sección estará divida en dos subsecciones en las cuales, se mostrará cómo se valida el ambiente y las políticas utilizando nuestra interfaz gráfica.

La capacidad de poder realizar las validaciones de las políticas y su ambiente a través de la interfaz, es una capacidad nueva que se le agregó a la interfaz ya que el administrador de políticas organizacionales que presentó M en C Karen García Gamboa no cuenta con esta opción.

5.5.1 VALIDACIONES DEL AMBIENTE.

Para las validaciones del ambiente, se le agregó a la interfaz un módulo llamado "*Probar ambiente*" en el cual, se podrán hacer verificaciones de los empleados (sujetos) o de la información de la empresa (objetos). La pantalla que aparecerá al entrar a este módulo será la siguiente:

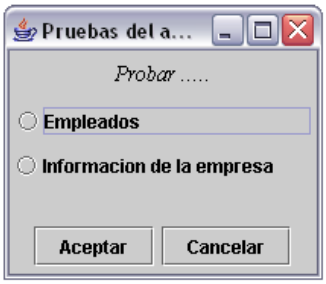

Figura 5.11 Pantalla para probar el ambiente de las políticas

Una vez seleccionado cualquiera de las dos opciones anteriores, la interfaz le mostrará al administrador del sistema otra ventana, en la cual se irá formulando la pregunta (validación) que se desee realizar.

Cuando la validación del ambiente está formulada, entonces el administrador de políticas la traducirá a lógica de primer orden (como se explico en el capítulo 5) y la mostrará en pantalla para que posteriormente nuestra herramienta ejecute a Otter/Mace.

5.5.2 VALIDACIONES DE LAS POLÍTICAS.

Como sabemos el administrador de políticas puede realizar las siguientes validaciones: i) verificar qué sujeto (s) tienen ciertos permisos sobre un objeto en especial, ii) verificar si un sujeto puede o no hacer una acción x, iii) verificar si las políticas escritas son consistentes y iv) verificar si las políticas escritas satisfacen el modelo de Bell – La Padula.

Para realizar las dos primeras, se ha adicionado en nuestra interfaz un módulo llamado "*Validación de políticas*". Al entrar a este módulo se desplegará la siguiente pantalla:

Figura 5.12 Pantalla para verificar políticas

Al seleccionar la primer opción, la interfaz mostrará una ventana (muy similar a la ventana de la figura 5.4) en la cual se seleccionará el nombre del sujeto. Posteriormente, aparecerá otra ventana en la cual el administrador del sistema seleccionará la acción que realizará sobre el objeto. Y por último, aparecerá una ventana en la cual el administrador seleccionará el objeto (información) en el cual recae dicha acción.

Una vez seleccionado todo lo anterior, el administrador de políticas escribirá en lógica de primer orden la pregunta correspondiente, la cual será de la forma - *((-) Puede (x,Accion(y))) y* ejecutará a Otter/Mace para realizar la validación.

Para validar "Qué sujetos tienen ciertos permisos sobre un objeto en especial", a la interfaz se le ha adicionado un modulo el cual sólo utilizará dos ventanas muy similares a las que utiliza la validación anterior. Estas ventanas tienen como objetivo el crear una pregunta de la forma *Puede(x,Accion(y)) | $ANS(X)* para posteriormente evaluarla con Otter.

En el administrador de políticas organizaciones presentado por M en C Karen García está muy confusa la forma de validar la concordancia de las políticas, ya que únicamente se colocó un botón con el que se ejecuta Otter. Y dicho botón no tiene ninguna indicación al respecto por lo que un usuario que no esté familiarizado con el administrador de políticas organizacionales no sabrá como realizar esta tarea.

Para resolver lo anterior, en nuestra interfaz se han adicionado una serie de módulos que nos permiten de forma clara y precisa verificar la concordancia de políticas de la información y de las políticas de siguen el modelo de Bell – La Padula.

Estas validaciones se realizan de dos formas en la interfaz. La primer forma es después de escribir una nueva política (consultar la sección 6.3.2) y la segunda forma es entrando al módulo *"Políticas de la información"*.

Con la segunda opción, la herramienta formaliza todas las políticas que se tienen y se muestra en pantalla dicha formalización. Entonces la herramienta ejecutará a Otter/Mace para su validación.

5.6 OTTER/MACE EN LA INTERFAZ

Debido a que todas las validaciones se realizan utilizando Otter o Mace o ambos. En la siguiente sección se explica cómo la nueva interfaz manda ejecutar estos programas.

5.6.1 OTTER.

Para realizar las validaciones de las políticas de seguridad utilizando Otter, se ha adicionado a la interfaz gráfica la opción de escoger cualquiera de estas dos interfaz gráficas: Otter/Mace y XOtter. Para tener acceso a estas interfaces, el administrador deberá entrar al menú de *"Formalización"* y seleccionar la opción *"Otter/Mace"* o la opción *"XOtter"*.

El adicionar esta capacidad a nuestra interfaz tiene como objetivo que el administrador del sistema puede trabajar con la interfaz que más le satisfaga. Por ejemplo, la interfaz Otter/Mace está programada en java y permite utilizar a Otter y su complemento Mace de una manera muy sencilla, sin embargo se requiere que el usuario tenga conocimientos de lógica de primer orden y de programación en Otter. Por otro lado, la interfaz XOtter está programada en Visual Basic y surge de un proyecto de ciencias de la computación e inteligencia artificial de la universidad de Sevilla. Dicha interfaz está más orientada para personas que no tienen muchos conocimientos de programación con Otter ya que por default tiene activadas las banderas básicas y la ayuda siempre está presente auxiliando al usuario en la utilización de Otter. Ambas interfaces son de distribución libre y gratuita.

5.6.2 MACE.

Para utilizar a Mace a través de la interfaz, el administrador del sistema tiene dos alternativas. La primera es utilizando la interfaz gráfica de Otter/Mace, la cual ya se comento en la sección anterior y la segunda es seleccionando *"Mace en línea"* en la interfaz.

La interfaz gráfica de Otter/Mace maneja una ventana para seleccionar las opciones con las que se desea trabajar con Mace, sin embargo el formato de salida para la visualización de los modelos que construye Mace es el formato para sistemas prolog[6] (parámetro P de Mace). Dicho formato es muy difícil de leer para aquellos usuarios que no están familiarizados con este lenguaje.

Por tal motivo, dentro del administrador de políticas realizado, se implemento en la interfaz de la herramienta de seguridad un módulo (*"Mace en línea")* que le permite al administrador del sistema personalizar los parámetros de configuración de Mace para ejecutarlo. La pantalla que aparece al seleccionar esta opción es la siguiente:

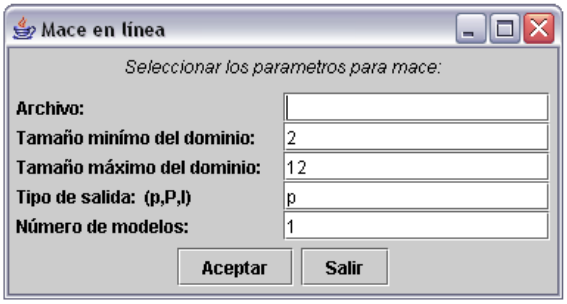

Figura 5.13 Pantalla para utilizar mace en línea

En esta ventana se puede observar que se necesitan escribir los parámetros de configuración de Mace. Primero, se le da la ruta y el archivo de entrada. Posteriormente, se debe seleccionar el tamaño del dominio. Este dominio tiene un rango entre un valor mínimo y un valor máximo. Por default esta entre 2 y12.

El siguiente parámetro a configurar es el formato en que Otter imprime el (los) modelo(s) encontrado(s). Mace maneja 3 formatos de salida: I ,p y P. El parámetro I, Mace imprime el modelo en formato IVY [21], el cual sirve como entrada a un sistema preprocesador y verificador de lógica de primer orden creado por el mismo autor de Mace. El parámetro p, se

[6] Prolog es un lenguaje de programación simple pero poderoso desarrollado en la Universidad de Marsella y Edimburgo como una herramienta práctica para programación lógica.

imprime el modelo como una tabla de verdad muy sencilla de leer. Y el parámetro P, genera un modelo que puede ser utilizado como entrada para sistemas que utilicen prolog.

El último parámetro a configurar se refiere a la cantidad de modelos que se desea que Mace busque, por default se tiene el valor de 1.

Una vez, que Mace encuentra el modelo, la interfaz abre el archivo de salida en formato Word. Este archivo, tiene el mismo nombre que el archivo de entrada, pero con la extensión ".out" y se localiza en el directorio c:\mace\bin.

5.7 CONCLUSIONES.

La interfaz gráfica juega un papel muy importante dentro de la herramienta de seguridad ya que su objetivo principal es facilitar la tarea de la administración de políticas. Por tal motivo, se dedicó un especial tiempo para hacer un correcto análisis. Este análisis consistió encontrar las deficiencias de la interfaz realizada en el administrador de políticas organizacionales presentado por [8]. Una vez identificadas las limitantes de dicha interfaz se procedió no sólo a eliminarlas sino a agregarle nuevas capacidades.

6 EVALUACIÓN DEL RENDIMIENTO DEL ADMINISTRADOR DE POLÍTICAS, CONCLUSIONES Y TRABAJO FUTURO.

6. 1 INTRODUCCIÓN.

El objetivo de este capítulo es determinar la magnitud de la degradación en el desempeño del administrador de políticas. Para lograr lo anterior, creamos una empresa ficticia de 114 empleados con 231 políticas de dicha organización.

Por otro lado, para probar si las políticas de seguridad que se rigen por el modelo Bell – La Padula si cumplen con las dos principales propiedades de este modelo, capturamos en el administrador de políticas un escenario militar. Con este escenario, medimos también el rendimiento del administrador de políticas al evaluar estas políticas.

Además en este capítulo realizaremos un análisis de los tiempos de procesamiento de Otter y Mace. Y por último, expondremos nuestras conclusiones al respecto y hablaremos del trabajo futuro.

6.2 EMPRESA FICTICIA.

Nuestra empresa ficticia "X" se dedica a la comercialización de equipo de oficina, principalmente equipo de impresión. La empresa cuenta con 114 empleados quienes ocupan los siguientes roles:

1. Abogado
2. Administrador
3. Analista
4. Auditor
5. Capturista
6. Contador
7. Contralor
8. Director
9. Gerente
10. Ingeniero A
11. Ingeniero B
12. Mensajero
13. Operador A
14. Operador B
15. Programador
16. Recepcionista
17. Secretaria
18. Telefonista
19. Vendedor

Los departamentos con los cuales cuenta esta empresa son:

1. Administración.
2. Atención a Clientes.
3. Contabilidad.
4. Cuentas por cobrar.
5. Dirección.
6. Finanzas.
7. Jurídico.
8. Logística.
9. Mercadotecnia.
10. Producción.
11. Recepción.
12. Recursos humanos.

13. Seguridad informática.
14. Sistemas.
15. Soporte técnico.
16. Ventas.

Para proteger la información de la empresa el administrador de seguridad divide la información de acuerdo a su importancia en ultra secreta, secreta, clasificada y no clasificada. Además, la información que esta empresa maneja es la siguiente:

1. Análisis de mercados.
2. Análisis de riesgos.
3. Archivo de passwords.
4. Balance general.
5. Cartera de clientes.
6. Contratos con clientes.
7. Contratos con proveedores.
8. Estados financieros.
9. Facturas.
10. Inventarios.
11. Lista de asistencia.
12. Nomina.
13. Ordenes de trabajo.
14. Plan de contingencia.
15. Procedimientos de recuperación.
16. Programas.
17. Remisiones.
18. Reportes de clientes.
19. Reportes de servicio
20. Reportes de ventas.

Los datos anteriores fueron capturados en el administrador de políticas. Posteriormente la herramienta generó el ambiente. Primero validamos el ambiente. Estas validaciones se describen en las siguientes secciones.

6.3 VALIDACIONES DEL AMBIENTE.

Recordemos que el administrador de políticas puede probar dos aspectos importantes del ambiente utilizando Otter. Estos aspectos son a) los empleados y b) la información.

En lo que se refiere a los empleados, validamos que cada uno de los empleados pertenecieran a un departamento. El administrador de seguridad genera la siguiente fórmula:

```
all x (-(exists y (Departamento(y) & Empleado(x,y)))).
```

Posteriormente, el administrador de políticas mediante un botón de la interfaz gráfica ejecuta a Otter. La verificación se detuvo en el momento en que fue encontrada la cláusula vacía. En este caso, nos va a encontrar una prueba (cláusula vacía) por cada empleado que este relacionado con un departamento. El tiempo de procesamiento ocupado por el demostrador en esta validación fue de 1.97 segundos, en cambio cuándo se había capturado en la empresa juguete (mostrada en el capítulo 4) se obtuvo un tiempo de procesamiento de 0.12 segundos.

De acuerdo a lo anterior, se puede concluir que cuanto mayor sea el ambiente más tiempo le llevará a Otter encontrar una prueba.

La salida que nos arroja Otter de un sólo empleado[7] fue la siguiente:

```
---------------- PROOF ----------------

13 [] Departamento(Produccion).
446 [] Puesto(Zeleny_Meza_Roberto,Operador_A,Produccion).
447 [] -Puesto(x,y,z)|Empleado(x,z).
705 [] -Departamento(x)| -Empleado(y,x).
1442 [hyper,446,447] Empleado(Zeleny_Meza_Roberto,Produccion).
1444 [hyper,1442,705,13] $F.

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

Al analizar esta prueba vemos que la cláusula vacía se encuentra al momento en que Otter verifica que en el departamento Producción (línea 13) existe un empleado (Zeleny Meza Roberto). Es decir que dicho empleado está relacionado con este departamento (línea 1442).

[7] Otter encuentra una prueba por cada empleado, en este caso 114 pruebas.

Otra validación que realizamos es verificar que no exista ningún empleado que esté en dos departamentos. Para realizar esta validación, el administrador de políticas genera la siguiente fórmula:

```
all x y z (Empleado(x,y) & Empleado(x,z) -> y=z).
```

Entonces cuando la herramienta ejecuta a Otter, nos arroja el siguiente resultado en un tiempo de procesamiento de 1.38 segundos.

```
Search stopped because sos empty.
```

Este resultado nos indica que no existe ningún empleado que esté relacionado con dos departamentos a la vez. Si por error, se hubiera capturado el mismo empleado en dos departamentos al ejecutar Otter, éste se detendría encontrando la cláusula vacía por cada empleado duplicado.

Se realizo ésta misma prueba al ejemplo juguete mostrado en el capítulo 4 y se obtuvo un tiempo de procesamiento de 0.17 segundos.

Por otro lado, en lo que respecta a la información, podemos validar utilizando Otter, por ejemplo, qué información pertenece al departamento de finanzas. Para realizar dicha validación el administrador debe formular la siguiente pregunta: ¿Qué información pertenece al departamento de finanzas utilizando la interfaz gráfica de nuestra herramienta.

Inmediatamente, nuestro administrador de políticas la traduce a lógica de primer orden adicionando el predicado $ANS, que le indica a Otter que evalué lo anterior y nos de como respuestas todas las pruebas que encuentre. La pregunta anterior queda de la siguiente forma:

```
-Informacion1(y,Finanzas)|$ANS(y).
```

Posteriormente, el administrador de políticas ejecuta Otter, el cual da la siguiente respuesta:

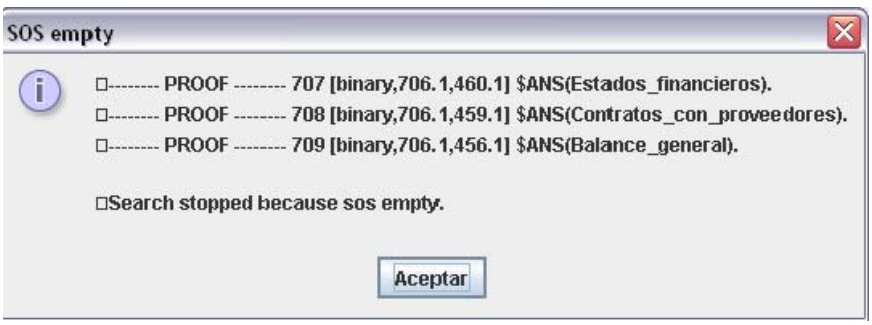

Fig. 6.1. Pantalla que despliega la validación de la información

Como se puede observar, Otter nos está respondiendo que la información que pertenece al departamento de finanzas son los estados financieros, los contratos con proveedores y el balance general. El tiempo de procesamiento de Otter para darnos estas respuestas fue de 0.41 segundos.

Se realizó ésta misma prueba utilizando el ejemplo juguete descrito en el capítulo 4 y se obtuvo un tiempo de procesamiento de 0.05 segundos. Al compara los tiempos de procesamiento que obtuvimos al realizar esta prueba en cada ambiente (empresa ficticia y ejemplo juguete) concluimos que conforme el ambiente sea mayor más tiempo tardará Otter en darnos todas las respuestas.

Además de las validaciones anteriores, podemos verificar la concordancia lógica del ambiente utilizando Otter/Mace. Si el ambiente es consistente, Otter se detendrá y nos indicará que la lista de soporte esta vacía. En el caso de Mace, éste construirá un modelo.

En el caso contrario, si el ambiente no es consistente, al momento de que el administrador de políticas ejecuta a Otter, éste se detendrá al encontrar una prueba (cláusula vacía). En este caso, el administrador de seguridad deberá analizar el resultado que Otter nos arrojó para identificar los elementos del ambiente que se están contradiciendo. Y al ejecutar a Mace, éste no podrá construir un modelo debido a que el ambiente no es consistente y nos dará el siguiente mensaje:

The search is complete. No Models were found

En nuestro caso, cuando la herramienta ejecutó a Otter, éste tardo 1.79 segundos en detenerse y avisarnos que la lista de soporte está vacía. Mace construyó un modelo en un tiempo de proceso de 0.65 segundos.

Al evaluar el ambiente del ejemplo juguete, Otter tardo 0.25 segundos en avisarnos que ya no tenía más cláusulas para inferir. Por otro lado, Mace construyó el modelo en un tiempo de 0.02 segundos.

6.4 VALIDACIONES DE LAS POLÍTICAS.

Una vez que verificamos que el ambiente que se generó al capturar los datos de nuestra empresa ficticia, procedimos a introducir las políticas de dicha organización. Se capturaron 115 políticas de control de acceso sobre archivos; 97 políticas de control de acceso sobre archivos; 7 políticas particulares; 6 políticas generales y 6 políticas sobre el administrador de seguridad informática. En total fueron 231 políticas.

En las siguientes secciones se describe cómo podemos validar las 231 políticas de esta empresa ficticia utilizando Otter/Mace. Recordemos que con el administrador de políticas podemos realizar las siguientes cuatro validaciones: i) verificar qué sujeto(s) tiene(n) ciertos permisos sobre un objeto en especial, ii) verificar si un sujeto puede o no hacer una acción *x*, iii) verificar si las políticas escritas son consistentes y iv) verificar si las políticas escritas satisfacen el modelo de Bell – La Padula.

6.4.1 VERIFICAR QUÉ SUJETO(S) TIENE(N) CIERTOS PERMISOS SOBRE UN OBJETO EN ESPECIAL.

Con esta validación podemos saber en forma rápida quién o quienes tienen permisos sobre un objeto (información) en especial. Por ejemplo, en el caso de nuestra empresa ficticia podemos verificar qué empleado(s) puede(n) leer el balance general de la empresa.

Para realizar esta validación el administrador de políticas genera la pregunta ¿Quién puede leer el balance general? mediante la interfaz gráfica. Posteriormente, nuestro administrador de políticas formaliza dicha pregunta de la siguiente forma:

```
-Puede(x,Leer(Balance_general)) | $ANS(x).
```

Mediante un botón de la interfaz grafica, la herramienta ejecuta Otter. La respuesta que obtuvimos fue la siguiente:

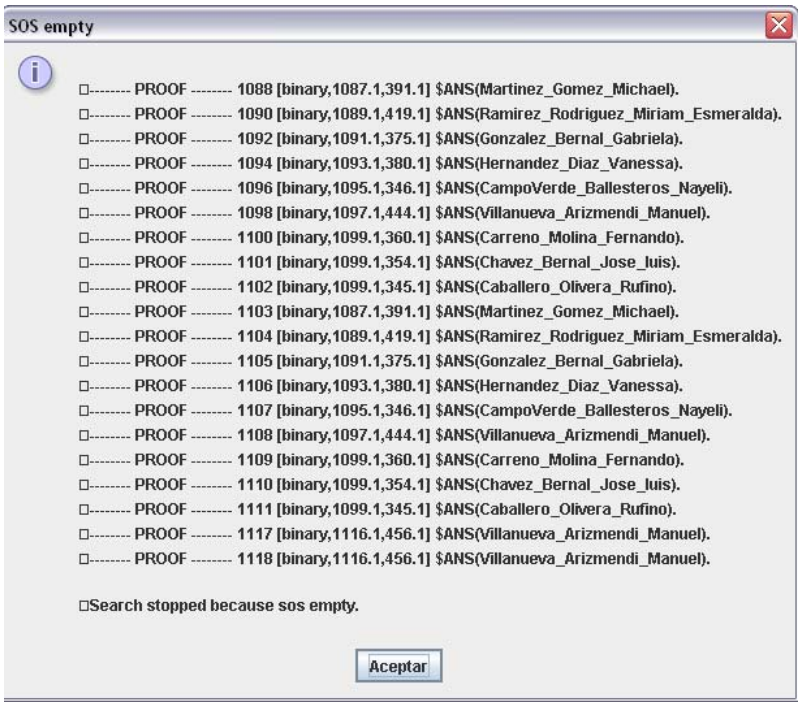

Fig. 6.2. Resultado de Otter para validar quien tiene permisos sobre los Estados financieros

Si observamos la respuesta que nos dio Otter, podemos ver que los empleados que pueden leer el balance General de la empresa son los empleados del departamento de finanzas y de dirección. El tiempo de procesamiento de Otter para darnos esta respuesta fue 0.98 segundos. Al realizar esta misma prueba con el ambiente juguete, obtuvimos un tiempo de procesamiento de 0.03 segundos. Dicho tiempo como podemos ver es mucho menor que el tiempo de procesamiento de la empresa ficticia.

6.4.2 VERIFICAR SI UN SUJETO PUEDE HACER UNA ACCIÓN *X* .

Para realizar esta validación el administrador de políticas le da a Otter/Mace un archivo de entrada el cual contiene $E_0 \wedge E_1 \wedge P_1 \wedge \ldots \wedge P_n \wedge \neg puede(S, A)$. Siendo S un individuo y A la acción que realiza dicho individuo sobre un objeto en especial.

Para ejemplificar lo anterior, validaremos si el Gerente de finanzas de nuestra empresa ficticia puede modificar los estados financieros de la empresa. Para formular lo anterior, el administrador de seguridad realiza la pregunta ¿puede el Gerente de finanzas (Martínez Gómez Michael) modificar los estados financieros? utilizando la interfaz gráfica de nuestro administrador de políticas. Inmediatamente, la herramienta traduce dicha pregunta a lógica de primer orden, quedando de la siguiente forma:

```
Puede(Martinez_Gomez_Michael,Escribir(Estados_financieros)).
```

Cómo se puede observar, el administrador de políticas sustituye la acción "Modificar" por "Escribir" utilizando la tabla de sinónimos de forma interna. Dicha tabla tiene el objetivo el evitar redundancias y ambigüedades en acciones equivalentes.

En nuestro caso, podemos ver que dicha sustitución es correcta ya que como sabemos el modificar un archivo es equivalente a escribir en él. A continuación, ejecutamos a Otter (llamándolo a través de un botón de la interfaz) para realizar esta validación y el resultado que nos arroja es el siguiente:

```
Search stopped because sos empty.
```

El tiempo de procesamiento para darnos esta respuesta fue de 12 minutos, ya que Otter busca una prueba utilizando todas las técnicas de inferencia con las que cuenta.

Al realizar esta validación utilizando Mace, esta herramienta construyó un modelo en un tiempo de 10 minutos aproximadamente. Cuándo introdujimos en la herramienta la empresa juguete, Otter tardaba 0.5 segundos en parar sin encontrar la cláusula vacía y Mace construyó el modelo en un tiempo de 60 segundos. Esta comparación nos reitera que conforme las políticas de seguridad de una organización se incrementan, los tiempos de procesamiento de Otter/Mace también aumentarán.

Con el objetivo de verificar lo que NO puede realizar un sujeto, negamos la pregunta anterior, es decir, ahora a Otter/Mace les preguntaremos si ¿El Gerente de finanzas (Martínez Gómez Michael) no puede modificar los estados financieros?

De antemano, sabemos que la respuesta de Otter para esta pregunta será encontrar la cláusula vacía ya que ya comprobamos mediante la prueba anterior que el gerente de finanzas SI puede modificar los estados financieros. En el caso de Mace, sabemos que la respuesta que obtendremos será que éste no podrá construir un modelo.

Para verificar lo anterior, nuestro administrador de políticas formaliza la pregunta anterior de la siguiente forma:

```
- Puede(Martinez_Gomez_Michael,Escribir(Estados_financieros)).
```

Al llamar ejecutar Otter mediante la interfaz de nuestra herramienta, la salida que obtenemos es la siguiente:

```
---------------- PROOF ----------------
391 [] Puesto(Martinez_Gomez_Michael,Gerente,Finanzas).
679 [] -Puede(x,RWI(y))|Puede(x,Escribir(y)).
800 [] -Puesto(x,Gerente,Finanzas)|Puede(x,RWI(Estados_financieros)).
1071 [] -Puede(Martinez_Gomez_Michael,Escribir(Estados_financieros)).
4173[hyper,391,800] Puede(Martinez_Gomez_Michael,RWI(Estados_financieros)).
15995[hyper,4173,679]
Puede(Martinez_Gomez_Michael,Escribir(Estados_financieros)).
15996 [binary,15995.1,1071.1] $F.
------------ end of proof ------------
```

El tiempo que le llevó a Otter encontrar esta prueba fue de 39.14 segundos. Al realizar ésta misma prueba utilizando la empresa juguete descrita en el capítulo 4, Otter nos dio un tiempo de procesamiento de 0.15 segundos.

Cuando ejecutamos a Mace mediante la interfaz de nuestra herramienta, la salida que obtenemos es la siguiente:

The search is complete. No Models were found

Los resultados obtenidos en esta última validación nos indican que la pregunta que estamos realizando se contradice con la política de control de acceso de los estados financieros, por lo tanto el gerente de finanzas puede modificar los estados financieros.

6.4.3 VERIFICAR SI LAS POLÍTICAS ESCRITAS SON CONSISTENTES.

En esta sección mostraremos los resultados de verificar la consistencia lógica de las políticas utilizando el administrador de políticas.

Para lograr lo anterior, consideremos qué en esta empresa ficticia se tienen 15 gerentes (uno por cada departamento), los cuales pueden leer, escribir y compartir la información de su departamento. Por tal motivo, en la herramienta se han capturado mediante listas de control de acceso sobre directorios una política por cada gerente la cual indica que el gerente del departamento *x* puede leer, escribir y compartir la información de su departamento. Teniendo lo anterior en mente y para probar la consistencia lógica de las políticas, escribimos una

política utilizando la interfaz de nuestra herramienta que diga que el "El gerente de Recursos humanos puede copiar la nomina de la empresa". La política anterior, la formalizada nuestro administrador de políticas de la siguiente forma:

```
all x y (Puesto(x,Gerente,Recursos_humanos) &
Informacion1(y,Recursos_humanos) -> Puede(x,EscribirD(y))).
```

Como esta política no se contradice con la lista de control de acceso de la nomina de la empresa, al evaluarla usando Otter, éste nos responde:

```
Search stopped because sos empty.
```

El tiempo de procesamiento de Otter para evaluar esta política fue de 13 minutos y Mace construyó un modelo en un tiempo de 10 minutos aproximadamente. En el caso de la empresa juguete descrita en el capítulo 4, el tiempo de procesamiento para evaluar el conjunto de políticas fue de 0.10 segundos y su modelo Mace lo construyó en 0.02 segundos.

Tomando este mismo escenario, ahora escribimos una política utilizando la interfaz del administrador de políticas que diga que "El Gerente de sistemas no puede eliminar a la información de su departamento". Dicha política se contradice con la lista de control de acceso del directorio de sistemas.

Una vez capturada, el administrador de seguridad formalizará esta política de la siguiente forma:

```
all x y (Puesto(x,Gerente,Sistemas) & Informacion1(y,Sistemas) -> -
Puede(x,EscribirD(y))).
```

Al ejecutar a Otter, el administrador obtiene y entrega la siguiente respuesta:

```
---------------- PROOF ----------------

335 [] Puesto(Alvarez_Bejar_Anna_Valeska,Gerente,Sistemas).
468 [] Informacion2(Programas,Sistemas,Ultra_Secreta).
473 [] -Informacion2(x,y,z)|Informacion1(x,y).
626 [] Estan(Programas,DSeguridad).
700 [] -Puede(x,RWSd(y))|Puede(x,EscribirD(y)).
1373 [] -Puesto(x,Gerente,Sistemas)| -Estan(y,DSeguridad)|Puede(x,RWSd(y)).
1449 [] -Puesto(x,Gerente,Sistemas)| -Informacion1(y,Sistemas)| -
Puede(x,EscribirD(y)).
1960 [hyper,626,1373,335]
Puede(Alvarez_Bejar_Anna_Valeska,RWSd(Programas)).
6746 [hyper,468,473] Informacion1(Programas,Sistemas).
```

```
6753 [ur,6746,1449,335] -
Puede(Alvarez_Bejar_Anna_Valeska,EscribirD(Programas)).
8521 [hyper,1960,700]
Puede(Alvarez_Bejar_Anna_Valeska,EscribirD(Programas)).
8522 [binary,8521.1,6753.1] $F.

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

Otter encontró esta prueba en un tiempo de 11.50 segundos. Como podemos observar, Otter encuentra la contradicción al probar si el gerente de sistemas puede borrar los programas (información del departamento de sistemas) en las líneas 6753 y 8521. Al realizar éste tipo de pruebas utilizando el ambiente juguete descrito en el capítulo 4, se obtuvo un tiempo de procesamiento de Otter de 0.13 segundos.

Por otro lado, al verificar esta política utilizando Mace (Mace se manda ejecutar mediante la interfaz del administrador de políticas), el resultado que obtenemos es:

The search is complete. No Models were found

Al analizar los anteriores resultados se concluye que la política sobre el gerente de sistemas se contradice por lo deberá ser revisada y modificada antes de ser implementada en esta organización.

6.4.5 VERIFICAR SI LAS POLÍTICAS ESCRITAS SATISFACEN EL MODELO DE BELL- LA PADULA.

El implementar el modelo de Bell- La Padula en el administrador de políticas implica que las políticas que se escriban mediante la interfaz de la herramienta, se deben de regir por las dos propiedades principales de este modelo.

Para probar este modelo introducimos en el administrador de políticas un ambiente militar con los siguientes puestos:

1. Teniente General.
2.- General.
3.- Coronel.
4.- Teniente Coronel.
5.- Mayor.
6.- Capitán.

7.- Teniente Primero.
8.- Teniente Segundo.
9.- Alferez.
10.- Suboficial mayor.
11.- Sargento.
12.- Cabo.

Cada uno de estos rangos militares tiene una autorización (permisos), la cual puede ser ultra secreta, secreta, clasificada y no clasificada. En nuestro caso, el Teniente general y el Coronel tienen permisos de ultra secreta; el Capitán y el Teniente primero tienen permisos de secreta; el Sargento tiene permisos de clasificada y el Cabo de no clasificada.

La información que se capturó tiene una clasificación que es ultra secreta, secreta, clasificada y no clasifica. Dicha información es la siguiente:

1.- Contraespionaje, la cual es secreta.
2.- Control de armamento, la cual es clasificada.
3.- Informes políticos, la cual es No clasificada.
4.- Operaciones militares, la cual es ultra secreta.

Teniendo en cuenta este escenario, se capturaron las siguientes políticas para verificar si siguen fielmente el modelo de Bell-La Padula.

1.- El Teniente General puede leer el archivo de contraespionaje.
2.- El Capitán puede escribir los informes políticos.

El administrador de políticas formaliza estas políticas de la siguiente forma:

```
all x (Puesto(x,Teniente_General) -> Puede(x,Leer(Contraespionaje))).
all x (Puesto(x,Capitan) -> Puede(x,Escribir(Informes_politicos))).
```

Al evaluar la primera política utilizando Otter obtenemos el siguiente resultado en un tiempo de procesamiento de 0.47 segundos.

```
Search stopped because sos empty.
```

Mediante la interfaz del administrador de políticas ejecutamos Mace para evaluar esta política y Mace nos construye un modelo en un tiempo de 0.02 segundos.

Al obtener los anteriores resultados, se concluye que la primer política no se contradice con el modelo de Bell – La Padula ya que el Teniente General al tener permisos de ultra secreta y aplicando la primer propiedad puede leer información con su misma clasificación y menor, es decir, puede leer información que tenga clasificación ultra secreta, secreta, no clasificada y clasificada.

Al evaluar la segunda política utilizando Otter obtenemos el siguiente resultado en un tiempo de procesamiento de 0.27 segundos.

```
---------------- PROOF ----------------
81 [] Puesto(Gilberto_Sosa,Capitan).
88 [] -Puesto(x,Capitan)|Permisos(x,Secreta).
104 [] Secreta>No_Clasificada.
114 [] Informacion2(Informes_politicos,No_Clasificada).
123 [] -Permisos(x,y)| -Informacion2(z,u)| -(y>u)| -Puede(x,Escribir(z)).
130 [] -Puesto(x,Capitan)|Puede(x,Escribir(Informes_politicos)).
164 [hyper,81,130] Puede(Gilberto_Sosa,Escribir(Informes_politicos)).
167 [hyper,81,88] Permisos(Gilberto_Sosa,Secreta).
198[ur,167,123,114,104] -Puede(Gilberto_Sosa,Escribir(Informes_politicos)).
199 [binary,198.1,164.1] $F.
------------ end of proof -------------
```

Al momento que Otter encuentra una prueba nos está indicando que esta política se contradice con la segunda propiedad del modelo ya que el capitán al tener permisos de secreta no puede escribir información de menor nivel que el suyo, en este caso los informes políticos. Debido a esta contradicción al ejecutar Mace, éste no puede construir un modelo.

6.5 ANÁLISIS DE LOS TIEMPOS DE PROCESAMIENTO DE OTTER Y MACE.

El objetivo de la presente sección es poder comparar los tiempos de procesamiento que Otter y Mace arrojan al realizar las validaciones de las políticas con nuestra herramienta. Para

realizar lo anterior, se partió del ejemplo juguete presentado en esta tesis en el capítulo 4 el cual consta de 10 empleados y 4 tipos de información, con lo que se generó 20 políticas. Este ambiente se fue incrementado hasta llegar a la empresa ficticia presentada en el capítulo 6 creándose en total 5 escenarios. La descripción general de estos escenarios se muestra a continuación:

No Escenario	No Empleados	No Información	Políticas
1	10	4	20
2	35	8	70
3	70	12	150
4	95	16	197
5	114	20	230

Tabla 6.1. Descripción de los escenarios

Una vez capturado cada uno de estos escenarios en el administrador de políticas, se procedió a realizar las validaciones tanto del ambiente como de las políticas registrando los tiempos de procesamiento de Otter y Mace para posteriormente poder analizarlos mediante gráficas.

6.5.1 ANÁLISIS DE LAS VALIDACIONES DEL AMBIENTE.

La primer validación que se realizó del ambiente consistió en verificar que cada uno de los empleados pertenecieran a un departamento. El administrador de seguridad generó la siguiente fórmula:

```
all x (-(exists y (Departamento(y) & Empleado(x,y)))).
```

Como sabemos, esta validación se realiza con Otter y los tiempos de procesamiento que Otter arrojó en cada uno de los escenarios fue el siguiente:

No Escenario	Tiempo de procesamiento [8]
1	0.12
2	0.45
3	0.95
4	1.23
5	1.97

Tabla 6.2. Tiempos de procesamiento del ambiente utilizando Otter

[8] El tiempo de procesamiento esta en segundos

Graficando lo anterior obtuvimos:

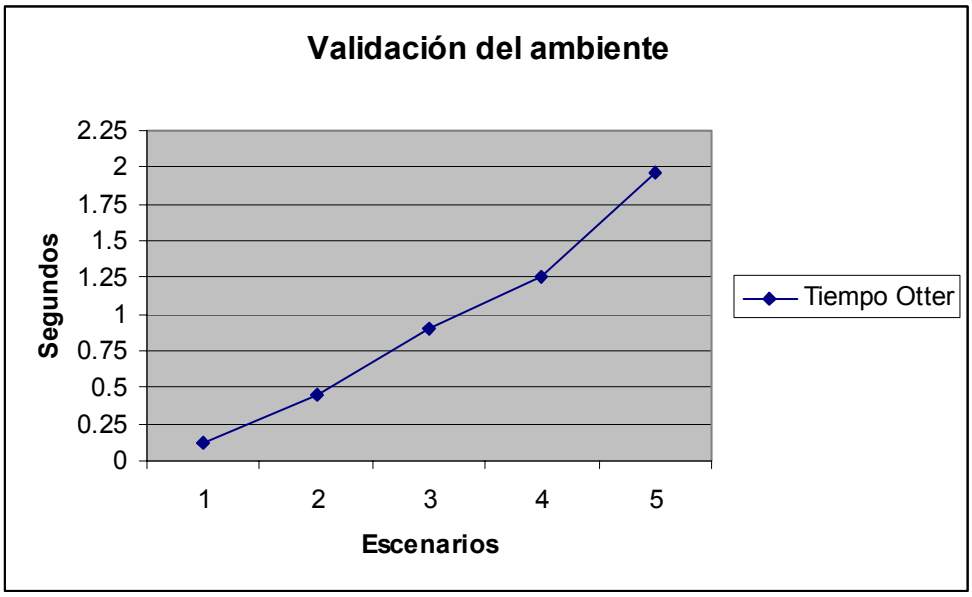

Fig. 6.3. Gráfica de la validación del ambiente utilizando Otter

Como se puede observar en esta gráfica los tiempos de procesamiento al realizar esta validación del ambiente con Otter se fueron incrementando conforme el ambiente fue creciendo en cada escenario.

6.5.2 VALIDACIÓN DE LA CONSISTENCIA DEL AMBIENTE.

Otra validación que podemos realizar al ambiente se refiere a verificar que éste sea consistente, es decir que no tenga elementos que se contradigan. Para realizar esta validación utilizamos a Otte/Mace y los resultados que obtuvimos en cada escenario fueron los siguientes:

No Escenario	Tiempo de procesamiento Otter	Tiempo de procesamiento Mace
1	0.25	0.019
2	0.67	0.032
3	1.13	0.038
4	1.55	0.05
5	1.79	0.065

Tabla 6.2. Tiempos de procesamiento para validar el ambiente utilizando Otter/Mace

Para realizar un análisis de estos valores, los graficamos de la siguiente forma:

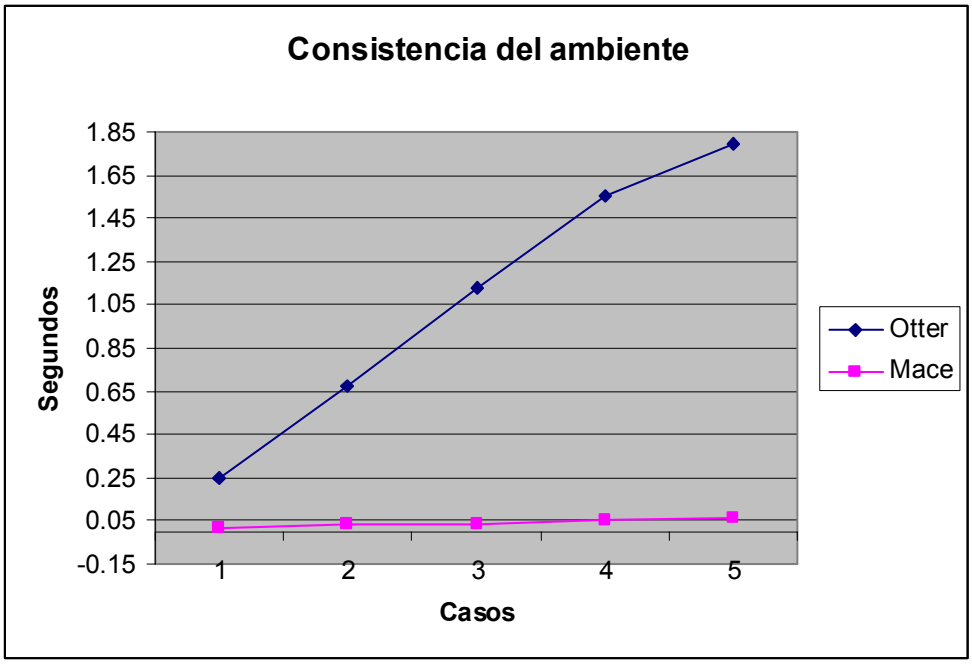

Fig. 6.3. Gráfica de la validación de la consistencia del ambiente utilizando Otter/Mace

Como se puede observa en esta gráfica el tiempo de procesamiento de Otter se incrementó conforme el ambiente de las políticas fue creciendo. También al analizar esta gráfica podemos concluir que Mace nos da una respuesta (si el ambiente es consistente o no) más rápida que Otter.

Sin embargo, en esta gráfica no podemos analizar el comportamiento de Mace, debido a la escala de tiempos que maneja. Por tal motivo, la siguiente figura graficamos únicamente a Mace.

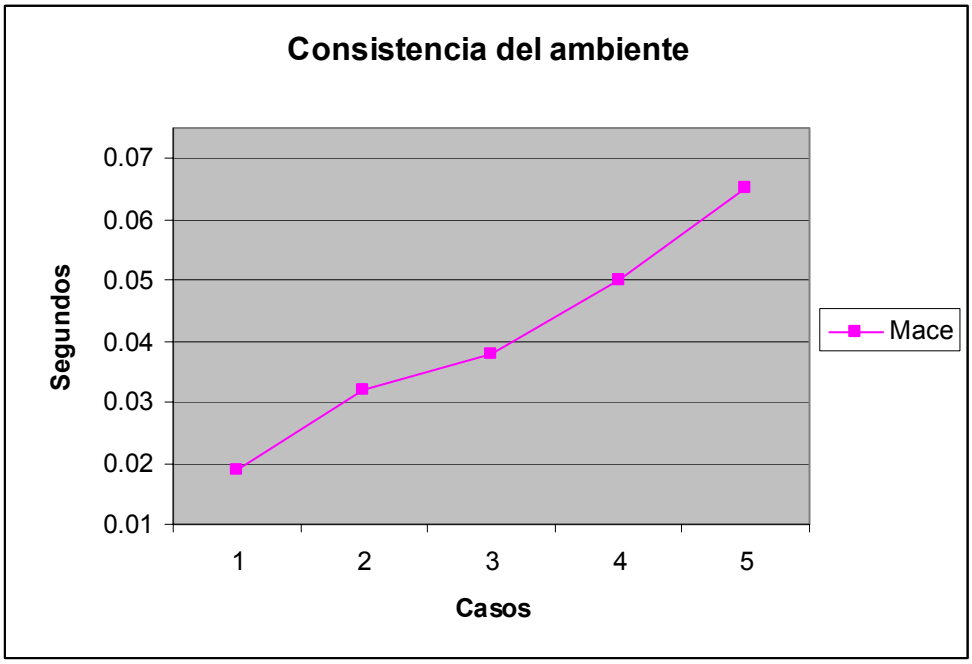

Fig. 6.4. Gráfica de la validación de la consistencia del ambiente utilizando Mace

Analizando el comportamiento de Mace, vemos que éste incrementa sus tiempo de procesamiento conforme el ambiente a evaluar es mayor.

6.5.3 VALIDACIÓN DE LAS POLÍTICAS.

Una vez que se evaluó el ambiente de los cinco escenarios se empezó por evaluar a las políticas que se crearon en cada escenario. La validación de las políticas consistió en lo siguiente: verificar si un sujeto tiene permisos sobre un objeto en especial, verificar el perfil de un sujeto y sobre todo, verificar la consistencia de las políticas. Los resultados de obtuvimos se reportan en las siguientes secciones.

6.5.4 VERIFICAR QUÉ SUJETO(S) TIENE(N) CIERTOS PERMISOS SOBRE UN OBJETO EN ESPECIAL.

La primer validación que realizamos sobre las políticas de cada uno de los escenarios fue verificar que sujeto(s) tiene(n) permisos sobre el balance general de la empresa. En este caso el administrador de políticas generó la pregunta ¿Quién puede leer el balance general? mediante la interfaz gráfica y únicamente utilizamos a Otter para obtener la respuesta. Al evaluar cada uno de los escenarios, se obtuvo la siguiente tabla de resultados:

No Escenario	Tiempo de procesamiento
1	0.03
2	0.15
3	0.54
4	0.72
5	0.98

Tabla 6.3. Tiempos de procesamiento para validar las políticas utilizando Otter

Al graficar lo anterior, se obtiene:

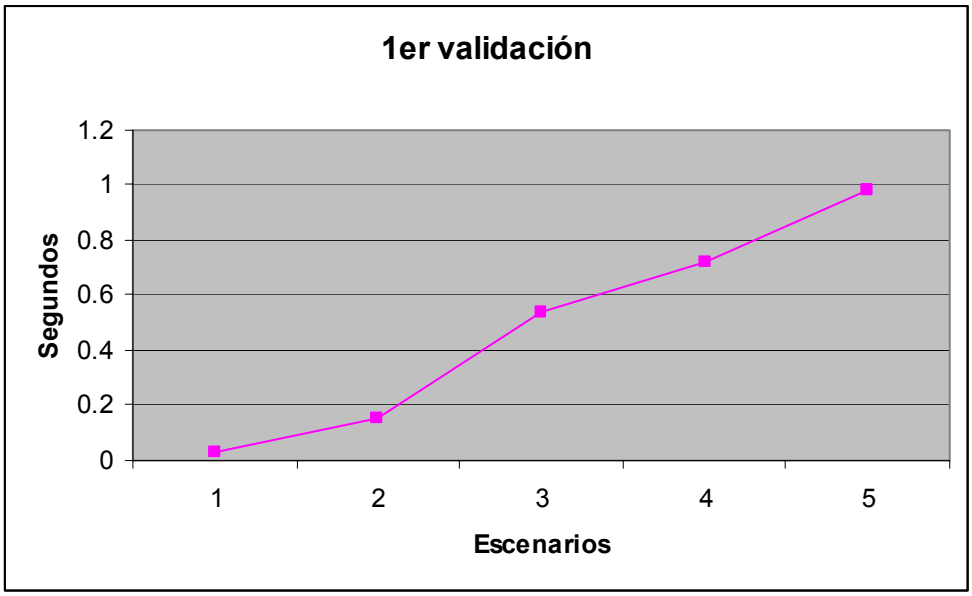

Fig. 6.5. Gráfica de la 1er validación de las políticas utilizando Otter

Al analizar esta gráfica observamos que conforme las políticas fueron creciendo Otter tardo más tiempo en darnos una respuesta. Este resultado es correcto debido a que como sabemos al crecer la cantidad de políticas dentro del sistema, Otter tiene más elementos a evaluar para darnos una respuesta.

6.5.5 VERIFICAR SI UN SUJETO PUEDE HACER UNA ACCIÓN *X* .

La siguiente validación que realizamos consistió en verificar el perfil de un sujeto. Para poder realizar esta validación le preguntamos a Otter/Mace a través de la interfaz del administrador de políticas si el Gerente de finanzas puede o no modificar los estados financieros de la empresa. En el caso de que el Gerente de finanzas si pueda modificar los estados financieros, los resultados que obtendríamos en cada uno de los escenario son los siguientes:

No Escenario	Tiempo de procesamiento Otter	Tiempo de procesamiento Mace
1	30	60
2	178	150
3	385	300
4	550	480
5	720	600

Tabla 6.4. Tiempos de procesamiento para validar las políticas utilizando Otter/Mace

Graficando lo anterior, tenemos:

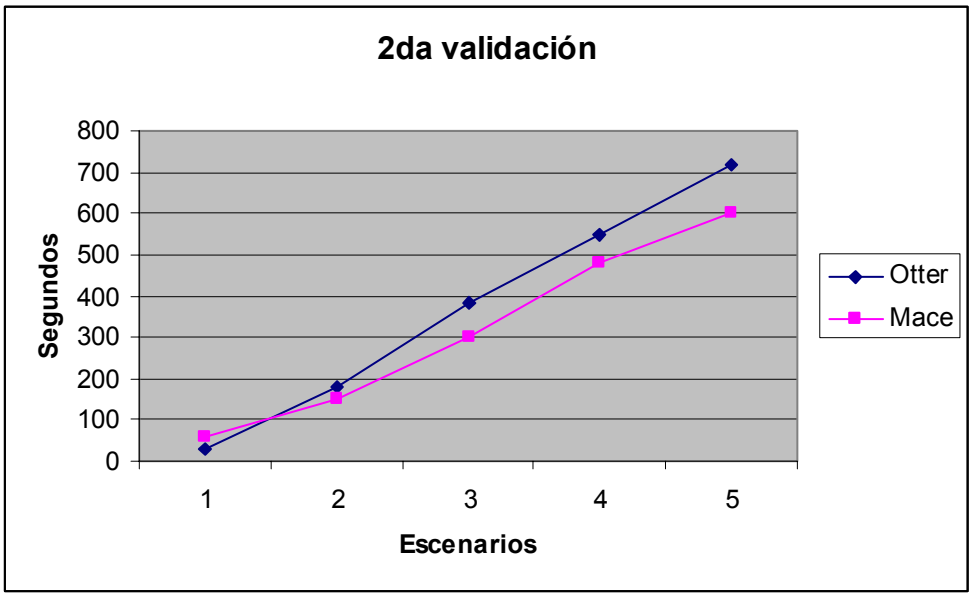

Fig. 6.6. Gráfica de la 2da validación de las políticas utilizando Otter/Mace

Analizando la gráfica anterior, observamos que Otter nos dio una respuesta más rápida que Mace pero sólo en el primer ambiente, ya que posteriormente vemos como conforme vamos teniendo un escenario más grande Mace tarda menos tiempo en construir un modelo.

6.5.6 VERIFICAR SI LAS POLÍTICAS ESCRITAS SON CONSISTENTES.

Para validar si las políticas son consistentes o no introducimos en el administrador de políticas la siguiente política "El Gerente de Recursos humanos puede copiar la nomina de la empresa". Si esta política no se contradice con el resto de políticas que ya se han capturado en el administrador, podremos esperar que Otter nos devuelva la lista de soporte vacía y que Mace nos encuentre un modelo. En este caso, los tiempos de procesamiento de Otter y Mace fueron:

No Escenario	Tiempo de procesamiento Otter	Tiempo de procesamiento Mace
1	0.01	0.02
2	60	45
3	420	385
4	595	480
5	780	600

Tabla 6.5. Tiempos de procesamiento para validar consistencia de las políticas utilizando Otter/Mace

Graficando lo anterior, tenemos:

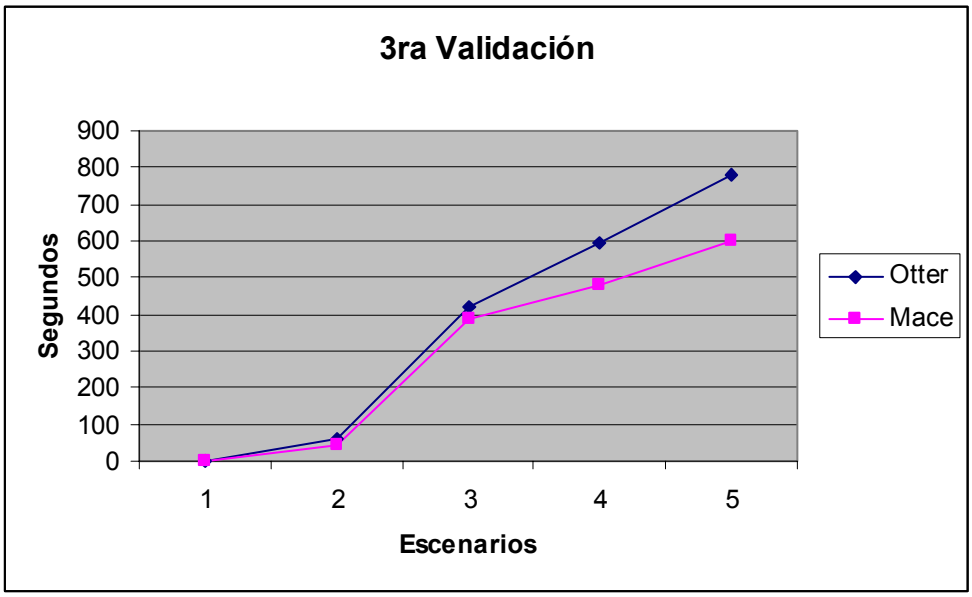

Fig. 6.7. Gráfica de la 3er validación de las políticas utilizando Otter/Mace

En esta gráfica podemos ver como Otter y Mace tienen un tiempo de procesamiento muy cercano en los dos primeros escenarios, sin embargo, conforme el escenario va creciendo el tiempo de procesamiento entre Otter y Mace se va separando cada vez más. Después de realizar un análisis del desempeño de Otter y Mace podemos concluir que Mace tiene mejores tiempos de procesamiento, es decir es más rápido en darnos una respuesta.

Revisamos algunas bibliografías [27] (sobre todo del autor de Otter William McCune y de Larry Wos) para encontrar algún análisis sobre la complejidad computacional de Otter. El resultado que esta búsqueda nos arrojó es que teóricamente se ha dicho muy poco sobre la complejidad computacional de Otter. Sin embargo como sabemos un problema en lógica de primer orden es indecidible. Cuando Otter tiene que resolver un problema en una subclase decidible de lógica de primer orden, éste puede tardarse un tiempo infinito e incluso nunca darnos una respuesta. En este caso, podemos decir que el algoritmo que utiliza Otter no es el mejor para resolver un problema. En cambio, si Otter encuentra una respuesta, la complejidad del algoritmo que Otter utiliza no es lineal.

6.6 CONCLUSIONES.

A lo largo del presente trabajo se ha recalcado la importancia que tiene la información dentro de una empresa. La pérdida o el mal uso de la información pueden ocasionar graves problemas para su organización. Dichos problemas van desde pérdidas económicas hasta

perdida de credibilidad y competitividad ante quienes les prestan un bien o servicio. Por tal motivo, las empresas han buscado diversas formas y técnicas para proteger la información.

Una forma de proteger la información de una empresa es mediante la creación e implementación de un conjunto de políticas de seguridad. Dichas políticas también son un punto de partida para una infraestructura apropiada. Sin embargo, no basta solamente con contar con dicho conjunto de políticas sino hay que garantizar su validez para que cumplan con su objetivo.

A partir de este hecho surge el trabajo presentado por Karen García Gamboa [8], el cual consistió en una herramienta que genere políticas y les de mantenimiento. Aunque esta herramienta cumple con su objetivo, un análisis detectó se detectó tres limitantes principales que son i) falta de semántica clara y precisa en la formalización de políticas, ii) razonamiento incompleto de las políticas al sólo utilizar Otter y iii) una interfaz en la cual resulta tediosa la interacción con las políticas.

El objetivo del presente trabajo consistió en subsanar estas limitantes. En lo que respecta a la primera de ellas nos basamos en el trabajo presentado por Halpern y Weissman [14], lo que nos facilitó el encontrar un sub-lenguaje de la lógica de primer orden que permitiera expresar las políticas con una sintaxis clara y una semántica precisa.

La segunda limitante de la herramienta fue resuelta al incluir en el razonamiento de políticas el uso de Mace. Consideramos muy importe incluir a Mace en el análisis que realiza el administrador de políticas debido a que ocasiones Otter puede tarda mucho tiempo (tiempo indefinido) en darnos un respuesta y necesitamos utilizar otra herramienta para saber si nuestras políticas son o no consistentes.

Además, usando Otter y Mace, complementamos el análisis de (in)consistencia de políticas y la demostración de otras propiedades como son: i) validar ciertas características importantes tanto de su contexto (ambiente) cómo de las políticas, ii) validar la consistencia lógica de las políticas y iii) implementar un modelo de seguridad, en este caso el modelo de Bell- La Padula para entonces poder validar que las políticas cumplan fielmente con las propiedades de este modelo.

El subsanar las limitantes que presentó la interfaz de [8] involucró un trabajo muy ingenioso

ya que se buscó tener una interfaz que sea sencilla de usar para cualquier usuario y sobre todo que la administración de las políticas no sea una tarea confusa y tediosa para quien use el administrador de políticas.

Debido a que las políticas de cualquier organización son de carácter confidencial, nos fue difícil probar el administrador de políticas en una empresa real. Por tal motivo creamos una empresa ficticia para poder probar el rendimiento del administrador de políticas. Consideramos que los resultados obtenidos al evaluar el administrador de políticas son muy satisfactorios por lo que pensamos que si en un futuro el administrador de políticas pueda ser implementado en una empresa real, éste funcionará de forma correcta.

Además, analizando el tiempo de procesamiento de Otter/Mace al evaluar las políticas, podemos observar que el tiempo se incrementa conforme el ambiente y el número de políticas es mayor. Por ejemplo, cuando se evaluó el ejemplo juguete (con 7 empleados y 20 políticas de seguridad) los tiempos de procesamiento tanto de Otter como de Mace fue de microsegundos. Al incrementar el ambiente y el número de políticas en la empresa ficticia (con 114 empleados y 231 políticas) este tiempo se incrementó de microsegundos a segundos.

De acuerdo a lo anterior, consideramos que si los tiempos de procesamiento de Otter/Mace se incrementan demasiado (en varias horas) debido a que la organización es muy grande[9]. Será necesario ejecutar el administrador de políticas en equipos de computo más robustos (tal vez, un servidor con más de un procesador, mayor memoria, etc.) para mejorar los tiempos de respuesta de Otter/Mace. Además, creemos que en un futuro será necesario buscar o desarrollar otras herramientas que puedan sustituir a Otter/Mace para realizar el análisis de las políticas en este tipo de organizaciones.

6.7 TRABAJO FUTURO.

Como trabajo futuro existen varios puntos a tratar, los cuales mencionamos a continuación:

a).- Usar el lenguaje natural para permitir que el usuario introduzca las políticas de seguridad de forma abierta, tal vez utilizando una caja de texto en lugar de ventanas dirigidas. b) Que la interpretación de los resultados obtenidos por el demostrador de teoremas sea realizado por la

[9] Consideramos organizaciones grandes a las empresas gubernamentales y transnacionales que cuentan con miles o millones de empleados.

herramienta en forma automática y no por el administrador de seguridad y c) poder administrar no sólo políticas de seguridad referentes a la información sino cualquier tipo de políticas como son las políticas de firewalls, ruteadores, etc.

En lo que respecta al último punto en la siguiente subsección realizamos un análisis de cómo podríamos administrar políticas de seguridad de firewalls.

6.7.1 ANALISIS PARA MANEJAR POLITICAS DE FIREWALLS.

La finalidad de la presente sección es mostrar como el administrador de políticas puede administrar políticas para firewall. Para lograr lo anterior empezaremos por dar una breve introducción de qué es un firewall y en especial hablaremos de los iptables. Posteriormente expondremos que se necesita desarrollar para que el administrador de políticas pueda administrar este tipo de políticas y por último daremos un ejemplo para mostrar como quedaría la formalización de estas políticas .

6.7.1.1 FIREWALL.

Un firewall es un dispositivo que filtra el tráfico de redes. El firewall puede ser un dispositivo físico o un software sobre un sistema operativo. En general se puede ver como una caja con dos o más interfaces de red en las que se establecen unas reglas de filtrado con las que se decide si una conexión determinada puede establecerse o no.

En sí, se puede definir un firewall como un hardware con un sistema operativo o una interfaz gráfica que filtra el tráfico TCP/UDP/ICMP... etc. y decide si un paquete pasa, se modifica, se convierte o se descarga.

6.7.1.2 IPTABLES.

Iptables es un sistema de firewall vinculado al kernel de Linux que se ha extendido a partir del kernel 2.4 del sistema operativo. Al lo igual que el anterior sistema ipchains, un firewall de iptables no es un servidor que se inicia o detiene según convenga, lo que provocaría errores de

programación. En lugar de esto, iptables está integrado con el kernel y es parte del sistema operativo. Iptables es un simple script de shell en el que se van ejecutando las reglas del firewall.

La estructura de iptables es básicamente una cola, cuando un paquete llega, éste es validado contra cada una de las reglas del firewall, en el momento que una regla coincide, se ejecuta la acción que haya sido definida en la política.

El tipo de políticas que se utilizan por omisión son: **prohibitiva**, si todo lo que no está expresamente permitido está denegado, o **permisiva**, si todo lo que no está expresamente prohibido está permitido.

La estructura del comando de iptables es la siguiente:

iptables –t[tabla] – [opción básica] [regla o política] [criterio] –j [acción a realizar]

En donde:

-t[tabla] Se especifica cual es la tabla que queremos añadir a la regla. Existen tres tipos de tablas válidas: Nat, filter y magle, siendo filter la tabla por defecto si se omite esta parte del comando. **Nat** se refiere a las conexiones que serán modificadas por el firewall, como por ejemplo enmascarar conexiones, realizar redirecciones de puertos, etc. **Filter** es la tabla en la cual se añade las relaciones de filtrado. **Magle** modifica cualquier aspecto del paquete.

– [opción básica] [regla o política] Hay cuatro opciones básicas: **A** es añadir una regla, la cual puede ser INPUT, FORWARD y OUTPUT. **L** es para listar las reglas. **F** es para borrar las reglas o en el caso de INPUT, FORWARD o OUTPUT sean dados como argumento se borrarán las reglas asociadas solo a esa clase. **P** establece las políticas por default del firewall.

- [criterio] Aquí es donde se especificarán las características del tipo de paquete que coincidirá con esta regla. Para establecer las reglas operamos con las siguientes opciones:

- s dirección ip/red fuente
- d dirección ip/red destino
- sport puerto fuente

-dport puerto destino
- p protocolo

–j [acción a realizar] Se establece la acción a realizar con el paquete. Las posibles opciones son:

Accept. Se acepta el paquete
Reject o Drop. Se desecha el paquete.
Redirect. Redirigirá el paquete a donde lo indique la política
Log. Lo guardará para su posterior análisis.

6.7.1.3 SOLUCIÓN PROPUESTA.

Para que el administrador de políticas pueda administrar este tipo de políticas se deberán programar tres módulos. El primer módulo tiene como objetivo guardar en la base de datos las políticas así como sus elementos principales. Para lograr lo anterior, este módulo leerá las políticas (iptables) a partir de un archivo de texto para posteriormente almacenar cada una de las partes que conforman el ambiente en una tabla dentro de la base de datos así como almacenar las políticas en otra tabla dentro de la misma.

El segundo módulo extraerá los datos para generar el ambiente formalizado (E_0 y E_1) de acuerdo con los lineamientos de Halpern y Weissman. Como ambiente E_0 estaremos considerando las direcciones Ip fuente, direcciones Ip destino, el puerto fuente, el puerto destino y los protocolos. La parte del ambiente E_1 estará formado por las reglas que se establecen por default. Es decir, las políticas que indican si el firewall es de tipo prohibitivo o permisivo. Y por último, el tercer módulo traducirá las políticas en lógica de primer orden siguiendo los lineamientos de Halpern y Weissman.

Para una mejor entendimiento de lo anterior, analizaremos el siguiente ejemplo: Un equipo Linux esta conectado a Internet y se desea proteger con su propio firewall (iptables). El archivo de script de entrada será el siguiente:

```
#!/bin/sh
## SCRIPT de IPTABLES
## Ejemplo de script para proteger la propia máquina
```

```
echo −n Aplicando Reglas de Firewall...

## Establecemos política por defecto (Tipo permisivo)
iptables −P INPUT ACCEPT
iptables −P OUTPUT ACCEPT
iptables −P FORWARD ACCEPT
iptables −t nat −P PREROUTING ACCEPT
iptables −t nat −P POSTROUTING ACCEPT

## Empezamos a filtrar
# A nuestra IP le dejamos que acepte todos los paquetes
iptables −A INPUT −s 195.65.34.234 −j ACCEPT

# A orto equipo le dejamos entrar al mysql para que mantenga la BBDD
iptables −A INPUT −s 231.45.134.23 −p tcp −−dport 3306 −j ACCEPT

# A un diseñador le dejamos usar el FTP
iptables −A INPUT −s 80.37.45.194 −p tcp −dport 20:21 −j ACCEPT

# El puerto 80 de www debe estar abierto, es un servidor web.
iptables −A INPUT −p tcp −−dport 80 −j ACCEPT

# Y el resto, lo cerramos
iptables −A INPUT −p tcp −−dport 20:21 −j DROP
iptables −A INPUT −p tcp −−dport 3306 −j DROP
iptables −A INPUT −p tcp −−dport 22 −j DROP
iptables −A INPUT −p tcp −−dport 10000 −j DROP
echo " OK . Verifique que lo que se aplica con: iptables −L −n"

# Fin del script
```

Utilizando el primer módulo del administrador de políticas, este archivo es leído. Y en la base de datos se tendrá una tabla llamada db_Ambiente en donde se almacenarán cada uno de los componentes del ambiente y otra tabla llamada db_políticas la cual almacenará cada una de las políticas (iptables).

Posteriormente, el segundo módulo extraerá cada uno de los elementos de la tabla db_Ambiente para generar el ambiente (E_0 y E_1) formalizado. Para realizar dicha formalización, el administrador de políticas se basará en los lineamientos de Halpern y

Weissman, tal y como lo hace con las políticas de la información expuestas en esta tesis. Por tal motivo para formalizar el ambiente E_0, se considerará lo siguiente:

```
DireF(xx_xx_xx_xx) lo cual significa que xx_xx_xx_xx es la dirección fuente.
DireD(xx_xx_xx_xx) lo cual significa que xx_xx_xx_xx es la dirección destino.
PuertoF(xx:yy) lo cual significa que xx:yy es el puerto origen.
PuertoD(xx:yy) lo cual significa que xx:yy es el puerto destino.
Proto(x) lo cual significa que x es el protocolo a utilizar.
```

En nuestro caso el ambiente E_0 quedaría:

```
%Dirección Ip fuente
DireF(195_65_34_234). % ip propia
DireF(231_45_134_23). % ip del compañero
DireF(80_37_45_194). % ip del diseñador

%Protocolo
Proto(tcp). %El protocolo que se filtrará.

%Puertos Destino
PuertoD(3306).
PuertoD(20).
PuertoD(21).
PuertoD(80).
PuertoD(22).
PuertoD(10000).
```

Posteriormente, el administrador de políticas formaliza el ambiente E_1. En nuestro caso el ambiente E_1 esta formado por las políticas por default.

```
%Axiomas de igualdad
all x (x = x). % Reflexividad
all x y (x=y -> y=x). % Simetría
all x y z (x=y & y=z -> x=z). % Transitividad

% Utilizando Unique Names Assumption
all x y (x = y) -> (DireF(x) = DireF(y)).
all x y (x = y) -> (DireD(x) = DireD(y)).
all x y (x = y) -> (PuertoF(x) = PuertoF(y)).
```

```
all x y (x = y) -> (PuertoD(x) = PuertoD(y)).
all x y (x = y) -> (Proto(x) = Proto(y)).

%Todo equipo tiene una dirección ip propia
all x z  (Tiene(x, z) -> Equipo(x) & DireF(z)).

%Todo paquete pasa en un puerto
all y z (Esta(y,z) -> Paquete(y)  & PuertoF(z)).

%Reglas default para la tabla filter

all x y (Tiene(x,195_65_34_234)  & Paquete(y) -> Debe(x,Recibir(y))).
all x y (Tiene(x,195_65_34_234)  & Paquete(y) -> Debe(x,Enviar(y))).
all x y (Tiene(x,195_65_34_234)  & Paquete(y) -> Debe(x,Redireccionar(y))).

%Reglas default para la tabla nat
all x y (Tiene(x,195_65_34_234)  & Paquete(y) -> Debe(x,Prerutear(y))).
all x y (Tiene(x,195_65_34_234)  & Paquete(y) -> Debe(x,Postrutear(y))).
```

Una vez que el administrador de políticas formaliza el ambiente entonces puede formalizar las políticas de la siguiente forma:

```
% A nuestra IP le dejamos todo
all  x  y (Tiene(x,195_65_34_234) & Paquete(y) -> Debe(x,Recibe(y))).

% A otro equipo le dejamos entrar al mysql para que mantenga la BBDD
all  x  y (Tiene(x,231_45_134_23) & Proto(tcp) & PuertoD(3306) & Paquete(y)
-> Debe(x,Recibir(y))).

% A un diseñador le dejamos usar el FTP
all  x  y (Tiene(x,80_37_45_194) & Proto(tcp) & (PuertoD(20) |
PuertoD(21)) & Paquete(y) -> Debe(x,Recibir(y))).

% El puerto 80 de www debe estar abierto, es un servidor web.
all  x  y (Esta(x,80) & Proto(tcp)  & Paquete(y) -> Debe(x,Recibir(y))).

% Y el resto, lo cerramos
all  x  y ((Esta(x,20) | (Esta(x,21))  & Proto(tcp) & Paquete(y) -> -
Debe(x,Recibir(y)))).

all  x  y (Esta(x,3306)  & Proto(tcp) & Paquete(y) -> -
Debe(x,Recibir(y))).
```

```
all  x  y (Esta(x,22)  & Proto(tcp) & Paquete(y) -> - Debe(x,Recibir(y))).

all  x  y (Esta(x,10000)  & Proto(tcp) & Paquete(y) -> -
Debe(x,Recibir(y))).
```

Otro punto que quedaría para trabajo futuro sería el poder manejar excepciones en las políticas generales ya que hasta este momento el administrador de políticas únicamente maneja excepciones en políticas particulares (específicamente políticas de control de acceso sobre archivos y control de acceso sobre directorios). En la siguiente sección explicamos brevemente como se podrían solucionar las excepciones en políticas generales.

6.7.2 MANEJO DE EXCEPCIONES EN LAS POLÍTICAS GENERALES.

Como se ha comentado a lo largo de esta tesis, el administrador de políticas puede escribir políticas generales. Dichas políticas se escriben en base a un rol que se tenga en la organización, por ejemplo se puede escribir una política que diga "Todos los gerentes de la empresa pueden leer el archivo de passwords".

Sin embargo, como sabemos las empresas están en constante cambio por lo los permisos que tienen los gerentes sobre el archivo de passwords puede cambiar de acuerdo a las necesidades de la organización. Por ejemplo, supongamos que nuestra empresa ha tenido algunos cambios organizacionales y en este momento ya no se desea que el Gerente de Recursos humanos pueda leer el archivo de passwords.

Si en este momento se escribe en el administrador una política que indique que el Gerente de Recursos humanos no puede leer el archivo de passwords se generaría una inconsistencia en las políticas. Esta inconsistencia se debe a que tenemos en el administrador de políticas primero una política general que nos indica que "Todos los gerentes pueden leer el archivo de passwords" y después introducimos la política que indica que el Gerente de Recursos humanos no puede leer el archivo de passwords.

Para evitar lo anterior proponemos dos soluciones. La primera solución propuesta consiste en primero borrar las políticas que están generando la inconsistencia. Una vez borradas el

administrador del sistema introduciría una política por cada puesto (sujeto). Por ejemplo en nuestro caso escribiríamos que el Gerente de Finanzas, Sistemas y Seguridad pueden leer el archivo de passwords. Y posteriormente, indicaríamos mediante una cerradura que el gerente de Recursos humanos no puede leer el archivo de passwords. Lo anterior quedaría formalizado de la siguiente forma:

```
all x z (Puesto(x,Gerente,Finanzas) ->
Puede(x,Leer(Archivo_de_passwords))).

all x z (Puesto(x,Gerente,Sistemas) ->
Puede(x,Leer(Archivo_de_passwords))).

all x z (Puesto(x,Gerente,Seguridad) ->
Puede(x,Leer(Archivo_de_passwords))).

all x z (Puesto(x,Gerente,z) & z!= Recursos_Humanos -> -
Puede(x,Leer(Archivo_de_passwords))).
```

Esta solución tiene como desventaja que el administrador del sistema tardaría mucho tiempo en escribir cada política por sujeto involucrado sobre todo si la organización es muy grande. Por tal motivo proponemos como segunda solución el crear un submódulo dentro del módulo en el cual se crean y/o modifican las políticas generales en la herramienta. Este submódulo tiene como finalidad el crear una excepción de la política general. Dicha excepción se puede realizar al momento de crear la política (módulo de *generación de políticas generales*) o después de un tiempo de ser creada (módulo *modificar políticas generales*). En este submódulo de *excepciones* el administrador del sistema deberá seleccionar el o los puesto(s) (incluyendo el departamento) del o de los sujeto(s) que no podrá(n) realizar dicha acción.

Para un mejor entendimiento de esta solución retómenos el ejemplo anterior y supongamos que en este momento generamos por primera vez la política "Todos los gerentes de la empresa pueden leer el archivo de passwords" y posteriormente seleccionamos el submódulo de *excepciones* para indicar que el Gerente de Recursos humanos no puede leer el archivo de passwords. Una vez creadas estas dos políticas, el administrador de políticas formalizaría la política y creará una cerradura por la excepción seleccionada. Nuestro ejemplo, queda formalizado de la siguiente forma:

```
Política:
all x z (Puesto(x,Gerente,z) -> Puede(x,Leer(Archivo_de_passwords))).

Cerradura:

all x z (Puesto(x,Gerente,z) & z!= Recursos_Humanos -> -
Puede(x,Leer(Archivo_de_passwords))).
```

Ahora, supongamos que se desea introducir una nueva excepción la cual consiste en indicar que el gerente de sistemas tampoco puede leer el archivo de passwords. Entonces el administrador del sistema entraría al módulo de *modificar políticas generales* y al submódulo de *excepciones* y daría de alta dicha excepción. Una vez concluido lo anterior, el administrador de políticas únicamente modificaría la cerradura la cual quedaría formalizada de la siguiente forma:

```
all x z (Puesto(x,Gerente,z) & (z!= Recursos_Humanos | z!= Sistemas) -> -
Puede(x,Leer(Archivo_de_passwords))).
```

Por último, consideramos importante mencionar que como una aplicación futura que se puede realizar con las tablas de verdad que nos construye Mace, es crear un sistema de alertas (un IDS de políticas). Dicho sistema de alertas consistiría en un servidor en el cual estén almacenadas las tablas de verdad que representan a cada política y se tendrían agentes inteligentes que estarían viajando en la red y verificado que dichas políticas se cumplan. En caso de que algún usuario viole alguna política, el sistema mandará una alerta al administrador de seguridad, para que él intervenga. Esta idea[10] sin duda, habría que analizarla y ver de qué forma puede ser viable.

Debido a que las necesidades de las empresas en materia de políticas de seguridad están constantemente en aumento y se vuelven más complejas, creemos que en un futuro se deberá desarrollar una herramienta que cubra estos nuevos requerimientos. Por tal motivo, esperamos que nuestra aportación sea la base para desarrollar en un futuro dicha herramienta.

[10] Idea original del Dr. Jesús Vázquez

APÉNDICE A : OTTER (DEMOSTRADOR DE TEOREMAS).

INTRODUCCIÓN.

La demostración de teoremas automáticos trabaja en el desarrollo de programas computacionales que demuestran que alguna declaración (una conjetura) es una consecuencia lógica de un conjunto de declaraciones (axiomas e hipótesis).

Un demostrador de teoremas automáticos para lógica de primer orden, consiste en un procedimiento o programa que puede usarse para demostrar que dada una fórmula objetivo, ésta es derivada a partir de una teoría de primer orden, normalmente representado por un conjunto de axiomas.

Los demostradores de teoremas automáticos pueden aplicarse a un gran número de áreas de investigación, principalmente en las Ciencias Computacionales, algunas de estas aplicaciones son: generación de software, verificación formal de software, verificación de hardware y representación de conocimiento

La mayoría de los sistemas demostradores de teoremas automáticos son divididos en dos categorías, de acuerdo a su modo de evaluación o a la forma en que generan las cláusulas, la primera categoría es el sistema *Top-down*, el cual inician desde un objetivo reduciéndolo en subobjetivos hasta que éstos son tratados como axiomas. Dichos sistemas se ven como un árbol de búsqueda y usualmente exploran una sola rama y la siguiente, es el sistema *Bottom-up* que normalmente tratan con una base de datos de cláusulas y generan nuevas cláusulas aplicando reglas de inferencia a dichas cláusulas.

La mayoría de los demostradores de teoremas desarrollados para lógica de primer orden trabajan mediante el uso de reglas de inferencia basadas en resolución y paramodulación. El método de resolución incluye resolución binaria, resolución-UR e hiper resolución.

La resolución binaria es la regla de inferencia que toma dos cláusulas, selecciona una literal en cada una de las cláusulas que corresponde al mismo predicado, pero de signo diferente, y produce una cláusula proporcionando dos literales seleccionadas unificadas.

La resolución-UR es la regla de inferencia que se aplica a un conjunto de cláusulas, una de las cuales debe ser una cláusula no unitaria, el resto deben ser cláusulas unitarias. Además la cláusula no unitaria debe contener exactamente una literal más que el número de (no necesariamente distinto) las cláusulas unitarias en el conjunto al cual la resolución UR es aplicable.

Hiperresolución es la regla de inferencia que se aplica a un conjunto de cláusulas, una de las cuales debe ser una cláusula negativa o una cláusula mixta (negativa y positiva), el resto deben ser cláusulas positivas y su número debe ser igual al número de literales negativas en la cláusula negativa o la cláusula mixta.

Paramodulación es la regla de inferencia que se aplica a un par de cláusulas y requiere que al menos una de las dos contenga una literal de igualdad positiva. Esta regla produce una cláusula en la cual la sustitución de igualdad corresponde a la literal de igualdad que ocurrió.

Actualmente, existen varios demostradores de teoremas automáticos. Sin embargo, el demostrador de teoremas utilizado en la herramienta de seguridad realizada fue Otter. A continuación se explica cómo trabaja Otter.

OTTER.

Otter es un demostrador de teoremas escritos en lógica de primer orden con igualdad. Las reglas de inferencia de Otter se basan en resolución Binaria, resolución-UR, hiperresolución, y paramodulación.

Este sistema incluye facilidades para re-escritura de términos, ordenamiento de términos, terminación de *Knuth-Bendix*, y estrategias para dirigir y restringir búsquedas de prueba. Otter maneja cuatro listas de cláusulas:

- Lista "usable": Contiene las cláusulas que están disponibles para hacer inferencia.
- Lista "sos": Son las cláusulas de soporte que no están disponibles para hacer inferencia, estas cláusulas sólo esperan participar en las búsquedas.
- Lista pasiva: Estas cláusulas no participan directamente en las búsquedas, son usadas para las siguientes búsquedas y unificación de conflictos.
- Lista demoduladora: Son igualdades que son usadas como reglas para escribir una nueva cláusula inferida.

El funcionamiento general de Otter [13] es el siguiente:

Las reglas de inferencia de Otter toman un pequeño conjunto de cláusulas e infieren una cláusula, si la cláusula inferida es nueva y usable, ésta es almacenada y podrá estar disponible para inferencias subsecuentes.

Los enunciados del problema pueden ser capturados como fórmulas de primer orden o como cláusulas (una cláusula es una disyunción con cuantificadores universales implícitos). Si se ingresan fórmulas de primer orden, Otter transforma éstas a cláusulas.

El proceso de demodulación hacia adelante re-escribe y simplifica cláusulas inferidas con un conjunto de igualdades, y la demodulación hacia atrás usa una igualdad inferida para re-escribir todas las cláusulas existentes.

La clasificación hacia adelante borra una cláusula inferida si ésta es clasificada por cualquier cláusula existente, y la clasificación hacia atrás borra todas las cláusulas que son clasificadas por una cláusula inferida.

Otter trabaja de modo autónomo, sin embargo algunos de los trabajos realizados con Otter involucran interacción con los usuarios. Después de codificar un problema en lógica de primer orden o en forma de cláusulas, el usuario normalmente cambia las reglas de inferencia, el conjunto de opciones para controlar el procesamiento de las cláusulas inferidas, y decide que fórmulas o cláusulas de entrada estarán en el conjunto inicial de soporte. Si Otter falla en

la búsqueda de una prueba, el usuario puede intentar otra vez con condiciones iniciales diferentes.

En el modo autónomo, el usuario ingresa un conjunto de cláusulas y/o fórmulas y Otter hace un análisis sintáctico simple y decide las reglas de inferencia y estrategias que utilizará en la búsqueda de la prueba. El modo autónomo es normalmente útil en un primer intento de la prueba.

Para la realización de las validaciones de las políticas de seguridad, se tuvo que realizar una configuración especial en las banderas de Otter. Dicha configuración, se describe a continuación.

CONFIGURACIÓN NECESARIA PARA OTTER EN LA HERRAMIENTA DE SEGURIDAD

Primeramente, para realizar las validaciones de las políticas descritas en el capítulo cinco de esta tesis, estuvimos trabajando con Otter en modo autónomo. Sin embargo, al ir haciendo las pruebas, Otter tardaba mucho tiempo (más de 30 minutos) en dar una respuesta. Por tales motivos e investigando en el manual [13] se encontró que en lugar de trabajar en modo autónomo se podía trabajar en modo auto2 (activando la bandera set (auto2)).

Utilizando esta bandera, Otter redujo su tiempo de respuesta de varios minutos (hasta horas) a microsegundos. Esto se debe principalmente a que se habilitan algunas otras banderas o les asigna algún valor especial de tal forma que acota la búsqueda haciendo que se encuentre una respuesta más rápido. A continuación, se explica que banderas son activas y que valores se les asigna.

Una de las banderas que se activa al utilizar el modo auto2 es *PROCESS_INPUT*. El activar esta bandera produce que las cláusulas de entrada en la lista “usable” y en la lista de soporte (incluyendo las cláusulas de la formula de entrada) sean procesadas como si hubieran sido generadas por una regla de inferencia.

La siguiente bandera que se activa es *PICK_GIVEN_RATIO*. Esta bandera al activarse origina que algunas cláusulas dadas sean seleccionadas por pesos y otras mediante una búsqueda en

anchura. Si su valor es menor o mayor a -1, entonces las cláusulas dadas son seleccionadas por peso.

Con el modo auto2, se le asigna el valor de 1 a la bandera *STATS_LEVEL*, esto indica que sólo las estadísticas[11] importantes de búsqueda y tiempo serán escritas. Este parámetro no afecta a la velocidad de Otter, ya que todas las estadísticas son siempre almacenadas.

[11] Las estadísticas que Otter entrega es un reporte que se muestra en el momento que Otter se detiene. Este reporte incluye los tiempos que tardo en realizar las búsquedas, el número de cláusuas generadas, la cantidad de memoria utilizada, etc.

REFERENCIAS.

[1] International Standards Organization. Information Processing Systems - OSI RM.

[2] Technical Report 97 7498-2, ISO/TC, 1988. *Part 2: Security Architecture.* Sead Muftic, Ahmed Patel, Peter Sanders, Rafael Colon, Jan Heijnsdijk, and Unto Pulkkinen. *Security in Open Systems*. John Wiley and Sons, 1993.

[3] Reynolds, J. *Site Security Handbook*. July 1991

[4] SPAFFORD, Gene. *Manual de seguridad en redes*. Acert. Argentina 2000.

[5] PARKER ,Donn B. Demonstrating the elements of information security with threats. *In Proceedings of the 17th National Computer Security Conference*, pag 421-430, 1994

[6] 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.

[7] FERNANDEZ, Carlos. *Seguridad en sistemas informáticos.* Ediciones Diaz de santos s.a. España, 1988

[8] GARCIA ,Karen. *Administrador de políticas de seguridad organizacionales*. Instituto Tecnológico y de Estudios Superiores de Monterrey. Estado de México. 2005

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

[10] ContentGuard. *XrML: The digital rights language for trusted content and services.* http://www.xrml.org/, 2001.

[11] R. Iannella. *ODRL: The open digital rights language initiative.*http://odrl.net/, 2001.

[12] Richard Char-Tung Lee Chin-Liang Chang. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.

[13] McCUNE, W. Otter 3.3. Reference Manual. Argonne National Laboratory, 9700 South Cass Avenue Argonne, IL 60439, pages 1-11, August 2003.

[14] HALPERN, Joseph Y.. *Using First-Order Logic to Reason about Policies.* Cornell University Ithaca, NY 14853

[15] McCUNE, W. *Mace 2.0. Reference Manual and Guide.* Argonne National Laboratory, 9700 South Cass Avenue Argonne, IL 60439, August 2003.

[16] ARENAS, Loudes. *Lógica Formal para informáticos*. Ed. Alianza. 1995.

[17] McCUNE, W. *David-Putman Program and it´s application to finite first-order model seach: Quasigroup existence Problems.* Argonne National Laboratory, 9700 South Cass Avenue Argonne, IL 60439, August 2003.

[18] W. McCune and O. Shumsky. IVY: *A preprocessor and proof checker for first-order logic.* In M. Kaufmann, P. Manolios, and J Moore, editors, *Computer-Aided Reasoning: ACL2 Case Studies*, chapter 16. Kluwer Academic, 2000.

[19] QUINTANA, M: *Complejidad computacional.* Apuntes del curso de Fundamentos de la computación. ITESM Estado de México. 2004.

[20] FEHILY, C. *Visual Quickstart guide SQL.* Peacfpit press. 2002

[21] W. McCune and O. Shumsky. IVY: *A preprocessor and proof checker for first-order logic.* In M. Kaufmann, P. Manolios, and J Moore, editors, *Computer-Aided Reasoning: ACL2 Case Studies*, chapter 16. Kluwer Academic, 2000.

[22] GASSER, Morrier, B*uilding a secure computer system.* Charper 9.1998

[23] BISHOP, Matt, *Computer security*, Addison-wesley , 2003 pags 124 – 127.

[24] …… pags 160 – 165.

[25] …… pág 175

[26] H. Putnam M. Davis. *A computing procedure for quantification theory: Its justification and realization.* J. ACM, pages 201–215, 1960.

[27] Larry Wos et al. *Automated reasoning. Introduction and applications* Prentice-hall, inc. pages 160–178, 1984.